

https://dspace.org/

Mitigating class co-occurrence ambiguity in weakly supervised semantic segmentation with cross image attention

Hill, William Skipsey

2025-09
University of the Witwatersrand, Johannesburg
https://hdl.handle.net/10539/50287
Hill, William Skipsey. (2025). Mitigating class co-occurrence ambiguity in weakly supervised semantic segmentation with cross image attention. [Master's dissertation, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/50287

Downloaded from a DSpace repository. https://dspace.org/

Mitigating class co-occurrence ambiguity in weakly supervised semantic segmentation with cross image attention

School of Computer Science & Applied Mathematics
University of the Witwatersrand

William Hill
2115261

Supervised by Prof. Richard Klein and Dr Hairong Bau

September 4, 2025



A dissertation submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science

Abstract

Weakly Supervised Semantic Segmentation (WSSS) has gained traction as a cost effective alternative to fully supervised segmentation, which requires extensive pixel-level labels. The primary challenge in WSSS is generating accurate segmentation masks from only image-level labels. While various approaches have made progress, the issue of class co-occurrence, where certain classes frequently appear together within the same image, remains a significant obstacle. Unlike traditional methods that focus on separating objects from the background, we address the more challenging problem of disambiguating co-occurring classes, such as "baseball bat" and "baseball glove" or "mouse" and "keyboard".

We empirically demonstrate that class co-occurrence is a critical factor degrading segmentation performance, especially for datasets where co-occurrence is prevalent, such as MS COCO 2014. In contrast, the PASCAL VOC 2012 dataset, which exhibits less co-occurrence, shows minimal performance degradation. This discrepancy highlights that co-occurrence is a dataset-dependent challenge, overlooked in previous studies focusing primarily on PASCAL VOC 2012.

To address this problem, we propose a novel dual-branch transformer architecture built upon MCTFormer+, where one branch processes the primary image and the other processes an auxiliary image containing a complementary class. The two branches interact via a new cross-attention mechanism called Cross Image Attention from the Auxiliary to Primary branch (CIA-AP), which selectively augments the primary image's attention maps using information from the auxiliary image. This approach effectively improves segmentation accuracy for co-occurring classes by enhancing the model's ability to distinguish between them.

Furthermore, we introduce CCT Prime, a new contrastive learning module designed to align class token embeddings between the two branches, ensuring consistent representations for effective information transfer. This alignment proves essential for enhancing performance when co-occurring classes are present, while still maintaining competitive overall performance.

Experiments show substantial improvements for co-occurring classes while maintaining competitive mean Intersection over Union (mIoU). Our findings highlight the effectiveness of leveraging cross-image information to address class co-occurrence, providing a promising direction for future WSSS research.

Declaration

I, William Hill, hereby declare the contents of this research dissertation to be my own work. This dissertation is submitted for the degree of Master of Science in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Acknowledgements

Computations were performed using High Performance Computing infrastructure provided by the Mathematical Sciences Support unit at the University of the Witwatersrand.

Contents

Preface
Abstract i
Declaration ii
Acknowledgements iii
Table of Contents iv

1 Introduction **1**

2 Background and Related Work **5**
2.1 Weakly Supervised Semantic Segmentation 6
2.2 Metrics 6
2.2.1 Mean Intersection over Union 7
2.2.2 Multi-Label Soft Margin Loss 7
2.3 CNN-based Approaches 8
2.3.1 Class Activation Maps (CAMs) 8
2.3.2 Adversarial Erasing 10
2.3.3 Memory Banks 11
2.4 Saliency Maps 11
2.5 Post-processing 12
2.5.1 Random Walk 12
2.5.2 Conditional Random Field 13
2.6 Concepts Discussed in Later Chapters 14
2.7 Conclusion 15

3 Class Co-Occurrence **16**
3.1 Definition 17
3.2 Data 18
3.2.1 PASCAL VOC 2012 Dataset 18
3.2.2 MS COCO 2014 Dataset 18
3.2.2.1 Constructing Ground-Truth Pixel Masks 19
3.3 Related Work 21
3.4 Quantification 23
3.4.1 Dataset Class Co-Occurrence Levels 24
3.4.2 Performance with Class Co-occurrence 25
3.5 Summary 30

4 Final Framework **31**
4.1 Background . . . 32
4.1.1 Vision Transformers . . . 32
4.1.1.1 Basic Mechanics . . . 32
4.1.1.2 Class and Patch Tokens . . . 33
4.1.1.3 Positional Embeddings . . . 33
4.1.1.4 Self-attention . . . 34
4.1.1.5 Multi-Headed Self-Attention (MHSA) . . . 35
4.2 Related Work . . . 36
4.2.1 Cross Attention . . . 37
4.2.2 Semantic Reassignment . . . 39
4.2.3 Efficient Training of Transformers . . . 39
4.3 Base Model . . . 40
4.3.1 MCTFormer . . . 40
4.3.2 MCTFormer+ . . . 43
4.3.3 Attention Map Extraction . . . 44
4.3.4 Attention Map Refinement . . . 45
4.3.5 Reimplementation . . . 46
4.4 Experimental Details . . . 47
4.5 Our Approach . . . 49
4.5.1 Final Cross Image Attention . . . 49
4.5.1.1 Mechanics . . . 52
4.5.1.2 Mechanics Example . . . 53
4.5.2 CCT Prime . . . 53
4.5.3 Loss Function . . . 55
4.5.4 Results . . . 56
4.5.5 Comparison to State-of-the-Art . . . 66
4.6 Summary . . . 69

5 Cross Image Attention Ablation **72**
5.1 Siamese Weights . . . 73
5.2 Evolution of Cross Image Attention and Auxiliary Image Mining . . . 74
5.2.1 Conventional Cross Attention . . . 76
5.2.2 Impact of Overlap Task . . . 79
5.2.3 Explicit Cross Image Attention at a Single Layer . . . 82
5.2.4 Explicit Cross Image Attention at Multiple Layers . . . 85
5.2.5 Critical Analysis of Cross Image Attention . . . 88
5.2.6 Principled Revision of Cross Image Attention . . . 89
5.2.7 CIA-AP Query Augmentation Methods . . . 96
5.2.8 CIA-AP Layer Choices . . . 102
5.2.9 Third Branch . . . 104
5.3 Inference . . . 107
5.3.1 Cross Image Attention Strategy . . . 107
5.3.2 Layers for Extracting Attention Maps . . . 109
5.4 Summary . . . 111

6 CCT Prime: Class Token Alignment with Contrastive Learning **112**
6.1 Background . . . 113
6.1.1 Cross Entropy Loss . . . 113
6.2 Related Work . . . 114
6.2.1 Metric Learning (Distance Measures) . . . 114
6.2.2 Contrastive Learning Approaches . . . 115
6.3 Contrastive Learning in our Dual-Branch Framework . . . 115
6.3.1 Contrastive Class Tokens (CCT) . . . 116
6.3.2 CCT Behaviour . . . 117
6.3.3 CCT Initial Performance . . . 120
6.3.4 CCT Prime . . . 121
6.3.4.1 Initial Experimental Results . . . 127
6.3.5 Contrastive Learning with CIA-AP . . . 128
6.4 Summary . . . 133

7 Conclusion and Future Work **134**
7.1 Future Work . . . 134
7.1.1 Investigating Trade-Off for Co-Occurring Class Performance Increase . . . 134
7.1.2 Improving Cross-Attention Mechanisms . . . 135
7.1.3 Enhancing Contrastive Learning . . . 135
7.1.4 Improving Segmentation Model Performance . . . 136
7.1.5 Evaluation with COCO Masks . . . 136
7.1.6 Broader Applications and Domain Adaptation . . . 137
7.2 Conclusion . . . 137

References **147**

A Supporting Experimental Data **148**

Chapter 1

Introduction

Semantic Segmentation is a task with a wide range of real-world use cases: it is used in the first stage of many pipelines, including for self-driving cars and extracting regions of interest in medical imaging. Usually, there are high-quality segmentation masks available for accurate fully supervised learning. However, the Weakly Supervised variant has gained popularity as it does not require obtaining pixel-level labels, which is a time-consuming and expensive process. Various types of supervision have been investigated, including Scribbles [Lin *et al.* 2016], Points [Bearman *et al.* 2016], Bounding Boxes [Khoreva *et al.* 2017; Dai *et al.* 2015], and image-level (classification) labels [Ru *et al.* 2023; Xu *et al.* 2022; Zhou *et al.* 2022]. The latter is the most popular, due to the extensive availability of such labels. It is also the form we investigate in this work. Weakly Supervised Semantic Segmentation has not yet been solved with sufficient quality to warrant its use in real-world applications, although current performance shows promise and its potential for cost saving remains appealing.

The general approach to tackling Weakly Supervised Semantic Segmentation (WSSS) involves training a classifier using image-level labels and subsequently inspecting the classifier to determine which image features are associated with each class prediction. This knowledge is then employed to generate dense pixel-level pseudo labels, which are used to train a dedicated segmentation model for future inference. This methodology enables inference on new images even in the absence of image-level labels for the new images. Despite substantial progress, there remains significant room for improvement, with many works following similar solution trends that focus primarily on single-image approaches.

A notable challenge in WSSS is the co-occurrence problem, which refers to the phenomenon where certain classes frequently appear alongside specific aspects of an image, such as cars on roads or aeroplanes in the sky. However, unlike most works that focus on co-occurrence between classes and background features [Lee *et al.* 2021b; Su *et al.* 2021; Lee *et al.* 2022], we address a more challenging and previously underexplored form of co-occurrence: where classes from the dataset frequently occur together in the same image [Yang *et al.* 2024]. For instance, “baseball bat” and “baseball glove” or “mouse” and “keyboard” often co-exist, leading to incorrect associations between the classes during segmentation. When models are trained without explicit

separation of such co-occurring classes, there is classification confusion between the classes, which follows from the model struggling to disambiguate the co-occurring object regions, resulting in degraded segmentation performance. Co-occurrence of this nature has only recently become a prominent research focus. Previous works generally assumed co-occurrence was problematic and attempted to mitigate its effects without providing direct evidence of its impact on performance — a discussion of such works is provided in Section 3.3.

In this work, we go further by systematically quantifying the extent of class co-occurrence in the PASCAL VOC 2012 [Everingham *et al.* 2012] and MS COCO 2014 [Lin *et al.* 2014] datasets, which can be detected using the image-level labels alone. PASCAL VOC 2012 is the most commonly used dataset in the WSSS space, and we show that it is minimally affected by class co-occurrence. This relative lack of co-occurrence means that reported performance on PASCAL VOC 2012 largely overlooks the severity of the problem. In contrast, we demonstrate that on MS COCO 2014, where class co-occurrence is far more prevalent, segmentation performance drops substantially for affected classes. This evidence highlights that co-occurrence between classes is indeed a major factor contributing to degraded segmentation performance, particularly on datasets like MS COCO 2014 where the issue is more prominent. Addressing this problem is the primary goal of our research.

Most previous approaches focus on single-image methods that attempt to extract as much information as possible from the given image. However, there exists a wealth of information across the entire dataset that can be leveraged to improve segmentation performance for a given image. Regional Semantic Contrast and Aggregation (RCA) [Zhou *et al.* 2022] was the first work to utilise information from multiple images to enhance segmentation by learning a collective embedding for each class over multiple images. In addition, transformer-based approaches have recently achieved high-performing [Xu *et al.* 2024; Yang *et al.* 2024; Ru *et al.* 2023; Lin *et al.* 2023] and otherwise competitive results [Chen *et al.* 2023b; Cheng *et al.* 2023].

We hypothesise that the presence of co-occurring classes makes it challenging for the model to accurately attend to individual objects associated with those classes within an image. To address this, our approach leverages attention from an auxiliary image that does not suffer from the co-occurrence issue — specifically, an image containing only one of the co-occurring classes (e.g. an image of a mouse without a keyboard). By incorporating attention from such an image, the model learns to better disambiguate the co-occurring classes, allowing it to focus on more relevant object regions. Importantly, this method relies solely on the image-level labels already available within the WSSS framework, meaning it introduces no additional requirements or annotations beyond what is typically provided in existing datasets.

Building on these ideas, we propose a transformer-based approach using a shared-weight dual-branch transformer architecture building upon the high-performing model MCTFormer+ [Xu *et al.* 2024]. In our framework, the image to be segmented is passed through the primary branch, while an auxiliary image, chosen for its complementary content, is processed through an auxiliary branch. Information sharing is achieved via cross-attention mechanisms, which facilitate interaction between the attention mech-

anisms of each branch. This approach particularly emphasises encouraging improved attention to predetermined co-occurring pairs of classes. Following extensive investigations into conventional and novel cross attention schemes, we develop Cross Image Attention from the Auxiliary to Primary branch (CIA-AP), which applies cross-attention at specific transformer layers to augment certain elements of the Query matrix from the auxiliary branch into the Query matrix of the primary image. This promotes enhanced attention to co-occurring classes, thereby improving segmentation accuracy.

Another area of interest in the literature has been the use of contrastive learning to help class and patch tokens learn more representative embeddings [Xu *et al.* 2024; Ru *et al.* 2023; Yang *et al.* 2024]. This technique allows the tokens to better encode their class characteristics and image content. Our base model, MCTFormer+, employs a specific contrastive learning module called Contrastive Class Tokens (CCT), designed to induce an orthonormal basis over the multiple class tokens, encouraging them to learn more distinct representations. However, we show that despite the theoretical basis for orthonormality, the tokens instead align themselves in broad directional clusters, where tokens for present classes point in one general direction and those for absent classes point in a roughly opposite direction. Orthogonality is only observed when the classifier is uncertain about the existence of classes.

However, while MCTFormer+ employs CCT to promote orthonormality among class embeddings, strictly orthogonal embeddings are not necessary for the model to perform its task. Moreover, the model does not require these embeddings to be consistent across different images. When we introduce the auxiliary image in our dual-branch framework to enhance performance on co-occurring classes, it becomes essential to ensure consistency between the class embeddings of the primary and auxiliary images. Without such alignment, the information exchanged through cross attention could be ineffective or even detrimental. To address this, we introduce an additional contrastive learning module, CCT Prime, designed to align the class tokens between the primary and auxiliary branches. By ensuring consistency in the directional alignment of class tokens — where the "present" direction is shared between branches — CCT Prime enables more effective information transfer and mitigates the negative impact of inconsistent embeddings.

We use the Intersection over Union (IoU) as our primary evaluation metric, along with the mean IoU (mIoU) for overall performance assessment. Unlike previous works that primarily focus on improving mIoU, we also emphasise improving the IoU of specific co-occurring classes of interest.

The main contributions of this work are as follows:

1. We provide empirical evidence that co-occurrence is a critical factor contributing to degraded performance in Weakly Supervised Semantic Segmentation, particularly in the MS COCO 2014 dataset.

2. We introduce Cross Image Attention from the Auxiliary to Primary branch (CIA-AP), a novel cross-attention mechanism that enhances segmentation performance by augmenting attention to co-occurring classes through a dual-branch transformer architecture, built on MCTFormer+.

3. We propose CCT Prime, a contrastive learning module designed to align class token embeddings between the primary and auxiliary branches to promote consistent directional alignment.
4. We achieve substantial improvements in segmentation performance for co-occurring classes while maintaining competitive overall accuracy. Interestingly, the model seems to balance the substantial accuracy gains for co-occurring classes with a slight reduction in performance for non-co-occurring classes, resulting in a similar mIoU. This trade-off is noteworthy given that the model encounters non-co-occurring classes more frequently due to their higher prevalence in the datasets.

The remainder of this dissertation is structured as follows: Chapter 2 presents general WSSS concepts, including technical details of the metrics and post-processing techniques used, with a general discussion about related work in these fields and work otherwise pertaining to CNN-based approaches. The background and related work for co-occurrence, cross attention, and contrastive learning are discussed in their respective chapters, namely, Chapters 3, 4, and 6. Chapter 3 formalises co-occurrence, provides our quantification of it in the PASCAL VOC 2012 and MS COCO 2014 datasets, and demonstrates its impact on segmentation performance. Chapter 4 details the proposed cross-image attention mechanism, CIA-AP, and presents the final results. Chapter 5 presents justification for various design decisions for cross image attention through ablation studies. Chapter 6 investigates the behaviour of the existing CCT module and presents the additional CCT Prime module, along with its integration with CIA-AP. Finally, Chapter 7 suggests future research directions and concludes the dissertation.

Chapter 2

Background and Related Work

This chapter provides an overview of the foundational concepts and related work relevant to Weakly Supervised Semantic Segmentation (WSSS). While specific principles underlying cross attention and contrastive learning are discussed in dedicated chapters (Section 4.1 and Section 6.1, respectively), this chapter presents general motivation for commonly employed pipelines and methods within the weakly supervised setting.

The general strategy for WSSS involves generating pixel-level pseudo labels from weak supervision signals, which are then used to train a segmentation network in a supervised manner. While earlier methods predominantly relied on Convolutional Neural Network (CNN)-based frameworks [Wei *et al.* 2017; Zhang *et al.* 2018; Mai *et al.* 2020], recent advancements have increasingly adopted Vision Transformers as backbone architectures [Xu *et al.* 2022; Ru *et al.* 2023; Xu *et al.* 2024; Yang *et al.* 2024], achieving notable improvements in segmentation performance. Transformer-based approaches are centred around manipulating the attention mechanism to enhance segmentation quality, and relevant methodologies will be discussed in detail in Section 4.1 and Section 6.2.

We begin by outlining the motivation for WSSS and describing the general overview of approaches used to tackle this challenging problem (Section 2.1). Additionally, we review standard evaluation metrics used to assess performance in Section 2.2. Furthermore, commonly used post-processing techniques, such as Random Walk and Conditional Random Fields (CRF), are discussed in Section 2.5 to provide insight into how predictions are refined by encouraging spatial and semantic consistency.

While most transformer-based approaches involve concepts such as class co-occurrence (Chapter 3), cross attention (Chapter 4), or contrastive learning (Chapter 6), this chapter primarily focuses on methodologies that do not inherently rely on transformers. However, where relevant, we will revisit certain methods in later chapters to illustrate how their underlying ideas can be adapted or extended within transformer frameworks.

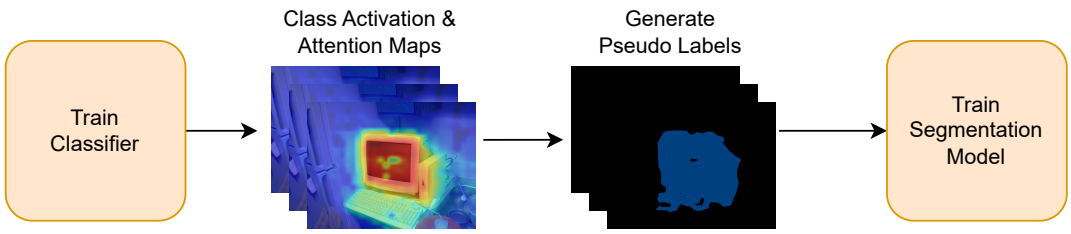


Figure 2.1: The general pipeline that is followed when performing weakly supervised semantic segmentation. Image-level labels are used to train a classifier, CAMs or Attention Maps are extracted to give an indication of where the model focuses when deciding about the existence of a class, which are then used to construct pixel-level pseudo labels for use in training a segmentation model.

2.1 Weakly Supervised Semantic Segmentation

Semantic Segmentation is the task of classifying every individual pixel in an image [Long *et al.* 2015]. Depending on the problem being considered, there may be one or more labels assigned to each pixel. The idea of Weakly Supervised Semantic Segmentation is borne from the fact that in order to train a Semantic Segmentation network, we will require pixel-level supervision in the form of a segmentation mask, which is expensive to obtain and may contain inaccuracies due to human error.

However, there exists a multitude of images with image-level labels which have normally been used for object recognition tasks. As such, we can train the network on the image classification task, but include extra processing to extract the areas of the image on which the network focuses when making its decision. For CNN-based approaches, this involves extracting Class Activation Maps (CAMs) (Section 2.3.1), and extracting Attention Maps for transformer-based approaches. These representations of where the model is focusing are used to construct pixel-level pseudo labels that can be used for training a segmentation model. This process is illustrated in Figure 2.1.

Therefore, the problem can be considered as one of fine-grained object localisation: we know the object is present, but we don't know where it is.

2.2 Metrics

The foremost metric used for quantifying the performance of semantic segmentation models (and weakly supervised variants) is the Intersection over Union, which is presented in Section 2.2.1.

Section 2.2.2 presents the main classification loss used in our work: Multi-Label Soft Margin Loss. We additionally use a Cross Entropy Loss for contrastive learning, and this is presented in Section 6.1.1.

2.2.1 Mean Intersection over Union

In general, the mean Intersection over Union (mIoU) is used as the performance measure for semantic segmentation tasks [Xu *et al.* 2024; Yang *et al.* 2024; Ru *et al.* 2023]. It is calculated by averaging the Intersection over Union (IoU) results for each class. IoU is calculated as follows:

$$\text{IoU}(A_c, B_c) = \frac{A_c \cap B_c}{A_c \cup B_c} = \frac{\text{Area of Overlap}}{\text{Area of Union}}, \tag{2.1}$$

where A_c is the predicted class-specific localisation map and B_c is the ground-truth class-specific localisation map for class c of an image. Note that our chosen datasets (presented in Section 3.2) have both image-level and pixel-level labels available: we use the image-level labels to provide a learning signal to train our framework, but only make use of the ground-truth segmentation masks when evaluating the framework's performance (i.e. the mIoU).

The class IoU scores are used to calculate the mIoU as follows:

$$\text{mIoU}(A, B) = \frac{1}{C}\sum_{i=0}^{C-1} \text{IoU}(A_c, B_c), \tag{2.2}$$

where A and B are the sets of all predicted and ground truth class-specific localisation maps, respectively. C is the number of classes.

We will use the IoU per class to get an idea of the performance improvements we get and what our failure cases are, and the mIoU as an overall performance measure which we use to compare against other works.

2.2.2 Multi-Label Soft Margin Loss

The multi-label soft margin loss is used in classification tasks where each instance can belong to multiple classes simultaneously [Xu *et al.* 2024]. It is a generalisation of the binary cross entropy loss, applied independently to each class. The loss function works by comparing the predicted probabilities for each class with the true binary labels, penalising incorrect predictions more as the difference between them grows. This is particularly useful for multi-label classification where the classes are not mutually exclusive.

Mathematically, the multi-label soft margin loss for a single image is:

$$\mathcal{L}_{\text{MLSML}} = -\frac{1}{C}\sum_{c=1}^{C} \left[y_c \log(\sigma(\hat{y}_c)) + (1 - y_c) \log(1 - \sigma(\hat{y}_c)) \right], \tag{2.3}$$

where $y \in \{0, 1\}^C$ is the image label (representing each class), $\hat{y} \in \mathbb{R}^C$ is the class score prediction (for each class), C is the number of classes, and σ is the sigmoid activation function which maps scalar values from $(-\infty, \infty)$ to $[0, 1]$.

We follow Xu *et al.* [2024] and use the multi-label soft margin loss as our classification loss, where we predict the existence of classes in images which may contain multiple different classes.

2.3 CNN-based Approaches

CNNs have been the preferred models for many Computer Vision problems, and were used to achieve state-of-the-art (at the time) on, and revolutionised the approach to, many well-known problems, such as image classification [Simonyan and Zisserman 2014; He *et al.* 2016], object detection [Redmon *et al.* 2016], semantic segmentation [Ronneberger *et al.* 2015], and instance segmentation [He *et al.* 2017]. Many attempts have been made to use CNN-based frameworks for WSSS, with some recent approaches achieving state-of-the-art [Zhou *et al.* 2022] on PASCAL VOC 2012 [Everingham *et al.* 2012] (however, they have since been outperformed by transformer-based approaches like MCTFormer+ [Xu *et al.* 2024] and SeCo [Yang *et al.* 2024]).

2.3.1 Class Activation Maps (CAMs)

It is difficult to interpret CNNs, since the learnt features get increasingly abstract when the network becomes deeper, making it a very nuanced problem. However, for the classification task (that is the overarching task for our weakly supervised research), methods, such as CAM [Zhou *et al.* 2016] and Grad-CAM [Selvaraju *et al.* 2017], have been devised to determine where the focus of the network is when it makes a classification decision, using the final Global Average Pooling layer. That is, Class Activation Maps (CAMs) form the basis of CNN-based approaches because it allows us to "see" where the classifier is focusing within an image when making its decision about the existence of certain classes.

The feature map produced by a convolutional layer in an image-based classification task typically has three dimensions: two spatial dimensions corresponding to the height and width of the feature map, and a third dimension corresponding to the channels. The number of channels is determined by the number of convolutional kernels applied in the convolutional layer. In image classification tasks, the final convolutional layer commonly outputs a feature map with as many channels as there are classes, where the intention is that each channel encodes the confidence of the corresponding class being present at each spatial location. It is important to note that, unless additional supervision is applied, the spatial locations in the feature map do not necessarily correspond directly to the original image locations; instead, they form a learned embedding.

Global Average Pooling (GAP) is a parameter-free down-sampling technique used to reduce the spatial dimensions of a feature map to a single value per channel, thereby producing a global representation of each channel. Unlike fully connected layers, which flatten the entire feature map, GAP computes the average of all spatial values (i.e. pixels) within each channel, resulting in a single scalar value per channel. This effectively represents the overall presence of each class within the image, making it particularly useful for image-level classification tasks. Mathematically, GAP performs for each channel, c:

$$y_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_{i,j,c}, \tag{2.4}$$

where $y_c \in \mathbb{R}$ is the class score (which may be further activated), and X is the feature map of size $H \times W \times C$ (H is the height, W is the width, and C is the number of channels or classes).

CAM [Zhou *et al.* 2016] operates by examining the final feature maps just before GAP is applied. The weights of the final layer (fully connected layer, or 1×1 convolutional layer in the case of fully-convolutional networks) associated with a particular class, c, are defined as w_c and have size C. Then the CAM for this class is obtained by:

$$\mathrm{CAM}_c(x, y) = \sum_{k=1}^{C} w_{c,k} \cdot F_k(x, y), \tag{2.5}$$

where $w_{c,k}$ is the weight corresponding to channel k for class c, and $F_k(x, y)$ is the feature map's activation at spatial position (x, y) in channel k. $\mathrm{CAM}_c \in \mathbb{R}^{H \times W}$ is the class activation map indicating the importance of each location for predicting class c, where high values are more strongly associated with that class. These CAMs are then upsampled to match the original image's resolution.

Grad-CAM [Selvaraju *et al.* 2017] is a generalisation of CAM that can be applied to a wider range of CNN architectures without requiring architectural modifications. Unlike CAM, which relies on a specific structure involving GAP layers to produce class activation maps, Grad-CAM operates on any CNN architecture, including those with fully connected layers. Instead of directly using network weights as in CAM, Grad-CAM utilises the gradients of the class score (logit) with respect to the feature maps of the last convolutional layer, which are obtained through backpropagation. These gradients act as importance weights, highlighting which parts of the feature map contribute most to the target class prediction. To obtain a single weight per channel, the gradients are globally averaged over the spatial dimensions. The resulting weights are then multiplied with the corresponding channels of the feature map, and the weighted feature maps are summed to produce a coarse localisation map for the desired class. To ensure only the regions that positively contribute to the class prediction are highlighted, a Rectified Linear Unit (ReLU) function is applied to the resulting map. Finally, similar to CAMs, the resulting Grad-CAM map can be upsampled to match the original image's resolution for visualisation purposes, providing valuable insight into which parts of the image influenced the model's decision.

However, Zhang *et al.* [2018] recognised that CAMs can be inconvenient to obtain, since they need to be computed in a post-inference manner. They mathematically proved that class localisation maps (akin to CAMs) can simply be extracted from the last convolutional layer by selecting the class-specific feature maps. This allows them to identify object regions of interest in a simple manner. However, CAMs still remain popular in later works.

In general, image classifiers tend to focus on the most discriminative parts of the object when deciding its class, since those are the regions that make the classification most certain. Consequently, the corresponding CAMs highlight only these discriminative regions, but fail to localise the entire object and are locally inconsistent.

Following MCTFormer+ [Xu *et al.* 2024], our approach directly utilises the normalised output feature maps from a convolutional layer as the Class Activation Maps (CAMs). This is made possible by the specific design of this convolutional layer, which maps the learned patch tokens from the transformer to a feature map with a number of channels equal to the number of classes. The resulting feature map is then directly used to predict class scores. This design allows the model to produce CAMs without requiring additional processing steps. A detailed explanation, including the mathematical formulation, is provided in Section 4.3.4.

2.3.2 Adversarial Erasing

Adversarial Erasing is the concept that was used to address CAMs highlighting only the most discriminative regions of the objects of interest. Adversarial Erasing aims to get the model to recognise less discriminative regions by removing ("erasing") the most discriminative regions (that it used to perform the classification). It does so by progressively localising and expanding object regions. This is basically a method of discovering a more global context for the object (i.e. learning what the object is on a global scale instead of a local one).

Three major works that introduced and developed the idea of adversarial erasing in a weakly supervised semantic segmentation setting were Wei *et al.* [2017], Zhang *et al.* [2018], and Mai *et al.* [2020]. Their methodology is based on the idea that the classifier needs to learn how to recognise other discriminative regions when its previous markers have been removed. The final localisation maps are obtained by aggregating the mined regions.

Wei *et al.* [2017] were the first to use this concept. They had a classification network to identify object regions of interest (using CAMs), erased those regions, and retrained the classifier on the remaining images. To accomplish more complete region mining, they use an online Prohibitive Segmentation Learning method with auxiliary segmentation supervision. Overall, they had to train three independent networks. Zhang *et al.* [2018] improved on the work of Wei *et al.* [2017] by recognising that training the three networks is too costly. Instead, they use only two parallel classifiers, trainable in an end-to-end manner, where the first classifier's mined region is removed so the second classifier identifies complementary object regions. Mai *et al.* [2020] further recognise that there is a shortcoming in previous Adversarial Erasing approaches: by only taking the CAMs from a particular layer, previous works are constraining their object region mining to a particular feature level, when there is exploitable information at the other feature levels as well. They remedy this by using modules that integrate region mining and erasing in a single component and applying these modules at multiple levels in the network.

A major failure case of previous works is that the aggregated CAMs do not conform well to the boundaries of the objects. Part of this imprecision stems from the Global Average Pooling layer in the classification networks [Yoon *et al.* 2022]. This layer averages all the features, including the irrelevant ones, thus diminishing the efficacy of localisation. Yoon *et al.* [2022] introduce a Gated Pyramid Pooling replacement, which is built on

Gated Convolutions (i.e. partial convolutions, but a learnt mask is used). They build a feature pyramid at multiple spatial resolutions, which allows coarse and fine details to be captured.

Finally, Yoon *et al.* [2022] introduce a Triplet Learning method to prevent the over-expansion of CAMs beyond an object's boundaries into other overlapping objects and to better differentiate foreground and background regions. Given an image (the anchor), a CAM is found and erased from the image, and the result then becomes the positive image. A negative image is simply one with no overlapping object classes. The negative image is generally considered as a background region with respect to the anchor and positive images. The intuition of attraction between the anchor and positive images is that it encourages the exploration of less-discriminative regions; repulsion between the anchor and negative images discourages over-expansion.

2.3.3 Memory Banks

The majority of weakly supervised semantic segmentation work has only ever considered a single input image at a time. There is valuable information across the entire training dataset which should be exploited when making decisions about the image currently being processed.

Zhou *et al.* [2022] recognise this and are able to take advantage of inter-image contextual information, by utilising a vast memory bank to store object patterns that occurred during training. They achieved state-of-the-art results which makes this idea significant, given that it has only been proposed recently.

Specifically, the memory bank stores embeddings for each class in the dataset. The work is composed of two major steps: Semantic Contrast and Semantic Aggregation. Similar to metric learning, Semantic Contrast forces the network to make embeddings of the same object category similar, but make those of different categories different, thereby maintaining unique and informative memory embeddings. The memory bank is large-scale, so Semantic Aggregation compresses the embeddings into a number of prototypes, which are aggregated to yield more holistic class representations. This final representation is used to determine the object localisations.

We find this idea of using context from multiple images to make decisions about content in a single current image appealing. Although we do not use the same memory-bank technique, the success of their work using multiple images motivates our approach using context from more than a single image in Chapter 4.

2.4 Saliency Maps

Saliency maps are visual representations used to highlight the most important or informative regions of an image as perceived by a model. In the context of computer vision, particularly in tasks such as image classification, object detection, and semantic segmentation, saliency maps help illustrate which parts of an input image contribute most strongly to a model's predictions. The concept is rooted in human visual perception,

where saliency refers to the distinctiveness or prominence of certain areas of a visual scene that attract attention.

In neural networks, saliency maps are typically generated by computing the gradient of the model's output with respect to the input image. This gradient-based approach produces a heatmap where higher intensity values indicate regions that have a stronger influence on the model's prediction. Popular methods for generating saliency maps include Grad-CAM [Selvaraju *et al.* 2017], Guided Backpropagation [Springenberg *et al.* 2014], and Integrated Gradients [Sundararajan *et al.* 2017].

For weakly supervised semantic segmentation, saliency maps can play a critical role in generating pseudo labels. Since only image-level labels are available during training, saliency maps provide an initial rough estimate of object locations by highlighting areas relevant to specific class predictions. These maps can then be refined through various techniques to produce dense pixel-level labels that are used for training segmentation models. As such, saliency maps form a crucial bridge between image-level supervision and pixel-level segmentation in weakly supervised learning.

2.5 Post-processing

Many prior works found that post-processing CAMs or Attention Maps from transformer using Random Walks (Section 2.5.1) and/or CRFs (Section 2.5.2) improved the quality of the segmentation maps [Ahn and Kwak 2018; Zhou *et al.* 2022; Yoon *et al.* 2022; Li *et al.* 2023; Xu *et al.* 2022].

2.5.1 Random Walk

A Random Walk is a probabilistic process in which an object takes a series of steps in various directions, with each step being determined probabilistically. In the context of Computer Vision, Random Walks form the basis of algorithms for image segmentation. The general idea involves starting with a set of labelled pixels, known as seeds, for which the class labels are known. In a weakly supervised setting, these seeds can be derived from an initial, incomplete segmentation map. An unlabelled pixel initiates a Random Walker, and the probability that this Walker reaches each of the seeds is calculated. The pixel is then assigned the label associated with the most likely seed; if multiple seeds share the same label, their probabilities are summed to determine the final assignment.

In practice, the image is represented as a graph where pixels correspond to nodes, and edges connect adjacent pixels. The weight of each edge is determined by the similarity or affinity between the connected pixels, which is a critical factor influencing the Random Walk process.

AffinityNet [Ahn and Kwak 2018] applied Random Walks by generating an initial, incomplete segmentation map from Class Activation Maps (CAMs) and training a network to predict pixel affinities — specifically, the similarity between adjacent pixels. These affinity predictions are then used as edge weights in the graph model for the Random

Walk. This approach has been adopted by multiple subsequent works [Li *et al.* 2023; Chen *et al.* 2023a] as a refinement technique to improve segmentation quality.

Other methods, such as RCA [Zhou *et al.* 2022] and MCTFormer+ [Xu *et al.* 2024], produce affinities as a by-product of their own broader segmentation processes. While these approaches do not employ traditional Random Walks, they still leverage pixel affinities to enhance segmentation maps through other refinement techniques.

MCTFormer+ [Xu *et al.* 2024], a transformer-based approach, directly utilises attention maps from the transformer encoder to identify object regions. Our work follows MCTFormer+'s innovative approach to computing affinities to refine these attention maps. This involves manipulating parts of the attention matrix to derive a relevance model that captures the relationship between each image location (i.e. patch) and every other location, which is discussed further in Section 4.3.4.

2.5.2 Conditional Random Field

A Conditional Random Field (CRF) [Krähenbühl and Koltun 2011; Liang-Chieh *et al.* 2015] is a discriminative statistical model commonly used to enhance the spatial consistency and accuracy of pixel-level segmentation predictions. It is particularly effective at modelling complex dependencies between image features and labels, such as the relationship between pixels that belong to the same object (e.g. in the predicted segmentation masks). In this context, labels for adjacent pixels are considered dependent, meaning that information from neighbouring pixels can help inform the label assigned to a given "central" pixel.

Following prior work [Zhang *et al.* 2021; Xu *et al.* 2021 2024], we employ a variant known as the Fully Connected CRF as a post-processing step to refine the final predicted segmentation masks. This approach aims to reduce noise in the segmentation predictions by promoting spatial smoothness and preserving object boundaries. Spatial smoothness encourages neighbouring pixels with similar appearance features (e.g. colour and texture) to have consistent labels. Additionally, CRFs help maintain sharp boundaries by effectively identifying and preserving sudden changes in pixel intensities, such as those occurring at object edges or transitions between foreground objects and the background.

Mathematically, a fully connected CRF [Krähenbühl and Koltun 2011] defines an energy function over the predicted labels, $\mathbf{Y}$, for all pixels:

$$E(\mathbf{Y}) = \sum_i \psi_u(y_i) + \sum_{i<j} \psi_p(y_i, y_j), \tag{2.6}$$

where $i, j \in \{1, 2, \cdots, N\}$ and N is the number of pixels. $\psi_u(y_i)$ is the Unary Potential, which measures the likelihood of pixel i having label y_i — this is usually taken as the output from the segmentation model. $\psi_p(y_i, y_j)$ is the Pairwise Potential, which encourages pixels i and j to be similar if they are similar in appearance (e.g. colour, texture, proximity, etc.).

Following Liang-Chieh *et al.* [2015], the unary potential is

$$\psi_u(y_i) = -\log P(y_i), \tag{2.7}$$

where $P(y_i)$ is the probability of label y_i being assigned to pixel i. In most cases, $P(y_i)$ is the output taken directly from the segmentation model, and has often been activated by a softmax function.

Following Krähenbühl and Koltun [2011] and [Liang-Chieh *et al.* 2015], the pairwise potential is

$$\psi_p(y_i, y_j) = \mu(y_i, y_j) \sum_{m=1}^{K} w^{(m)} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j), \tag{2.8}$$

where $\mu(y_i, y_j) = 1$ if $y_i \neq y_j$, and 0 otherwise. Each $k^{(m)}$ is a Gaussian kernel, depending on image features ($\mathbf{f}$) extracted for pixels i and j, weighted by the linear combination weights, $w^{(m)}$. Specifically, the kernels incorporate position and colour terms:

$$w_1 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}\right) + w_2 \exp\left(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2}\right), \tag{2.9}$$

where the first term depends on pixel position vectors, p, and colour intensity vectors, I, and the second term only depends on the pixel positions — the first term encourages nearby pixels with similar colour to have the same class label, and the second term removes small, isolated regions (i.e. erroneous activations). σ_α, σ_β, and σ_γ are hyperparameters which control the "scale" of the Gaussian kernels.

The pairwise potentials show that even though "potential" is defined for all pairs of pixels in an image, if the pixels are far away from each other, then they exert minimal influence on each other.

2.6 Concepts Discussed in Later Chapters

While this chapter provides a general overview of WSSS methods, several specific concepts and techniques that are central to our work will be discussed in detail in later chapters. These include:

1. **Co-occurrence Concepts** (Chapter 3): The phenomenon of class co-occurrence, where certain classes frequently appear together in images, will be thoroughly examined. Related work addressing this issue will be discussed in Section 3.3.

2. **Cross Attention Concepts** (Chapter 4): We will present an in-depth exploration of Vision Transformers and the self-attention mechanism, including how these mechanisms can be extended to cross attention. Related work involving cross attention will be reviewed, and we will detail the process of extracting attention maps from transformers and refining them using the novel approach introduced by MCTFormer+ [Xu *et al.* 2024], which replaces the need for traditional affinity-based methods.

3. **Contrastive Learning Concepts** (Chapter 6): The application of contrastive learning within our framework will be discussed, including the use of cross entropy loss as a foundational component.

These concepts are central to our approach and will be revisited throughout the dissertation as we introduce our proposed methodologies and their underlying motivations.

2.7 Conclusion

Overall, many approaches struggled with differentiating between the extremities of overlapping objects (i.e. where one object occludes the other) and robustly identifying intricate structures, such as the legs of chairs. These failure cases remained a key factor contributing to the gap between state-of-the-art performance and near-perfect segmentation.

Recent CNN-based approaches achieved state-of-the-art results by employing two key principles:

- **Metric Learning:** Utilising negative samples to more effectively distinguish between foreground and background regions.
- **Cross-image Context:** Incorporating information from the entire dataset when analysing a given image, rather than relying solely on individual images.

It follows that any further research, including ours, would benefit from integrating these principles.

However, achieving state-of-the-art performance with CNN-based frameworks often required increasingly complex architectures, involving intricate modules designed to extract additional information from the data. In contrast, more recent transformer-based frameworks achieved competitive, and even state-of-the-art, results using simpler processes, as detailed in Chapter 3 for methods related to co-occurrence, Chapter 4 for those involving cross attention, and Chapter 6 for approaches focused on metric learning. This advantage was attributed to the transformers' self-attention mechanism, which inherently facilitated the modelling of global context within an image. In comparison, CNN-based frameworks required exceptionally deep networks to achieve similar global coverage due to the limited receptive fields associated with convolutions.

Consequently, we opted to use transformers as our base architecture, particularly focusing on how they could be leveraged to model cross-image context and related principles, as discussed in the following chapters.

Chapter 3

Class Co-Occurrence

This chapter investigates the existence of class co-occurrence and show that it is a contributing factor to deteriorated performance for Weakly Supervised Semantic Segmentation (WSSS).

Co-occurrence is the phenomenon where an object category or class commonly occurs together with some other aspect of an image such as the background. For example, cars (object) generally occur on roads (background), and aeroplanes (object) are generally photographed in the sky (background). When it comes to segmenting these objects from the image, models can struggle to learn where the object boundaries are relative to their co-occurring background, since strict adherence to these boundaries is not required nor enforced during the classification training phase. Consequently, there are misclassified pixels in these regions, degrading the corresponding classes' IoU scores.

The co-occurrence problem in general has been a more recent area of study [Lee *et al.* 2022; Yang *et al.* 2024], with many earlier works focusing on solving issues related to object coverage and boundary adherence [Ahn and Kwak 2018; Yoon *et al.* 2022; Xu *et al.* 2022; Chen *et al.* 2023a; Xu *et al.* 2024], and oversmoothing [He *et al.* 2023; Ru *et al.* 2023].

We consider a variation of this co-occurrence, where the co-occurrence occurs between classes that exist in the dataset, which has not been directly addressed before (Yang *et al.* [2024] briefly mention it, but not as a direct focus of their work), and so is of particular interest to us and forms a core contribution of this work.

In Section 3.1 we formally define our notion of class co-occurrence, and in Section 3.2, we introduce and discuss the two main datasets for weakly supervised semantic segmentation, PASCAL VOC 2012 and MS COCO 2014. We discuss and compare other co-occurrence-related works and their solutions in Section 3.3. In Section 3.4, we quantify the degree of co-occurrence in our datasets, which is something that prior works fail to do, and provide empirical evidence that increased levels of class co-occurrence leads to deteriorated performance for the class pairs involved. To our knowledge, this is the first work to consider these aspected within the WSSS paradigm.

3.1 Definition

Two classes which both occur in a given image are said to *co-occur* in that image. Consequently, when we scale this definition to the whole dataset, we need to measure the frequency with which two classes occur together in images. For example, in a typical natural scene featuring a computer, it is common to find both a *mouse* and a *keyboard*, as they are standard interface devices for most users. A significant proportion of images containing a mouse are likely to depict this natural setting — alongside a computer and keyboard. By analysing such images, we can quantify the extent to which the *mouse* and *keyboard* objects appear together, as well as how often they occur independently of one another.

Let x_i be an image taken from a dataset S and let y_i be the set of class labels for image x_i. Given a class pair (A, B) which both exist in dataset S, we formally define the co-occurrence between classes A and B as follows:

$$\text{Cooccurrence}(A, B) = \frac{\sum_{x_i \in S} \{\mathbb{1}\}_{A \in y_i \wedge B \in y_i}}{\sum_{x_i \in S} \{\mathbb{1}\}_{A \in y_i \vee B \in y_i}}, \tag{3.1}$$

where $\{\mathbb{1}\}_{A \in y_i \wedge B \in y_i}$ is an indicator function taking on 1 when *both* A and B are in image x_i and 0 otherwise, and $\{\mathbb{1}\}_{A \in y_i \vee B \in y_i}$ is an indicator function taking on 1 when *either* A and B are in image x_i and 0 otherwise.

It is worth noting that $\text{Cooccurrence}(A, B) = \text{Cooccurrence}(B, A)$ due to the commutative properties of the conjunctive and disjunctive operators.

$\text{Cooccurrence}(A, B) \in [0, 1]$, since it is a ratio of images containing both classes to images containing only one of them — that is, the numerator is always smaller than or equal to the denominator. In the extreme cases, when the co-occurrence is 0, this means that the two classes never appear together — for example, you would not expect to see *mouse* alongside *shark*. When the co-occurrence is 1, this means that the two classes always appear together — for example, if all instances of *mouse* are accompanied by *keyboard*. Anything between these two extremes quantifies the degree to which the two classes appear together. For example, if we have 5 and 7 images of *mouse* and *keyboard* appearing without the other, respectively, and 3 images with both of them, then $\text{Cooccurrence}(\text{mouse}, \text{keyboard}) = \frac{3}{5+7} = 0.25$. This means that 25% of a time, *mouse* will occur with a *keyboard* and vice versa, and 75% of the time, one will appear without the other.

This definition of class co-occurrence, framed in terms of an Intersection over Union (IoU), is particularly attractive because it provides a natural and intuitive way to quantify the extent of overlap between two classes across a dataset. By measuring the proportion of images in which both classes are present relative to the total number of images containing at least one of the classes, this metric effectively captures how frequently the classes tend to appear together. The use of IoU in this context also aligns well with its widespread adoption as a performance metric in semantic segmentation, offering a consistent and familiar framework for analysis. Moreover, this approach inherently emphasises the co-occurrence of classes in a way that is robust to the total

number of images in the dataset, allowing for meaningful comparisons between different datasets or subsets thereof.

3.2 Data

There are two main benchmark datasets used for the task of Weakly Supervised Semantic Segmentation in the literature: PASCAL VOC 2012 [Everingham *et al.* 2012] and MS COCO 2014 [Lin *et al.* 2014].

3.2.1 PASCAL VOC 2012 Dataset

PASCAL VOC 2012 (VOC) dataset has 20 object classes and one background class, where object classes have been chosen such that they represent realistic scenes. Notably, each image may consist of multiple object classes.

There are 1464 training images, 1449 validation images, and 1456 testing images. Following prior work [Chang *et al.* 2020; Xu *et al.* 2021], the training set will be augmented to 10582 images using the Semantic Boundaries Dataset (SBD) [Hariharan *et al.* 2011]. SBD is an extension of PASCAL VOC 2011, with the same class labels as PSACAL VOC 2012, which originally had pixel-level annotations for the boundaries of objects. In order to integrate SBD into semantic segmentation datasets, the object regions defined by the boundaries are "filled in".

The official PASCAL VOC 2012 evaluation server includes an additional held-out test set that is not publicly available. As a result, most papers in the field report their results on the official validation set, effectively using it as a test set. To compensate, researchers typically create their own validation sets by splitting the available training set. In this work, we follow the same convention: while our reported results are based on the VOC validation set, we emphasise that it was used as a test set. To enable model validation during development, we split the original training set into separate *train* and *validation* sets, consisting of 8465 and 2117 images, respectively (i.e. an 80/20 split).

3.2.2 MS COCO 2014 Dataset

MS COCO 2014 (COCO) dataset has 80 object classes and one background class where each image may consist of multiple object classes. There are 82783 training images, 40504 validation images, and 40775 testing images.

The core contribution of this dataset is that when object classes appear in the same image, it is generally because they are semantically linked (hence the name *COCO*, which stands for *Common Objects in Context*). This means that semantically segmenting this dataset is substantially harder than PASCAL VOC because more intricacies need to be learnt to differentiate between the linked objects and their boundaries. For example, in this dataset, *mouse* and *keyboard* are considered "in context", which means that they are naturally likely to highly co-occur.

Similarly to VOC, the test set is not publicly available, necessitating the use of the official validation set as the test set. When necessary, we re-split the training set into new training and validation sets, with 66226 and 16557 images, respectively (i.e. an 80/20 split).

3.2.2.1 Constructing Ground-Truth Pixel Masks

Whilst the VOC dataset provides pixel-level masks for its images, COCO does not. Instead, they provide json-encoded annotations for the images, from which we need to construct the pixel-level masks. The annotations include metadata about each image, as well as a description of the contours of the bounding areas for each object in each image.

The issue is that some of the annotations for an image have overlapping regions, and there is no extra information provided in the annotations about the depth of each object. This means we cannot infer from the annotations themselves which object is meant to be on top in the different overlapping regions. Figure 3.1, rows (b) and (d), show the annotations and ambiguous regions for example images, respectively. For these examples, the pixels for the table and the dishes/cutlery on the table, all overlap. If we only use the provided annotations from COCO, then the objects could be assigned any of the labels present in the overlapping region — row (e) shows how by using the naive method of choosing the highest class index to be present, it results in the table always being considered on top, occluding the other objects incorrectly.

There is another dataset, COCO-Stuff [Caesar *et al.* 2018], which is a superset of MS COCO 2014 in that it adds extra classes to the dataset. MS COCO 2014 has 80 "thing" classes (which are objects with a well-defined shape, like people and furniture). COCO-Stuff builds on these classes by introducing another 91 "stuff" classes (which are general regions of the image, mostly including background regions like grass or sky). However, the attractive part of COCO-Stuff is that they include pixel-level masks, where both "thing" and "stuff" objects are labelled (example masks are shown in Figure 3.1, row (c)). These masks were obtained by first partitioning each image into superpixels and then asking human annotators to assign each superpixel to a class, with the basis that the superpixels respect object boundaries and only represent a single class. We will use these masks as a guide for resolving the ambiguities introduced by the MS COCO 2014 annotations, since they provide a stable (and published) baseline for the ground-truth masks. This is something that most other works overlook: although they do not mention how they obtain the pixel-level masks, it is likely that they simply resolve ambiguities by taking the argmax of the class indices available for a pixel (i.e. they choose the class with the largest index), which may lead to the wrong object being placed on top, thus incorrectly obscuring other objects. This naive method is shown in Figure 3.1, row (e).

Specifically, we follow these steps to construct the ground-truth pixel-level masks:

1. Use the provided annotations (from MS COCO 2014) to construct a binary map of all foreground objects. This is not affected by the annotations' ambiguity.

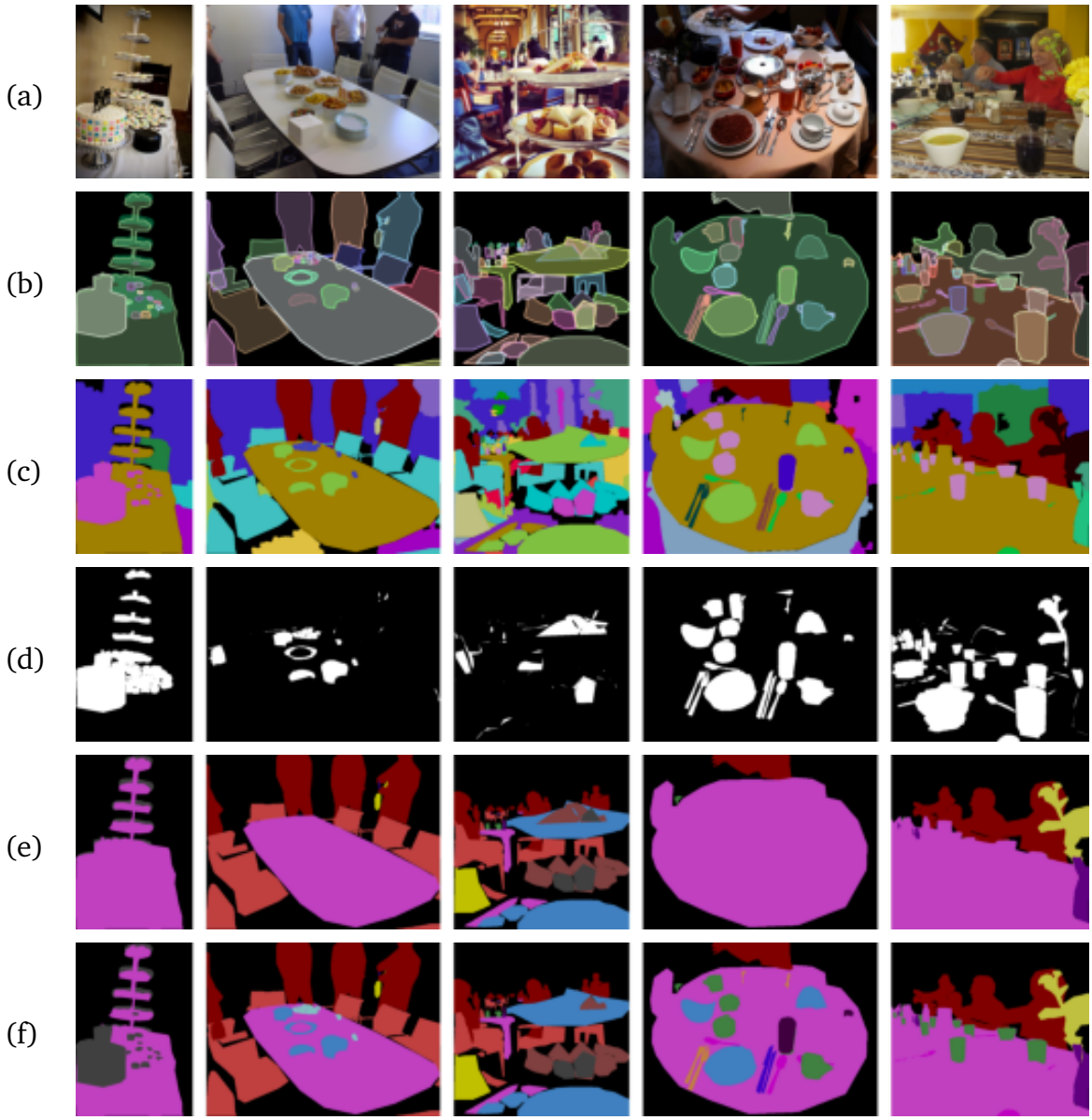


Figure 3.1: A few examples of our constructed Ground Truth Pixel-Level Masks for COCO, compared to the generally-accepted practice. (a) Original RGB images. (b) COCO dataset annotations. (c) COCO-Stuff Pixel-Level Masks. (d) Regions where COCO annotations overlap. (e) Naive masks, where any annotation ambiguity is resolved by taking the class with the higher index. (f) Our ground-truth pixel-level masks.

2. Use these foreground maps to mask out the background and "stuff" regions from the corresponding COCO-Stuff mask, so that only the required foreground objects remain. Since we are using the 2014 version of the dataset, in line with other works, we do not use the extra classes introduced in the more recent version of the dataset.

3. The masked COCO-Stuff mask will form the basis of our ground truth masks. The caveat is that the COCO-Stuff masks do not guarantee coverage or inclusion of the foreground objects identified by the annotations (although the vast majority are): there may be some objects included in the annotations that are not present in the COCO-Stuff mask.

4. In these rare cases where COCO-Stuff does not account for a foreground region, we resort to the same, naive method that other works generally use: take the class with the largest index. Sometimes this is trivial since the absent foreground region only has one option from the annotations, but if there is ambiguity, then we use the argmax method to resolve it. While not perfect, it does align with other works, and is the most reasonable option without resorting to more complex methods (which is something future work can address).

Our construction of the pixel masks is shown in Figure 3.1, row (f) — through the use of the COCO-Stuff masks there is minimal, if any, ambiguity that is not addressed. Therefore, our method resolves much of the ambiguity from the annotations, leading to more accurate ground-truth masks, which means our metrics and evaluation is more representative of our models' performance. We also expect that other works will experience an increase in IoU performance when comparing their segmentation masks to our more accurate ground-truth masks, although this is left to Future Work. However, we do reimplement MCTFormer+ [Xu *et al.* 2024], which we build upon in later chapters, in Section 4.3.5 and use the reimplemented results for any comparisons made (on both VOC and COCO).

3.3 Related Work

Awareness around co-occurrence-related problems in WSSS has only recently gathered pace. Zhang *et al.* [2015] was one of the earliest works to mention co-occurrence, and built a statistical model (a joint CRF) using image-level label co-occurrence and other visual features. However, they did not see the co-occurrence as an issue, and instead used it so that the existence of some image aspect would increase the probability of a class being present. Whilst obtaining the state-of-the-art at the time, this idea that co-occurrence is useful proved to be contrary in later works, where co-occurrence was viewed as a limitation and attempts were made to break it [Lee *et al.* 2021b; Su *et al.* 2021; Lee *et al.* 2022; Yang *et al.* 2024].

Lee *et al.* [2021b] recognised that co-occurring regions of images are frequently misclassified as being part of the accompanying class, leading to degraded performance (hence the title: "Railroad is not a Train", with *train* being the foreground object and the *railroad* being part of the background). Their core contribution is showing that re-

ducing this co-occurrence leads to improved results, which they achieved by using extra supervision from saliency maps generated using an off-the-shelf model. The saliency maps were not as susceptible to the co-occurrence issue.

Another framework, Cross Language Image Matching (CLIMS) [Xie *et al.* 2022], use natural language (instead of the saliency maps like Lee *et al.* [2021b]) to help disambiguate co-occurring image regions from the object of interest, by suppressing the background regions of objects. This framework was largely powered by the CLIP model [Radford *et al.* 2021], which showed that models pre-trained using the natural language task of matching captions with images were able to achieve high zero-shot performance on downstream tasks when additional natural language is used to reference various visual concepts, which may or may not have been previously learnt.

However, the use of the saliency maps by Lee *et al.* [2021b] and natural language by Xie *et al.* [2022] introduce another source of data and supervision to the training process. We differentiate ourselves by only utilising the data at hand, which also confirms that the additional saliency maps and natural language are not artificially reducing the degree to which the original co-occurrence problem occurs in our datasets, which they may do if their original datasets have low co-occurrence or otherwise unbiased data.

One of the best-performing and most recent models, Separate and Conquer (SeCo) [Yang *et al.* 2024] is the most generally-applicable model in that it can be applied to class-background co-occurrence and class-class co-occurrence. They divide each image into a number of patches (as is usual when using Vision Transformers), and assign a category tag to each one. The category tag is an estimate of the most prominent class present in that patch, and is based on CAMs generated from an auxiliary classification head they add to their model. They then perform contrastive learning between the different patch embeddings as well as between the patch embeddings and global class embeddings (of which there is one per class in the dataset). However, the authors acknowledge that the assignment of category tags using an initial CAM is a noisy process and attempt to mitigate this by using a confidence-threshold system. It is this fact that is unattractive to us: the contrastive learning is performed using the imperfect category tags as labels, which will compound the error and creates a non-stationary problem.

There are also two prominent data-driven methods, Context Decoupling Augmentation (CDA) [Su *et al.* 2021] and W-OoD [Lee *et al.* 2022], which aim to alleviate the co-occurrence issue by augmenting the dataset instead of making technical alterations to models. Su *et al.* [2021], and Zhang *et al.* [2015], recognise that the cause of co-occurrence is that some foreground objects occur quite frequently with the same background. Whilst the latter used this idea to build their statistical model on the basis that certain background features being present implies that the associated foreground object is likely to be present, the former follow the more recent train of thought and aim to directly break the dependency between the foreground object and the associated background by decoupling their context. That is, they argue that improving the WSSS performance is not simply related to increasing the semantic variation with which a particular class appears in the dataset (e.g. traditional augmentation techniques like random cropping, transformations, and colour jittering), but rather that the performance for certain classes might be directly inhibited by the frequent, similar

context in which those objects generally occur. Specifically, Su *et al.* [2021] use an initial off-the-shelf WSSS method to get initial segmentations for the objects of interest, and then randomly place these objects in other images, thereby giving them a different background context. However, this is inherently limited by the effectiveness of the off-the-shelf model, and any errors that exist there will be propagated downstream.

Likewise, Lee *et al.* [2022] (W-OoD) recognise the cause of co-occurrence, and suggest breaking the contextual dependence by introducing extra background-only images to the dataset. This out-of-distribution data is included in a targeted manner by choosing images that contain the co-occurring background features, but not the co-occurring foreground object. The model can use the background-only images to distinguish between the background and the object of interest, and is further encouraged to do so by using metric learning to ensure the embeddings for the foreground object are far away from the embedding for the corresponding background-only image.

MCTFormer+ [Xu *et al.* 2024] is currently one of the best performing models for WSSS, although it does not mention nor address the co-occurrence problem. CDA [Su *et al.* 2021], W-OoD [Lee *et al.* 2022], and CLIMS [Xie *et al.* 2022] all achieve competitive overall performance, but only SeCo [Yang *et al.* 2024] achieves performance on par with MCTFormer+. Hence, we are justified in using it as the base for our solution in Chapter 4 and we discuss it further in Section 4.3.

Finally, although all of these works (with the exception of Zhang *et al.* [2015] and Xu *et al.* [2024]) acknowledge that co-occurrence is an issue leading to greatly degraded performance and provide methods to address it, none of them quantify the degree to which co-occurrence occurs in the dataset. Furthermore, none of these works explicitly demonstrate that co-occurrence is a problem: they assume it is a problem and take the resulting performance improvements of their methods as confirmation of this assumption. We will go one step further and investigate the degree of class co-occurrence in the VOC and COCO datasets, and empirically demonstrate that increased class co-occurrence leads to degraded segmentation performance for those classes.

All of these works address the co-occurrence between a class and some background feature, except SeCo [Yang *et al.* 2024], which can also be applied to class-class co-occurrence. In contrast, our definition of co-occurrence leads us to focus on co-occurrence between explicit classes of the datasets. Notably, it is not possible to measure the co-occurrence between a class and certain background features in most datasets — the datasets only include class labels, and not labels for different background features, meaning the datasets do not support this kind of co-occurrence analysis. The class-background co-occurrence can only be described qualitatively, by observation.

3.4 Quantification

This section quantifies the level of class co-occurrence in VOC and COCO, and provides empirical evidence that increased co-occurrence for a pair of classes is detrimental to the weakly supervised IoU performance for those classes.

dataset	class1	class2	images with at least one of these classes	images with both of these classes	co-occurrence (portion of occurrence)
VOC	chair	diningtable	1427	414	0.2901
	bottle	diningtable	1137	190	0.1671
	chair	pottedplant	1556	194	0.1247
	chair	tvmonitor	1599	196	0.1226
	chair	sofa	1679	198	0.1179
	bus	car	1386	157	0.1133
	car	person	4781	524	0.1096
	chair	person	4854	529	0.1090
	bottle	chair	1767	175	0.0990
	bottle	person	4436	433	0.0976
COCO	baseball bat	baseball glove	2505	1183	0.4723
	mouse	keyboard	1883	878	0.4663
	sports ball	tennis racket	4093	1261	0.3081
	fork	knife	4344	1290	0.2970
	cup	dining table	11555	3341	0.2891
	toilet	sink	4425	1183	0.2673
	chair	dining table	13685	3643	0.2662
	microwave	oven	2453	639	0.2605
	car	truck	10339	2588	0.2503
	apple	orange	1911	476	0.2491

Table 3.1: A table showing the top 10 pairs of co-occurring classes in descending order. The two columns (the second and third) denote the pair of classes (order does not matter). The next two columns give the absolute number of images in the datasets with at least one of the pair of classes, or both of them, respectively. The final column provides the co-occurrence level in the range $[0, 1]$. Note that the maximum co-occurrence for VOC is 29.01% and for COCO is 47.23%.

3.4.1 Dataset Class Co-Occurrence Levels

Section 3.3 discussed prior work and found that whilst they consider co-occurrence a problem and propose solutions to it, they fail to investigate the degree to which co-occurrence exists in various datasets.

We initially find the co-occurrence between all possible pairs of classes in the VOC and COCO datasets and show the 10 highest pairs in Table 3.1.[1] We immediately notice that in the worst cases, the class co-occurrence is substantially worse for COCO than it is for VOC, with the highest COCO co-occurrence (47.23%) being nearly double that of VOC (29.01%). There is a decrease of $\sim$16% co-occurrence in VOC from the first to second case, and from the second to the third case in COCO. However, even after this drop-off, the COCO co-occurrence is still roughly double that of VOC.

To further visualise the skewness of the datasets in terms of class co-occurrence, we plot Figure 3.2. The figure shows that COCO has many more large outliers than VOC does, with a substantial number of the COCO outliers being in the same region as the two

[1]We release the complete list of class pairs and their co-occurrence as part of our code repository. See Section 4.4.

worst VOC outliers — although both datasets are skewed to lower class co-occurrence, COCO's large outliers are much worse than VOC's.

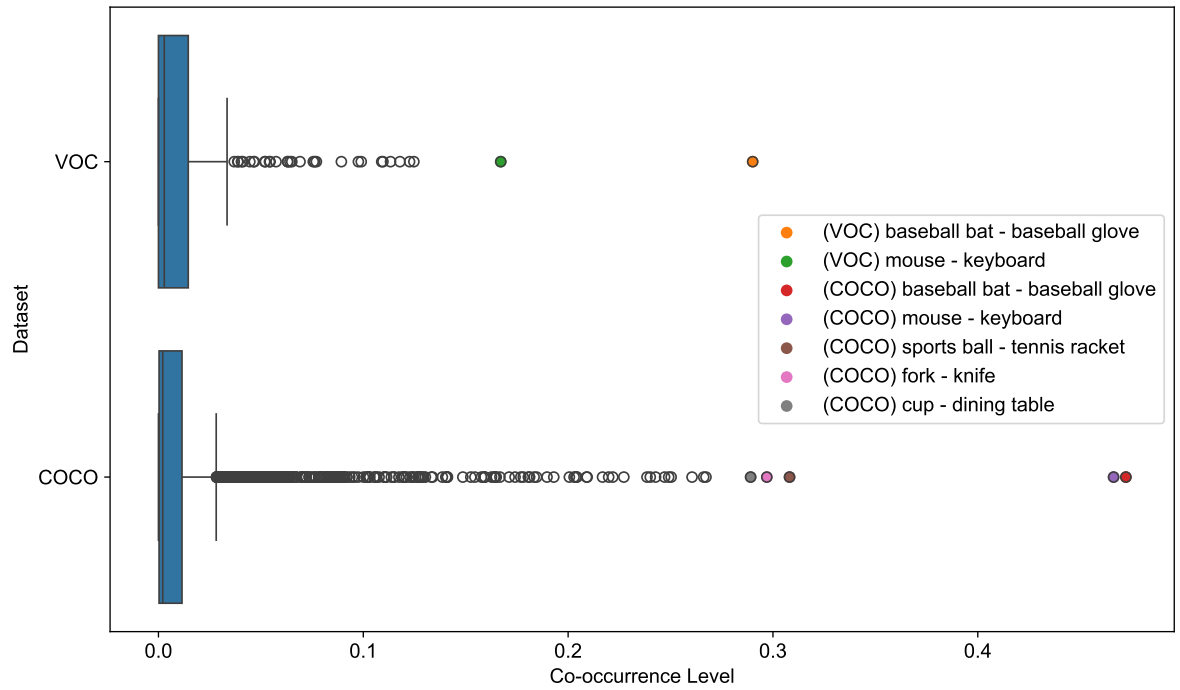


Figure 3.2: A Box-and-Whisker Diagram showing the distribution of class co-occurrence in VOC and COCO. The whiskers are drawn to $1.5 * IQR$ beyond the lower and upper quartiles, respectively. The circles denote outliers (co-occurrence values which lie beyond the whiskers) and the worst cases from VOC and COCO have been labelled. Both datasets are skewed to low co-occurrence values, hence the lower whisker, lower quartile, and median more-or-less overlap with each other.

In order to view the overall degree of co-occurrence in each dataset, we need to be able to compare them in a scale-independent way, considering that COCO is much larger than VOC. To this end, we first count the number of pair instances that have a co-occurrence in a given range, and convert the count to a portion of the respective dataset. We then compare the relative portion of each dataset in each range in Figure 3.3, up to a co-occurrence rate of 50% since no class pair exceeds that co-occurrence level. Evidently, the majority of the co-occurrence sub-ranges have a greater portion of the COCO dataset than the VOC dataset, with many of the higher sub-ranges favouring COCO.

Based on this analysis of the worst-case scenarios and the broad overview of the datasets, COCO contains class co-occurrence to a far greater degree than VOC.

3.4.2 Performance with Class Co-occurrence

Following Section 3.4.1 showing that COCO has more co-occurrence than VOC, we want to investigate the effect of this increased co-occurrence on an existing model for

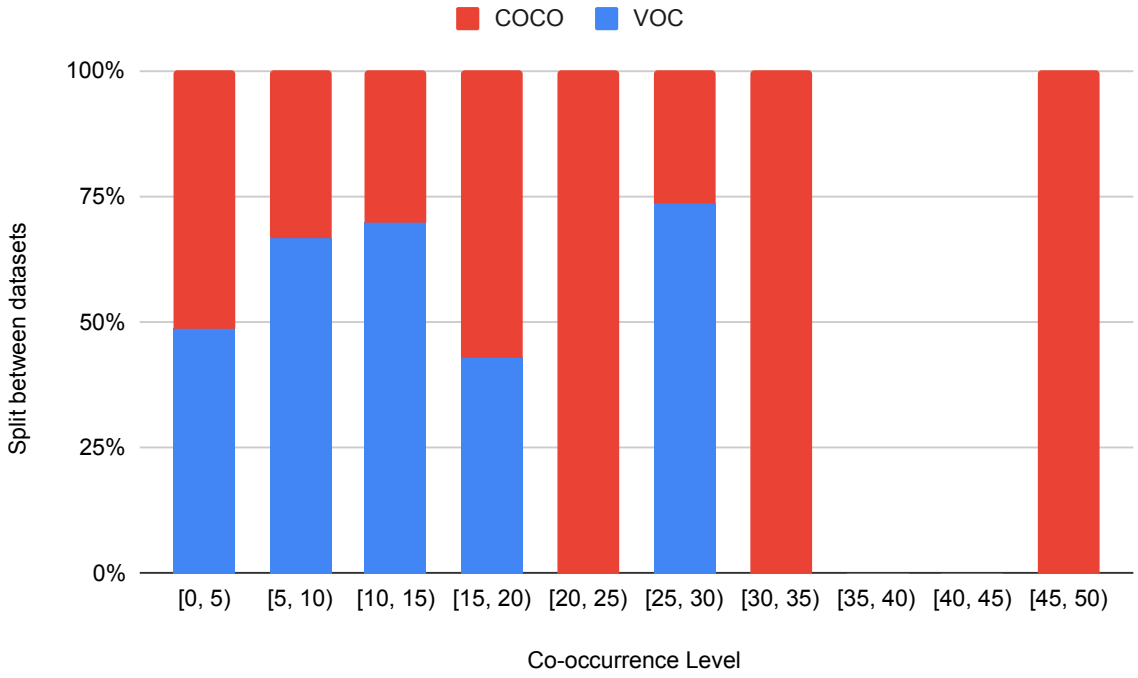


Figure 3.3: Graph showing the relative portion of each dataset that has pairs of classes with a co-occurrence level in various ranges. For each co-occurrence range, we determine the portion of class pairs in each dataset that fall in it (the scaling required to get the proportional values allow comparison of the different sized datasets). For a specific range, we then plot the relative size of each datasets' class pairs that fall in it. For example, of all class pairs with a co-occurrence between 0% and 5%, more than half of them are from COCO, hence the red bar extends just past 50% for this range. No pair of classes in either dataset have co-occurrence in the range $[35, 45)$, hence those columns are blank.

weakly supervised semantic segmentation. We focus on MCTFormer+ [Xu *et al.* 2024], since that is one of the current state-of-the-art models, and we used it as a base for our own contribution in Chapter 4.

We artificially control the level of co-occurrence for a chosen pair of classes. To do this in a fair manner, we created a series of image lists that are subsets of the original datasets and control the inclusion of images which possess at least one of the chosen pair of classes. To manipulate the co-occurrence level, we adjust the ratio between images containing only one of the classes and images containing both classes. Importantly, we ensure that the proportion of images containing only the first class and those containing only the second class remains consistent across different co-occurrence levels. Additionally, we keep the total number of images containing at least one of the classes constant, so that any observed changes in performance are exclusively due to varying co-occurrence levels, rather than differences in the amount of training data.

We selected four pairs of classes from COCO, with high, moderate, and low levels of co-occurrence, and for which there is sufficient data to sample according to this scheme:

1. Baseball Bat and Baseball Glove, which have an original co-occurrence of 47.23%
2. Mouse and Keyboard, which have an original co-occurrence of 46.63%
3. Cup and Chair, which have an original co-occurrence of 15.88%
4. Bicycle and Motorcycle, which have an original co-occurrence of 7.21%.

These results are presented in Figure 3.4. The first two pairs have a high co-occurrence in the original dataset: by reducing the co-occurrence, MCTFormer+ improves on the class IoU scores; and by further increasing the co-occurrence, MCTFormer+ experiences a further deterioration on the class IoU performance. We also examine the last two pairs to ensure that this behaviour is consistent for classes that originally have low co-occurrence, which it is. Note that the class-specific IoU performance does experience small changes in performance over some co-occurrence levels, but every class does experience a large drop off at one of the increases in co-occurrence. However, the decrease in performance for each pair of classes is not evenly split between the two classes: one may suffer a larger performance decrease than the other over the same increase in co-occurrence. This suggests that the model tends to bias towards one of the classes in a co-occurring pair, at the cost of the other.

We also want to confirm that this trend holds for VOC. We select the following class pairs:

1. Chair and Dining Table, which have an original co-occurrence of 29.01%, which is the highest in the dataset.
2. Car and Person, which have an original co-occurrence of 10.96%, which is relatively moderate co-occurrence for VOC.

Similarly to COCO, Figure 3.5 shows that as the co-occurrence between pairs of classes increases, MCTFormer+'s performance on those classes decreases. This change does not necessarily occur in a consistent manner: for small values of co-occurrence (e.g.

5% or 10%), it is possible that performance increases as co-occurrence increases, likely due to the co-occurrence not being an inhibiting factor yet, or even other factors in the dataset contributing to the model's ability to learn. Over larger increases in co-occurrence, there is always a decrease in performance. Notably, the "dining table" class shows a large spike in performance from 0% to 10% before decreasing again. It is unclear exactly why this happens, but we conjecture it might be due to the "dining table" images without "chair" (i.e. at 0% co-occurrence) not providing enough information to help disambiguate it from other classes, where the images including "chair" do provide such information.

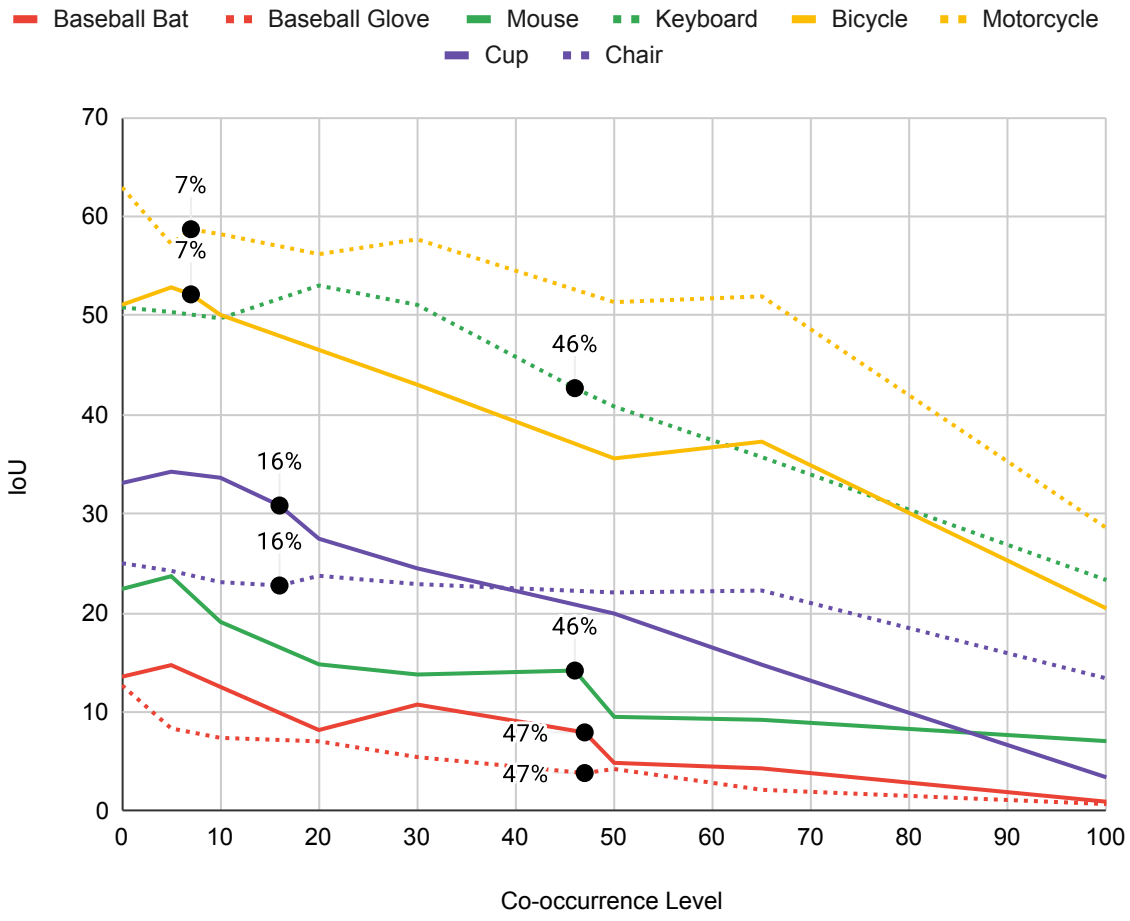


Figure 3.4: Graph showing the IoU performance for models trained on various levels of co-occurrence for certain class pairs from MS COCO 2014. The colours denote the pairs of classes considered, with one class having a solid line and the other a dashed line (which class is considered "first" has no significance). The black points show the co-occurrence level for the pairs of classes in the original dataset: the baseball bat and baseball glove, and the mouse and keyboard pairs have high co-occurrence in COCO, but the bicycle and motorcycle, and cup and chair pairs have low co-occurrence in COCO.

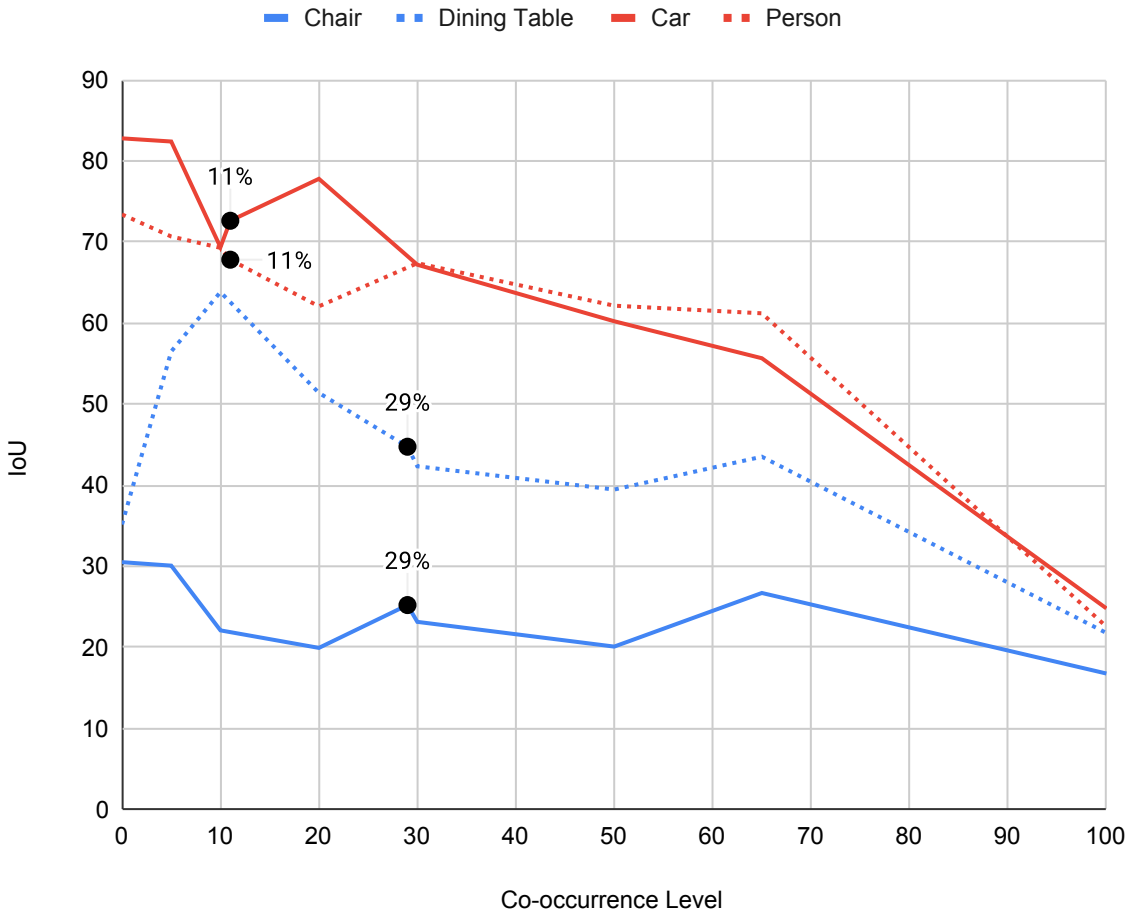


Figure 3.5: Graph showing the IoU performance for models trained on various levels of co-occurrence for certain class pairs from PASCAL VOC 2012. The colours denote the pairs of classes considered, with one class having a solid line and the other a dashed line (which class is considered "first" has no significance). The black points show the co-occurrence level for the pairs of classes in the original dataset: the chair and dining table have high co-occurrence in VOC, but the car and person have relatively moderate co-occurrence.

The finding that increased levels of co-occurrence correlate with deteriorated performance suggests a potentially widespread issue that has not yet been thoroughly investigated. Our results indicate that co-occurrence is a primary contributing factor, likely because the model struggles to disambiguate classes that frequently appear together. In such cases, the model expends more capacity on telling these classes apart, rather than on learning finer-grained details such as intra-class variance or object boundaries. This leads to degraded class IoU performance compared to classes with low co-occurrence, where the model can allocate more capacity to these tasks. While co-occurrence appears to be the dominant driver of this effect, we note that other dataset-specific factors (such as the relative difficulty of certain instances) may also play a role.

3.5 Summary

This chapter investigated the existence of the class co-occurrence problem, which is, in brief, the phenomenon where two classes frequently occur together in images, and proved that this degrades the performance for the corresponding classes for weakly supervised semantic segmentation. This degradation in performance is not equal for the classes: one may experience worsened performance compared to the other one. It is likely that the model spends more effort simply telling the classes apart and less effort on learning the finer aspects about the related objects, like learning where the respective object boundaries are, especially since this is not necessary for the classification task, which is necessitated by only image-level labels being available for the weakly supervised task of semantic segmentation.

This chapter also quantified the degree to which class co-occurrence exists in the two datasets, PASCAL VOC 2012 and MS COCO 2014, and found that class co-occurrence occurs to a far greater degree in COCO, and the class pairs on the higher end of co-occurrence are worse for COCO than for VOC.

In the following few chapters, we address the class co-occurrence problem and recover performance for the affected classes, and will focus predominantly on COCO. Since the decrease in performance is not split equally between the two classes of a co-occurring pair, we do not expect a balanced recovery that is equal between the classes: the larger the decrease in performance, the more pronounced an effect we expect to have.

Chapter 4

Final Framework

Chapter 3 investigated the prevalence of class co-occurrence within the PASCAL VOC 2012 and MS COCO 2014 datasets, quantifying its presence and demonstrating that increased co-occurrence leads to degraded performance in weakly supervised semantic segmentation. Chapter 2 reviewed various CNN-based approaches, while Section 3.3 examined transformer-based methods related to co-occurrence modelling. While those methods focused on leveraging co-occurrence within individual images, this chapter explores the application of cross image attention as a mechanism to incorporate cross-image context. An ablation study of our cross image attention mechanism is presented in Chapter 5, highlighting the importance of design choices in our proposed framework.

The need for cross-image context arises from the observation that most existing methods rely solely on information from the current image to make predictions [Ru *et al.* 2023; Yang *et al.* 2024; Xu *et al.* 2024], neglecting the wealth of information available across the entire dataset. In contrast, cross image attention allows us to model relevance between different, yet related images, providing a mechanism for transferring knowledge between them. This is particularly appealing for weakly supervised semantic segmentation, where limited supervision can be supplemented by leveraging information from other images containing related content.

Transformer-based approaches are especially suitable for this task due to their simpler architectures compared to CNN-based methods, which often require increasingly complex modules to achieve competitive performance (Section 2.3). The self-attention mechanism at the core of transformers is highly effective at modelling relevance by comparing queries and keys to produce attention maps [Xu *et al.* 2022 2024; Yang *et al.* 2024]. These maps are activated only for correlated regions, meaning that object class tokens will only generate strong attention values when the corresponding classes are present, while the remaining class tokens will produce broad, low-value attention maps.

Cross attention, an extension of the self-attention mechanism, spreads attention across different input sequences rather than limiting it to tokens within a single input. This chapter examines how this mechanism is harnessed to enhance weakly supervised semantic segmentation by incorporating information from multiple images.

As a foundation for our approach, we choose MCTFormer+ [Xu *et al.* 2024], a high-performing transformer-based model, which is discussed in detail in Section 4.3. Our proposed framework builds on MCTFormer+ by introducing two parallel branches, where cross attention is applied between them to enhance segmentation performance. The detailed mechanics of this cross attention mechanism are presented in Section 4.5.1 and our design choices are further explored through ablation studies presented in Chapter 5.

Furthermore, we introduce a contrastive learning module as an additional enhancement to our framework in Section 4.5.2. This module aims to improve the alignment of representations between the two branches, thereby increasing the effectiveness of the cross image attention mechanism. The development and investigation of this module are thoroughly discussed in Chapter 6.

Section 4.4 provides the experimental details of our approach, while Section 4.5 presents the final methodology and results. Ultimately, we demonstrate that our cross attention-based approach effectively mitigates the detrimental effects of class co-occurrence, achieving substantial improvements over previous methods, particularly for co-occurring classes.

4.1 Background

This section will introduce the foundational concepts required to understand and perform Cross Image Attention. Seeing as Vision Transformers are the central vehicle for cross attention, the concepts will be introduced within their context. Specifically, the mechanics of using a Vision Transformer to solve computer vision tasks will be discussed as well as their central self-attention mechanism.

There is no consensus about the correct way to implement Cross Attention in Vision Transformers — it is dependent on the work at hand and the task to solve. These methods will be discussed in Section 4.2.

4.1.1 Vision Transformers

Transformers [Vaswani *et al.* 2017] were originally designed to address Natural Language Processing (NLP) tasks. They have since become the *de facto* standard for such tasks due to their dominance over alternative architectures like LSTMs. Many attempts have been made to use transformers in a vision capacity, but the Vision Transformer [Dosovitskiy *et al.* 2020], ViT, was the first to demonstrate any success. They did so by following the ideology of Vaswani *et al.* [2017] as closely as possible, tweaking the ideas as necessary to accommodate an image-based context.

4.1.1.1 Basic Mechanics

Broadly speaking, transformers have two components: an encoder and a decoder. The encoder, shown in Figure 4.1, consists of a number of layers of Multi-Headed Self-Attention (refer to Section 4.1.1.4) and MLP (Multi-Layer Perceptron) blocks. It takes a

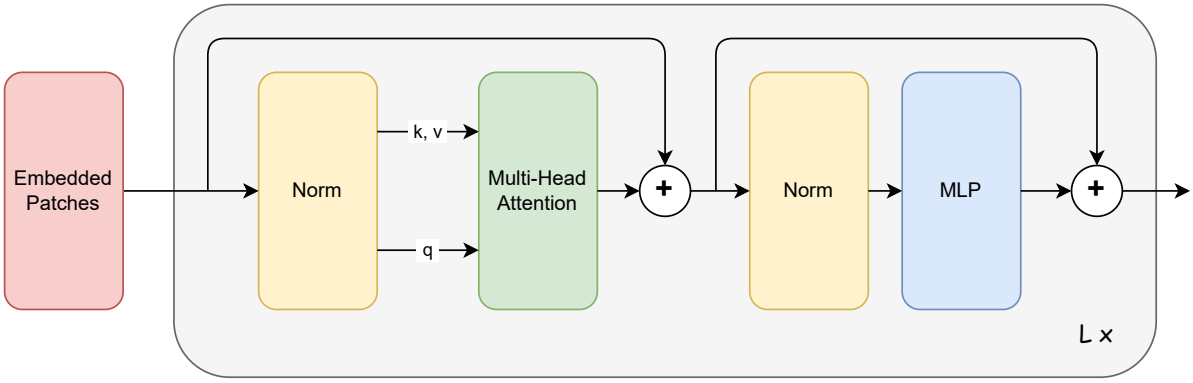


Figure 4.1: Transformer Encoder with a normal attention mechanism, where k, v, and q represent the Key, Value, and Query matrices, respectively.

1D input sequence of tokens and learns some representation between these tokens. The decoder takes the output of the encoder and produces an output sequence of tokens.

In an NLP case, we take an encoded/embedded sentence and produce another sentence. Images are 2D by nature and so need to be divided into M small $N \times N$ patches (called Patch Tokens), which are then rearranged into a linear sequence in order to be passed through the encoder. Most vision applications do not require the use of a decoder, since no output sequence is required. Instead, the learnt embeddings, or Patch Tokens, are used directly for the downstream task.

4.1.1.2 Class and Patch Tokens

When using a Vision Transformer, the input image is divided into M small $N \times N$ patches. Each of these patches is passed through a convolutional layer to learn an initial embedding for each patch — these embeddings are called Patch Tokens. Specifically, the convolutional layer maps the $N \times N \times 3$ RGB-encoded patch to a specified number of dimensions, d, to form the $1 \times 1 \times d$ patch token.

In both NLP [Devlin *et al.* 2018] and CV [Dosovitskiy *et al.* 2020] contexts, a learnable embedding is prepended to the sequence of word or patch tokens. This learnable embedding is called a Class Token. It serves to capture context information that is useful for the task at hand. Hence, it is generally used with some prediction head for the downstream task.

As the Class and Patch Tokens move through the transformer encoder, their embeddings evolve to take on a representation required to solve the training task.

4.1.1.3 Positional Embeddings

Transformers are non-sequential and compute the attention between all the tokens simultaneously (hence the name *self*-attention — discussed in Section 4.1.1.4). This means that the order of the input tokens does not matter when computing the attention. However, the order of the tokens is still important to the solving of the task (e.g. the content of adjacent patches should be more correlated than patches which are far

apart). Therefore, we add a positional embedding to each of the class and patch tokens in order to model the sequential dependence between the input tokens.

This positional embedding can be computed in two ways. Firstly, a function (e.g. a periodic function like $\sin$, $\cos$, or some combination thereof) can be used to directly calculate its values for every position in the input sequence. This method will mean its value is fixed. Secondly, it can be learnt during the training process. The former has the benefit of being fast and cheap to compute, but only possesses basic information. The latter has the benefit of more informative embeddings making more useful contributions to the task at hand, but introduces extra computation because they need to be learnt.

4.1.1.4 Self-attention

The core contribution of the Transformer architecture is the self-attention mechanism. Attention is the concept that allows the network to model the relevance of input tokens and self-attention is the concept whereby the relevance of the tokens is modelled relative to each other and to themselves (hence the *self* in self-attention).

The input class and patch tokens have three "hidden" representations: Query, Key, and Value, which function similarly to the hidden states of RNNs and LSTMs. Queries and Keys are used to model relevance between tokens, where each token asks the same question to all others with a query vector, and provides the same answer to each query with a key vector. Tokens with matching queries and keys have high relevance and so have high attention between them, whilst those without matching queries and keys have low relevance and attention between them. Non-linear combinations of query and key vectors are enabled through the value vector, allowing for expressive modelling of input tokens.

All three vectors have the same dimensionality, d, and are the output of three separate dense layers in a particular transformer layer. Consider that we have a batch of input token sequences, each of length N. Then we will have N query, key, and value vectors. Let $\mathbf{X}$ be our batch of input token sequences, each of length N, and $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ are the trainable weight matrices. The Query, Key, and Value matrices collectively represent the respective vectors for the whole batch as follows:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V.$$

The attention mechanism operates by comparing tokens through the dot product of their query and key vectors, $\mathbf{Q}\mathbf{K}^T$, which determines how strongly one token attends to another. To ensure that these dot products remain numerically stable across different vector dimensions, the result is scaled by the factor $\sqrt{d_k}$: $\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}$. This scaling prevents the dot products from growing disproportionately large with higher dimensions, which would otherwise push the softmax function into regions where it produces very sharp distributions and unstable gradients. The resulting scaled values form the attention weights, which quantify the relative importance of tokens to one another.

In order to utilise context from multiple other tokens, we cannot select the maximum attention weight per query (i.e. the most relevant key per query) since this throws

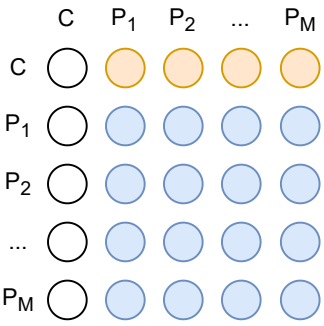


Figure 4.2: Illustration of the resulting attention matrix after self-attention (Equation (4.1)) is applied to a single class token and patch tokens. The orange elements represent the "class-to-patch" attention and the blue elements represent the "patch-to-patch" attention.

away most of the information. Therefore, we use a query-wise Softmax activation on the attention weights, which has the added benefit of being differentiable. Thereafter, the differences between large and small attention weights is smaller and smoother, allowing the gradients to flow more easily during backpropagation. The scaling by $\sqrt{d_k}$ prevents the saturation of the softmax distribution in high dimensions (which the class and patch tokens, and the query, key, and value vectors usually are).

Finally, we take the dot product of the activated attention weights with the value vectors. Therefore, the final attention is calculated as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{4.1}$$

Figure 4.2 provides a graphical interpretation of what different elements in this attention matrix, $\mathbf{A}$, represent at an image level. The orange elements show the "class-to-patch" attention, which models the relevance between the class token and every patch in the image. For example, this could be used to predict a saliency map of the foreground objects in the image. The blue elements show the "patch-to-patch" attention, which models the relevance between different patches and how they are related to each other. For example, patches with the same objects will be more related than patches with different objects.

4.1.1.5 Multi-Headed Self-Attention (MHSA)

A transformer uses multiple parallel applications of the self-attention mechanism (called "heads"), at each layer, to sufficiently model the complexity of the problem, where each head has its own set of learnable parameters. Each attention head (within a layer) will be slightly different to allow tokens the opportunity to attend to alternative tokens.

4.2 Related Work

Vision Transformers (Section 4.1.1) have seen a rapid rise in popularity since they were introduced [Dosovitskiy *et al.* 2020; Touvron *et al.* 2021; Caron *et al.* 2021]. Many attempts have also been made to use Vision Transformers for Weakly Supervised Semantic Segmentation, with even the simplest of frameworks [Li *et al.* 2023; Xu *et al.* 2024] achieving competitive results with state-of-the-art CNN-based frameworks like Zhou *et al.* [2022]. A transformer-based framework, SeCo [Yang *et al.* 2024], is currently the state-of-the-art for this task. Although SeCo is transformer-based, its primary contribution is with contrastive learning and their acknowledgement of the co-occurring problem. Therefore, it is discussed in Section 3.3 and Section 6.2 for its co-occurrence and contrastive learning content, respectively. Additionally, MCTFormer+ [Xu *et al.* 2024] is extremely competitive to SeCo, and is the work we use as our foundation — it is presented in-depth in Section 4.3. We use MCTFormer+ over SeCo as our foundation because SeCo was only released after this work had begun.

Weakly Supervised Semantic Segmentation contains the large sub-problem of being able to model long-range dependencies to get a holistic view of the objects. CNN-based approaches (Section 2.3) struggled to do this since CAMs only localise part of the objects and CNNs only have local receptive fields. Transformers, on the other hand, are perfectly suited to this problem with their self-attention and lack of local inductive biases. Figure 4.3 supports this, by showing that long-range dependencies are modelled from early on.

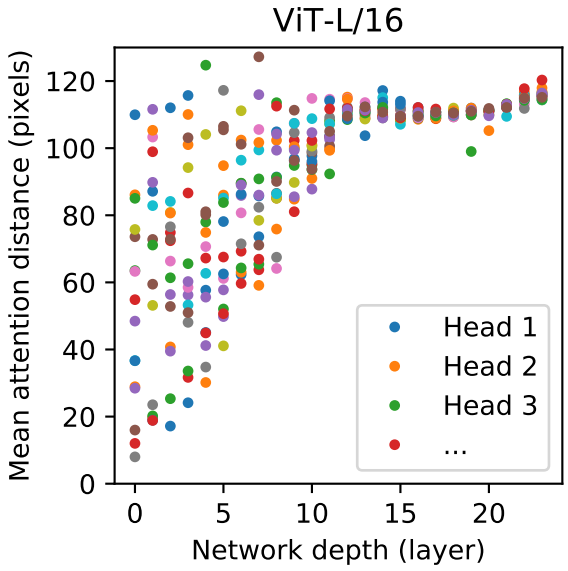


Figure 4.3: Relation between the head and network depth with the size of the attended area. Early layers have pixels attending to others, both near and far. Later layers have pixels only attending to other, far pixels. From Dosovitskiy *et al.* [2020].

It is worth noting that DINO [Caron *et al.* 2021] has provided evidence that the class token's attention maps contain clear information about the layout of the scene and its semantic segmentation. Moreover, the research shows that different attention heads

can focus on various semantic regions of the image. However, it is currently unknown how to relate a particular attention head to a specific semantic class since the attention maps do not depend on the class. Therefore, the self-supervised approach presented in DINO, though suitable for an unsupervised training objective, is not useful for our weakly supervised task.

This section will consider various transformer-based approaches with a focus on cross attention schemes.

4.2.1 Cross Attention

Attention models relevance between tokens, with questions (queries) and answers (keys) being compared, such that we only get a large attention value for "relevant" combinations of tokens. That is, we only get a large attention value if the key matches the query. Intuitively, this means that we will only get attention for correlated regions, so if we do not have classes present in various regions, the attention maps for the object class tokens should be inert, whilst the background attention maps should be largely activated.

The query, key, and value matrices are transformed by their respective set of learnable parameters, so this "question-answer" analogy does not strictly say that the query and key matrices are appropriately aligned, but rather that the learnt transformations are.

Cross Attention is a scheme whereby the attention of a transformer is spread to include tokens other than what was present in its own input. It allows the modelling of relevance across different input sequences and forms a way of transferring knowledge between different token sequences.

There are multiple different ways cross attention can be applied. Initially, images could be concatenated together, tokenised, and used as input. However, the self-attention mechanism has quadratic complexity, meaning that this naive approach is computationally expensive. Alternatively, two parallel branches can be used, with cross attention occurring between them. Traditionally, the Query and Key-Value pair have different sources, where the Key and Value source is kept constant, but the Query source alternates between the branches. This allows one image to ask questions of a different image. This is akin to the scheme used in Chen *et al.* [2023c], and which we diagrammatically present in Figure 4.4.

However, Chen *et al.* [2021] raised the idea of only swapping the class token between the parallel branches, in an alternate manner. The class token is the *only* Query source, whilst the entirety of the newly modified token sequence is the source for the Key and Value. This way, the complexity of the cross attention is linear in computation and memory since the Query does not include the patch tokens, but the holistic representations of object characteristics contained in the class token are shared. This is diagrammatically presented in Figure 4.5.

A final way of applying cross attention is to change the scope being considered. Lin *et al.* [2022] alternately apply cross attention between the patches of the input sequence and within each patch itself. The latter is the major contribution and is accomplished

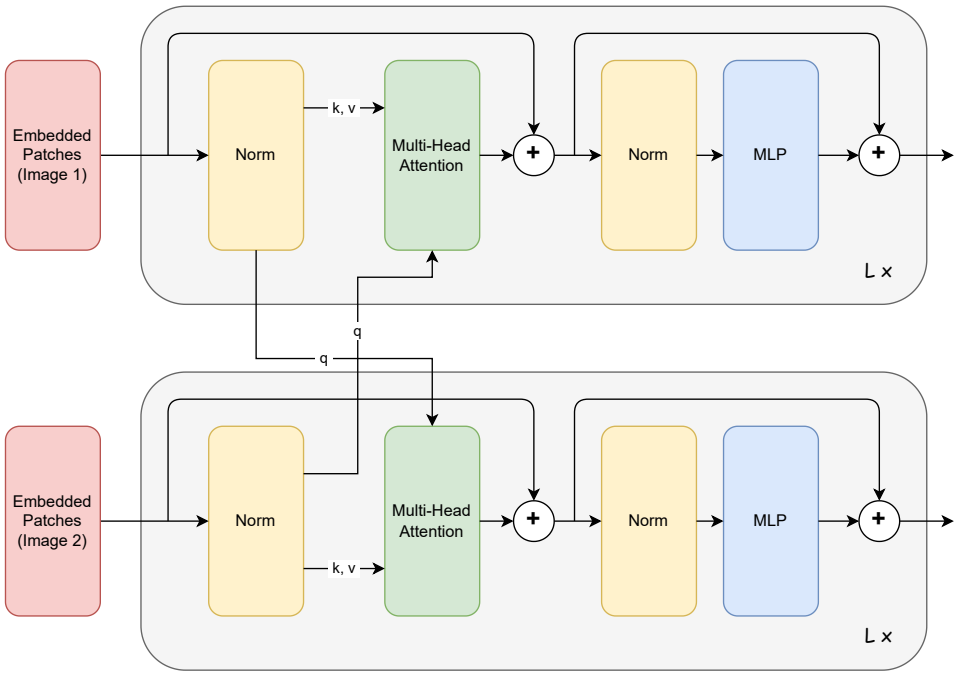


Figure 4.4: Two branches of Transformer Encoders with the conventional **Query Swap** method of cross attention. k, v, and q refer to the Key, Value, and Query matrices of the attention mechanism, respectively.

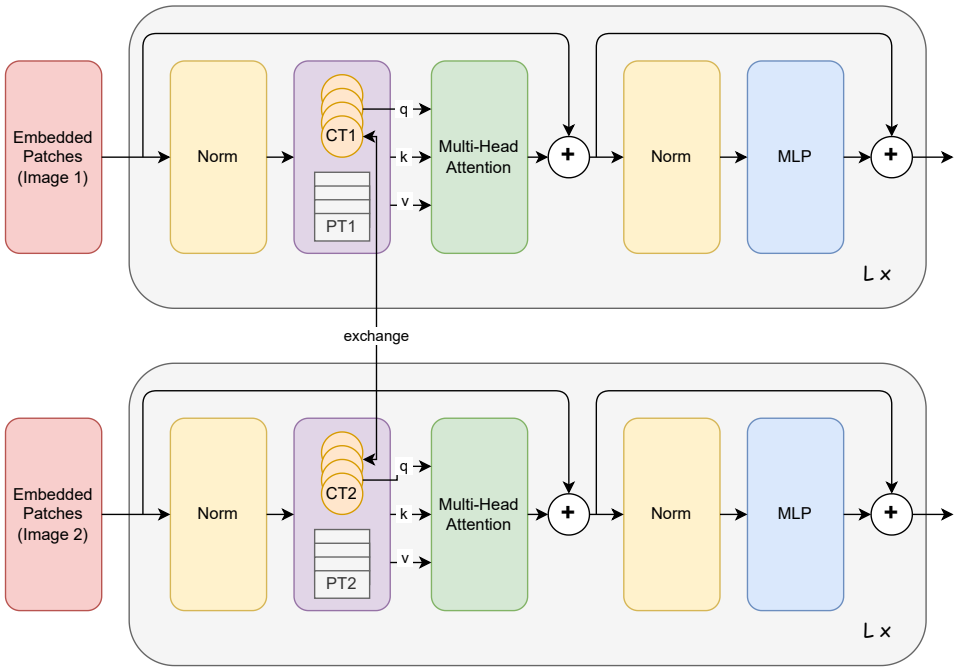


Figure 4.5: Two branches of Transformer Encoders with the conventional **Token Swap** method of cross attention. CT stands for Class Token; PT stands for Patch Token.

by applying convolutions within the patch. This allows the attention mechanism to capture the local and fine details that CNNs excel at, but also allows the modelling of global dependencies with the conventional attention module.

While all these cross attention methods show merit, we use the method of swapping the Query matrix between different attention modules (from Chen *et al.* [2023c] and illustrated in Figure 4.4) due to the attractiveness of its intuition — one image can "ask questions" about the content in another image by swapping the query matrices, which is not necessarily clear in the other methods.

4.2.2 Semantic Reassignment

Semantic Reassignment [Cao and Zhang 2023] is a method of recovering class-specific attention maps from the multiple class-agnostic attention maps of the attention heads across the transformer layers. It works as follows: given the classification score y_c for a class c, compute the gradient matrix $\nabla \mathbf{A}_l^{i,c}$ with respect to the attention matrix of the i^{th} head for the l^{th} layer, $\mathbf{A}_l^i$, and obtain a class-specific attention matrix with element-wise multiplication:

$$\mathbf{A}_l^{i,c} = \mathbf{A}_l^i \odot \text{ReLU}(\nabla \mathbf{A}_l^{i,c}). \tag{4.2}$$

The ReLU ensures that only positive attention scores (i.e. attention between relevant tokens) are considered, whilst those corresponding to the other classes are discarded. Finally, the aggregated class-specific attention matrix is:

$$\mathbf{A}^c = \sum_i \sum_l \mathbf{A}_l^{i,c}. \tag{4.3}$$

Xu *et al.* [2022] propose an alternative approach for extracting class-specific attention maps from the layers of a transformer encoder. As detailed in Section 4.3.1, this method employs a class-aware training strategy alongside the use of multiple class tokens, with one token allocated for each class in the dataset. This design encourages each class token to learn a representation specific to its corresponding class. Referring to the attention matrix illustrated in Figure 4.2, where the orange tokens represent class-to-patch attention, this approach effectively extends the concept by introducing multiple sets of class-to-patch attention scores — one set for each class token. Each set models the "relevance" of a particular class to all patches, enabling the direct extraction of class-specific attention maps from the transformer. This is also the method we use.

4.2.3 Efficient Training of Transformers

As Dosovitskiy *et al.* [2020] reported, Vision Transformers require an extremely large amount of data for training (at least 14 million images), in order to perform well on a small- to medium-sized downstream task. Obtaining this quantity of data is not feasible for our Weakly Supervised Semantic Segmentation task. However, Touvron *et al.* [2021] presented a Data-efficient Image Transformer (DeiT), which requires significantly less task-specific data for training, making it a viable backbone for our task. It uses a teacher-student training strategy, where the teacher is pre-trained on a large task

of interest (note that this teacher does not have to be a transformer). It appends a distillation token to the input token sequence of the student. As such, it interacts with the class and patch tokens through the self-attention mechanism. For the student training, the class tokens are trained against the true label, and the distillation token is trained to match the output of the teacher for the same input.

Following MCTFormer+ [Xu *et al.* 2024], we do not directly use the DeiT in our work, but we do use its learned parameters to initialise our small vision transformers, thereby using pretrained weights that already efficiently encode foundational elements for WSSS.

4.3 Base Model

MCTFormer+ [Xu *et al.* 2024] is amongst the work that has achieved the state-of-the-art performance for Weakly Supervised Semantic Segmentation, to which previous chapters alluded. It makes two improvements over its predecessor, MCTFormer [Xu *et al.* 2022], which also achieved competitive results when it was published.

Therefore, we use MCTFormer+ not only as our primary baseline, but also as the foundation for our contribution in this work.

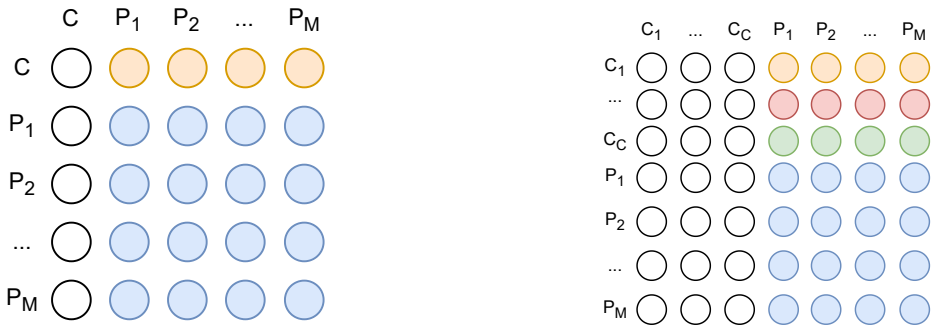
This section will initially present MCTFormer (Section 4.3.1), highlighting the core concepts which are relevant to our work, before presenting the improvements of MCTFormer+ (Section 4.3.2). Section 4.3.3 and Section 4.3.4 present the mathematical formulations for recovering the class-specific localisation maps and their subsequent refinement, respectively. Section 4.3.5 then compares the performance of our reimplementation against the reported results.

4.3.1 MCTFormer

MCTFormer is a transformer-based approach to Weakly Supervised Semantic Segmentation. It builds on the encoder portion of Vision Transformers [Dosovitskiy *et al.* 2020] (discussed in Section 4.1.1), which make use of a single, learnable class token (prepended to the patch tokens) to allow the model to learn a global representation of the image, and which can be trained to help perform the given task.

Xu *et al.* [2022] recognise that in a WSSS context, the class-to-patch attention[1] for this single class token can be used to recover a class-agnostic localisation map. They further argue that the use of multiple class tokens (one per class) will allow the model to learn class-specific attention maps, which is of immediate interest to WSSS, as it more holistically encodes the characteristic of each object type as opposed to merely encoding the existence of objects in the image, thus allowing us to recover localisation maps on a class-by-class basis. Figure 4.6 provides a graphical interpretation of this multiple-

[1] Refer to Section 4.1.1.4 for the description of the Attention Matrix.



(a) Single Class Tokens

(b) Multiple Class Tokens

Figure 4.6: Illustration comparing the class-to-patch (orange, red, green for different classes) and patch-to-patch (blue) attention for an attention mechanism using (a) a single class token and (b) multiple class tokens (as in MCTFormer and MCTFormer+).

class-token concept: the orange, red, and green elements represent the class-to-patch attention for different classes, and each set can be used to recover their respective class's localisation map.

However, the multiple class tokens must be accompanied by a suitable, class-aware, training strategy to enforce that each class token learns the embedding for a particular class. Consequently, in order to provide class-aware supervision, Xu *et al.* [2022] apply average pooling to each class token (once they have passed through the transformer encoder) to produce a scalar class score. This class score is passed through a sigmoid activation function, before being directly compared to a binary-encoded vector of the corresponding image-level class labels.

Intuitively, each class token should learn the representation for its class and be independent of whether the other classes are present or absent. Therefore, when deciding whether this class is present or not, we should only use information contained in this class token. Average pooling is applied to each class token separately in order to obtain a single initial score per class. However, this score is largely meaningless right now. It needs to be compared to a binary-encoded class label vector, which means we need to map the predicted class score to $[0, 1]$, which is easy to train against binary labels. This is done using the sigmoid activation function: class tokens with a positive average are generally considered "present" and those with a negative average are considered to be "absent".

Mathematically, this is:

$$\mathbf{y}(c) = \frac{1}{D} \sum_{j=1}^{D} \mathbf{T}_{cls}(c, j), \tag{4.4}$$

where $\mathbf{y} \in \mathbb{R}^C$ is the class prediction, $c \in \{1, .., C\}$ is the specific class, and $\mathbf{T}_{cls}(c, j)$ denotes the j^{th} feature of the c^{th} class token.

Figure 4.7 displays the architecture for MCTFormer. The authors found that additional classification training for the patch tokens, by way of the CAM-based [Gao *et al.* 2021;

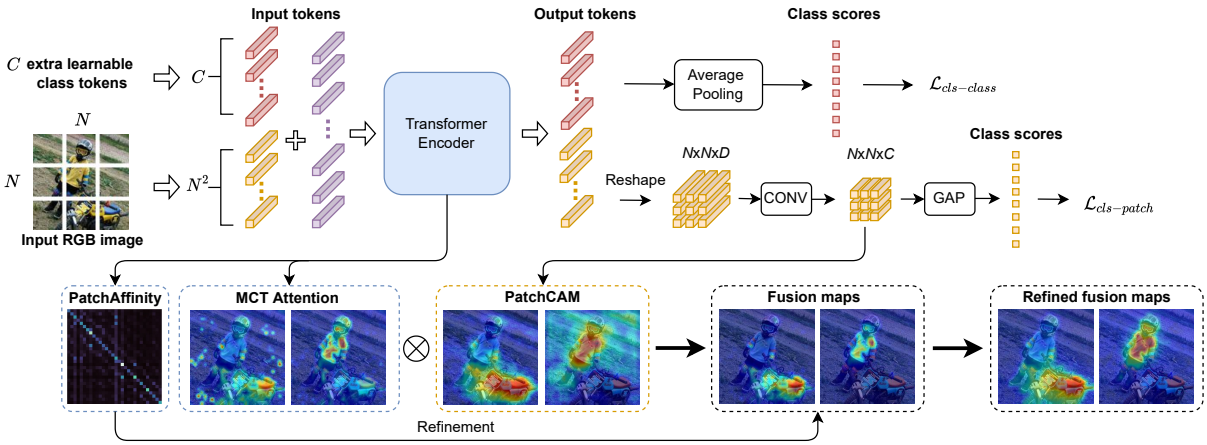


Figure 4.7: MCTFormer Architecture (from Xu *et al.* [2022]). Multiple, learnable class tokens (red), equal to the number of classes in the dataset, are prepended to the patch tokens (yellow), and added to positional embeddings (purple). Once the tokens have passed through the transformer encoder, average pooling is performed on the class tokens to obtain final class scores, for use in the loss calculation (this is the class-aware training strategy). Additionally, the patch tokens are reshaped to their original 2D dimensions, where the depth dimension is equal to the token's embedding dimension. They are passed through a convolutional layer to convert the depth dimension to be equal to the number of classes. Similarly to the class tokens, average pooling is performed on each depth dimension to get a class score, for use in the loss calculation. This lower branch for the patch tokens is called the PatchCAM module. Hence, classification supervision is applied to both the class and the patch tokens. Finally, class-specific attention from the transformer's attention maps is combined with the class activation maps from the convolutional layer of PatchCAM to obtain the Fused Attention Maps. Patch Affinity (patch-to-patch attention from the transformer's attention maps) is used for further refinement, resulting in the final, class-specific localisation maps.

Zhang *et al.* 2018; Zhou *et al.* 2016] PatchCAM module, allowed the class-specific attention maps from the transformer to be refined by a class activation map from PatchCAM's convolutional layer, and thereby improved the coverage (and accuracy) of the resulting class-specific localisation maps.

The major weakness of MCTFormer is the way the pseudo labels are constructed from the attention maps. The process of extracting the attention maps from the transformer layers and then refining them is described in Section 4.3.3 and Section 4.3.4, respectively. However, this extraction and refinement yield class-specific activation maps — these are not the pseudo labels, but are merely improved forms of describing the relative strength of each class at a pixel. In order to obtain pseudo-labels, the most relevant class needs to be selected for each pixel, which needs to include consideration for the background class. This is done by performing a background threshold search where a single "activation" value is chosen for the background class and then the class with the highest activation for a pixel gets assigned to that pixel. The background threshold is found by performing a grid search over possible values and finding the one which provides the highest mIoU with respect to the ground-truth pixel masks. Note that the

threshold is actually a hyperparameter and the selection strategy MCTFormer uses is to perform a search over the labelled training set. It is this comparison to the ground-truth pixel masks that is the weakness, since MCTFormer is no longer strictly using image-level labels, and so the "weakly supervised" constraint is marginally violated ("marginally" because pixel-level supervision is not actually included in the training of the model).

4.3.2 MCTFormer+

MCTFormer+ makes two improvements over MCTFormer, and is amongst the work to achieve the most recent state-of-the-art in overall performance (in terms of mIoU). The first improvement is to replace the Global Average Pooling in the PatchCAM module with Global Weighted Ranking Pooling [Kolesnikov and Lampert 2016]; the second improvement is to introduce contrastive learning for the class tokens at each layer of the transformer. These improvements are shown in Figure 4.8.

Global Average Pooling (GAP) has the primary drawback of assigning equal weight to each patch token when calculating its final, predicted class score [Kolesnikov and Lampert 2016]. Xu *et al.* [2024] recognised that this encourages the model to have a high response for all relevant regions of the image, leading to the size of object regions being overestimated, thereby including background regions as part of the object. They addressed this by replacing the GAP layer in the PatchCAM module with a Global Weighted Ranking Pooling (GWRP) layer [Kolesnikov and Lampert 2016], which aggregates the patch tokens by weighting each one in proportion with its activation relative to other patch tokens' activations. This allows the more informative patch tokens to contribute more to the final class prediction, thereby reducing the degree to which underestimation and overestimation of an object's boundaries occur. Mathematically, given N^2 patch tokens (N is the number of patch tokens along the length and breadth of the image) and C classes, this is:

$$G_c(\mathbf{P}) = \frac{1}{S(\lambda)} \sum_{j=1}^{N^2} \lambda^{j-1} \mathbf{P}^{r_j,c}, \tag{4.5}$$

where $\mathbf{P} \in \mathbb{R}^{N^2 \times C}$ denotes the patch tokens after having passed through the convolution layer of PatchCAM to reshape them from d dimensions to C dimensions; $G(\mathbf{F}) \in \mathbb{R}^C$ denotes the predicted class scores, and $G_c(\mathbf{F}) \in \mathbb{R}$ denotes the class score for a particular class, c; r_j denotes the ranking index, such that $\mathbf{F}^{r_1,c} > \mathbf{F}^{r_2,c} > ... > \mathbf{F}^{r_{N^2},c}$, where $\mathbf{F}^{\neg,c}$ is the c^{th} dimension of patch token $\mathbf{F}$, corresponding to class C; and $S(\lambda) = \sum_{j=1}^{N^2} \lambda^{j-1}$, where λ is the decay rate.

The second improvement is the introduction of contrastive learning on the class tokens at each layer of the transformer, by way of a Contrastive Class Tokens (CCT) module. This will be discussed in-depth, and mathematically, in Chapter 6 (where we include our own investigation using contrastive learning). Briefly, Xu *et al.* [2024] noticed that the class-to-patch attention maps (i.e. the portion of the attention matrix that tells us where different classes' objects exist in an image) contained overlapping object regions,

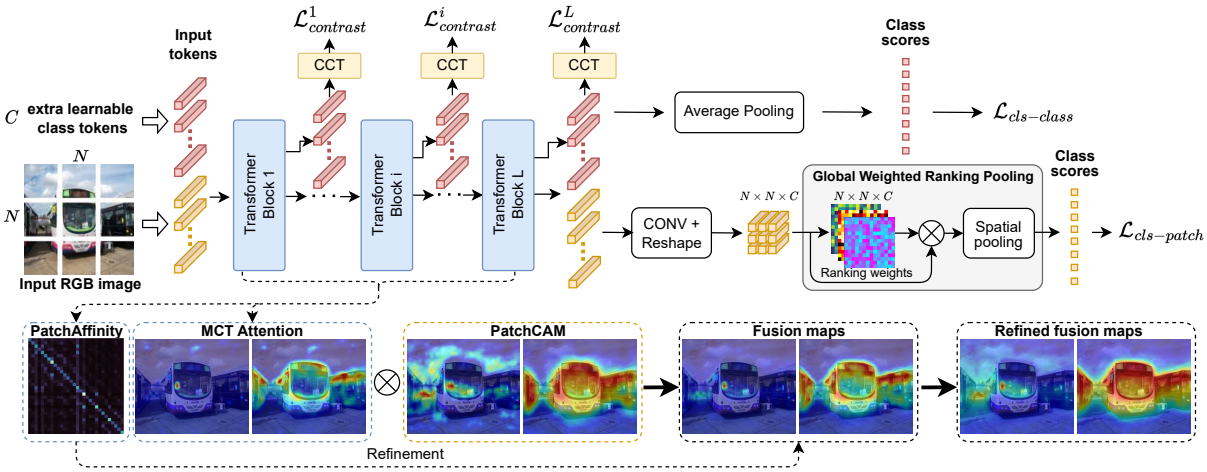


Figure 4.8: MCTFormer+ Architecture (from Xu *et al.* [2024]). The architecture of MCTFormer+ is largely the same as that of MCTFormer (Figure 4.7), except for two improvements. Firstly, the GAP layer of the PatchCAM module is replaced with the Global Weighted Ranking Pooling (GRWP) layer. Secondly, contrastive learning is applied to the class tokens at each layer of the transformer (and is called CCT here).

and that the classification loss (in this case, applied to the class tokens) only has a substantial effect for the final layers of the network, on which the classification decision is most strongly dependent. In order to provide stronger supervision to the class tokens, and to discriminate them from each other, a similarity matrix is constructed by multiplying the class token matrix by itself, such that the main diagonal is a class token multiplied by itself, and the off-diagonal elements are a class token multiplied by the other class tokens, respectively. This similarity matrix depicts the degree of latent similarity between the class tokens. The central idea is that each class should be completely separate to the other classes, and so the class tokens should have no similarity to each other. Therefore, the main diagonal elements should be perfectly correlated (since they are a comparison of a class token to itself), and the off-diagonal elements should not be correlated at all (since they are a comparison between class tokens). Consequently, the similarity matrix is encouraged, through a cross-entropy loss, to be similar to an identity matrix.

4.3.3 Attention Map Extraction

The initial contribution of MCTFormer and, subsequently MCTFormer+, is to use multiple class tokens, accompanied by the class-aware training strategy, so that the class-specific attention maps can be directly extracted from the transformer's attention matrices.

Let $\mathbf{A}^l_{t2t} \in \mathbb{R}^{M \times M}$ be an attention matrix from a transformer layer, l, where $M = N^2$, and there are $N \times N$ patches, and C is the number of classes. Then the class-to-patch

attention[2] (i.e. the class-specific attention maps) is:

$$\mathbf{A}^l_{c2p} = \mathbf{A}^l_{t2t}[1:C, C+1:C+M], \tag{4.6}$$

where $\mathbf{A}^l_{c2p} \in \mathbb{R}^{C\times M}$.

The attention maps from the last K transformer layers are aggregated (by taking the mean), where L is the number of layers:

$$\hat{\mathbf{A}}_{c2p_mct} = \frac{1}{K}\sum_{l=L-K}^{L-1} \mathbf{A}^l_{c2p}, \tag{4.7}$$

where $\hat{\mathbf{A}}_{c2p_mct} \in \mathbb{R}^{C\times M}$ and l is zero-indexed. This result is then reshaped to map the attention scores back to their original patch locations to obtain the final class-specific localisation maps from the transformer, which are then further normalised:

$$\mathbf{A}_{c2p_mct} = \text{normalise}(\text{reshape}(\hat{\mathbf{A}}_{c2p_mct})), \tag{4.8}$$

where $\mathbf{A}_{c2p_mct} \in \mathbb{R}^{C\times N\times N}$.

4.3.4 Attention Map Refinement

Many works include a refinement step, as post-processing for **inference**, to enhance the quality of the final localisation maps and, by extension, the subsequent pseudo labels. This is mainly done through the use of pairwise affinity [Ahn and Kwak 2018; Wang *et al.* 2020; Xu *et al.* 2021], which is generally achieved by learning additional parameters for this task specifically.

Both MCTFormer and MCTFormer+ use a novel approach for obtaining the pairwise affinity matrix, that does not require extra learning or supervision: it is taken directly from the transformer's attention maps in the form of the patch-to-patch attention.

Mathematically, the patch-to-patch attentions are

$$\mathbf{A}^l_{p2p} = \mathbf{A}^l_{t2t}[C+1:C+M, C+1:C+M], \tag{4.9}$$

and $\mathbf{A}^l_{p2p} \in \mathbb{R}^{M\times M}$ is reshaped to a 4D tensor, $\hat{\mathbf{A}}^l_{p2p} \in \mathbb{R}^{N\times N\times N\times N}$.

Additionally, the CAM maps of the PatchCAM convolution layer (derived from the patch tokens) are obtained by normalising the output feature map, $\mathbf{F}_{PatchCAM}$, to obtain $\mathbf{A}_{PatchCAM} \in \mathbb{R}^{C\times N\times N}$. These CAM maps are fused, via element-wise multiplication, with the attention maps from the transformer to improve the quality of the class-specific localisation maps:

$$\mathbf{A}_{fused} = \mathbf{A}_{PatchCAM} \circ \mathbf{A}_{cp_mct}, \tag{4.10}$$

where $\mathbf{A}_{fused} \in \mathbb{R}^{C\times N\times N}$ and $\circ$ denotes the Hadamard Product.

[2]Refer to Figure 4.6 for an interpretation of class-to-patch attention in a multi-class-token setting.

Dataset	Model	Includes CCT	Train Set (mIoU)		Val Set (mIoU)
			Pseudo Label	Segmentation Mask	Segmentation Mask
VOC	MCTFormer [Xu *et al.* 2022]	✗	61.7	69.1	71.9
	MCTFormer+ [Xu *et al.* 2024]	✓	**68.8**	**76.2**	**74**
	MCTFormer+ [Xu *et al.* 2024]	✗	65.9	—	—
	MCTFormer+ (ours)	✓	67.04	73.35	69.71
	MCTFormer+ (ours)	✗	66.80	73.67	69.20
COCO	MCTFormer [Xu *et al.* 2022]	✗	36.6	41.6	42
	MCTFormer+ [Xu *et al.* 2024]	✓	42.8	**48.1**	**45.2**
	MCTFormer+ [Xu *et al.* 2024]	✗	—	—	—
	MCTFormer+ (ours)	✓	43.14	43.64	42.01
	MCTFormer+ (ours)	✗	**44.69**	45.12	43.51

Table 4.1: Comparison of reported pseudo label and final segmentation results against our reimplemented results, for the *train* and *val* sets of PASCAL VOC 2012 and MS COCO 2014. MCTFormer+ only reported pseudo label results for the train set for VOC, and did not report results for COCO without using CCT (indicated with "—").

The exact refinement is as follows:

$$\mathbf{A}_{ref}(c,i,j) = \sum_{k}^{N}\sum_{l}^{N} \hat{\mathbf{A}}_{p2p}(i,j,k,l) \cdot \mathbf{A}_{fused}(c,k,l), \tag{4.11}$$

where $\mathbf{A}_{ref} \in \mathbb{R}^{C\times N\times N}$ is the refined class-specific localisation maps, c is the class under consideration, and $\cdot$ is the scalar product.

This is depicted graphically in the bottom halves of Figure 4.7 and Figure 4.8.

4.3.5 Reimplementation

MCTFormer+ [Xu *et al.* 2024] is among the current state-of-the-art works and therefore is our main baseline and the model that we use as a starting point for our own contributions. Consequently, we reimplement the model and compare the performance against the reported results.

The results are presented in Table 4.1. We test the model's performance on both our datasets, with and without the contrastive learning, CCT, enabled and compare results for the quality of the training set's pseudo labels, and the quality of the final segmentation masks for both the training and validation sets (remember that the validation sets are the *de-facto* test sets since the official test sets are held out).

The VOC results are reported for the original train set and not the augmented version. We agree that the inclusion of CCT leads to improved performance for the pseudo labels, albeit our results are less improved than Xu *et al.* [2024]'s reported results (0.24% compared to the reported 2.9%) — this is further investigated in Section 6.3.3. On the final version of MCTFormer+, which includes CCT, we struggle to achieve the same degree of success as Xu *et al.* [2024] — the segmentation performance for the pseudo labels, training and validation segmentation masks were decreased by 1.76%, 2.85%, and 4.29%, respectively. Additionally, our reimplemented final, segmented class scores

for VOC's validation test are compared to the reported results from Xu *et al.* [2024] in Table 4.2. The 4.29% discrepancy is spread out rather evenly, with the only large drops in performance happening for the plane, boat, person, and train classes.

Xu *et al.* [2024] only reported the overall COCO results, using CCT, and did not provide class scores. Our pseudo labels were able to outperform theirs slightly (0.34%), although this could be attributed to us using better quality ground-truth pixel masks[3] (Section 3.2.2.1). However, both the training and validation segmentation scores are worse than the reported results, by 4.46% and 3.19%, respectively. Interestingly, the exclusion of CCT leads to improved results for the pseudo labels and the segmentation scores on both image splits — this will be examined further in Chapter 6.

The results in Table 4.1 suggest we are able to reimplement MCTFormer+ to a good extent and have shown that their reported trends hold (the result on COCO without CCT was not originally reported, so is not relevant here). However, re-using the segmentation model (DeepLab V1 with a ResNet38 backbone) might be problematic. For VOC, our pseudo label performance with CCT falls only 1.76% short of theirs, but the training and validation segmentation masks are degraded by larger amounts, namely 2.85%, and 4.29%, respectively. For COCO, our pseudo labels with CCT outperformed theirs by 0.34%, but the training and validation segmentation masks both suffered large losses in performance of 4.46% and 3.19%, respectively. The fact that the pseudo label performance is close to or better than their reported scores, but that the segmentation mask performance is so degraded suggests there is an issue with the segmentation model in the pipeline. Nevertheless, we will use the same segmentation model, but use our reimplemented results for comparisons to our contributions. Specifically for COCO, we will mainly use MCTFormer+ with CCT as a baseline and only include the version without CCT when necessary.

Note that both MCTFormer and MCTFormer+ primarily focus on improving performance for VOC and therefore include analysis of their model on VOC only (e.g. performing ablation studies and providing empirical validation for some hyperparameter choices). Results on COCO in their papers are only included for completeness, and only at a high level (i.e. for the final version of their model). In contrast, we are primarily focusing on COCO, because of its increased class co-occurrence.

4.4 Experimental Details

This section outlines the experimental procedures and implementation details employed. The data utilised in all experiments is described in Section 3.2. As the test set for both PASCAL VOC 2012 and MS COCO 2014 is held-out, we follow prior work

[3]Intuitively, the model makes its pseudo label prediction based on image features, and measures performance by comparing the prediction to the ground-truth masks. Therefore, if the labels are of better quality, then they more closely resemble the image content, and the performance, as measured against them, will be improved.

Models	CCT	Classes (IoU)																				
		bkg	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbk	person	plant	sheep	sofa	train	tv	mIoU
MCTFormer [Xu *et al.* 2022]	✗	91.9	78.3	**39.5**	89.9	55.9	76.7	81.8	79.0	90.7	**32.6**	**87.1**	57.2	**87.0**	**84.6**	77.4	79.2	**55.1**	**89.2**	47.2	70.4	58.8	71.9
MCTFormer+ [Xu *et al.* 2024]	✓	**93.3**	**87.0**	37.8	**91.1**	**66.8**	**79.9**	**87.4**	**82.2**	**91.3**	32.1	84.8	**58.8**	86.2	82.2	**79.0**	**82.2**	54.4	87.5	**50.0**	**82.0**	57.3	**74.0**
MCTFormer+ (ours)	✓	90.21	72.70	33.52	88.89	54.11	77.74	83.40	80.41	88.19	32.23	83.39	55.51	84.46	80.28	72.96	67.09	51.30	86.48	47.76	72.70	**60.54**	69.71
Difference		-3.09	-14.30	-4.28	-2.21	-12.69	-2.16	-4.00	-1.79	-3.11	0.13	-1.41	-3.29	-1.74	-1.92	-6.04	-15.11	-3.10	-1.02	-2.24	-9.30	3.24	-4.29

Table 4.2: Comparison of reported class scores against our reimplemented results, for the *val* set of VOC. These scores are the final segmentation scores. The last shows the difference between our reimplemented scores and the reported scores (ours minus theirs).

[Ru *et al.* 2023; Xu *et al.* 2024; Yang *et al.* 2024] and use the validation set as the *de-facto* test set for reporting results. Importantly, the validation set is not used to guide model development or hyperparameter tuning. When a true "validation" set is required for specific tests, we randomly split the training set into a new training and validation set with an 80/20 ratio to prevent data leakage. However, all other experiments and descriptions of pseudo label quality present results on the entire training set.

Due to the absence of a traditional validation set, hyperparameter settings from prior work are largely adopted [Xu *et al.* 2024]. That is, we train our MCTFormer+-based dual-branch models for 45 epochs and the subsequent segmentation model for 30 epochs. This decision is further supported by Figure 5.2, which shows that training the dual-branch model for 45 epochs is reasonable and does not result in overfitting. Despite this, the batch size was adjusted from 32 to 16 to accommodate the memory limitations of the available hardware, specifically an NVIDIA RTX 3090 GPU, which has 24GB of VRAM. It is possible that the difference we see in the performance of our reimplemented MCTFormer+ model compared to the reported scores, in Section 4.3.5, may be the result of this reduced batch size.

The base model, MCTFormer+, is built using DeiT-S as the backbone [Xu *et al.* 2024; Gao *et al.* 2021; Touvron *et al.* 2021], initialized with pretrained weights from ImageNet [Deng *et al.* 2009]. Furthermore, each class token is initialised using the pretrained class token from DeiT-S [Touvron *et al.* 2021]. We also reuse data augmentation techniques and training hyperparameters established by Xu *et al.* [2024], which, in turn, are derived from the methodologies described in Gao *et al.* [2021] and Touvron *et al.* [2021].

The semantic segmentation network employed in this work follows prior conventions and utilises DeepLab-V1 [Ahn and Kwak 2018; Zhang *et al.* 2020 2021; Xu *et al.* 2021 2024] with a ResNet38 backbone [Wu *et al.* 2019]. This network is also initialised with pretrained weights from ImageNet. For inference, we adopt a multi-scale input strategy along with post-processing using Conditional Random Fields (CRFs) [Liang-Chieh *et al.* 2015].

Given the limited computational resources, it was not feasible to exhaustively evaluate all model configurations. Instead, we used configurations that previously demonstrated promising performance as a starting point and selectively explored alternative configurations as required.

The primary evaluation metric for our experiments is the Intersection over Union (IoU), as described in Section 2.2.1. While most existing work focuses on overall performance and emphasises the mean Intersection over Union (mIoU), this work also places particular emphasis on the IoU of specific co-occurring classes of interest.

The codebase for this research will be made publicly available after the dissertation examination period is completed or once the work is accepted and published.

4.5 Our Approach

This work aims to rectify the class co-occurrence problem (Chapter 3) and it shows that our framework can either maintain or improve the performance for co-occurring class pairs.

Initially, we present our final model which consists of two main concepts: cross image attention and contrastive learning. The intuition and mechanics of the final cross image attention used is presented in Section 4.5.1, and an ablation study is performed in the next chapter, Chapter 5, to justify our design choices. Only the intuition for the contrastive learning component is developed in Section 4.5.2, with the mechanics developed in Chapter 6 along with a corresponding ablation study.

Chapter 3 found that the COCO dataset has worse class co-occurrence than the VOC dataset. However, COCO is substantially larger than VOC and therefore takes longer to obtain results. Consequently, we conduct initial experiments using VOC, before transitioning to COCO for the bulk of our experiments.

Following prior work [Xu *et al.* 2024; Yang *et al.* 2024; Ru *et al.* 2023], we present segmentation performance only for the final versions of our models. Any work spent developing our framework will report the pseudo label performance because they are directly influenced by our contributions and will not be diluted or artificially inflated by a downstream segmentation model.

4.5.1 Final Cross Image Attention

Our final model builds upon MCTFormer+ [Xu *et al.* 2024] by taking inspiration from RCA [Zhou *et al.* 2022]: RCA is initially presented in Section 2.3.3, but, briefly, leverages other images in the dataset to provide contextual information when segmenting the image at hand. This is contrary to other works, which primarily focus on improving methods for extracting the maximum amount of information from a single image as opposed to utilising multiple images at once.

A natural application of multiple images' context in a transformer-based framework is to use cross image attention to share information between images. Specifically, the information needs to be exchanged between different attention modules, as described in Section 4.2.1. Additionally, although other cross attention methods exist [Chen *et al.* 2021; Lin *et al.* 2022], we choose to use the method of exchanging the Query matrices between self-attention modules due to its simple and intuitive interpretation for solving the class co-occurrence problem. As mentioned in Section 4.1.1.4, this cross

attention method can be viewed as a question-answer model, where the query vector for a class or patch token asks "questions" of all other class and patch tokens, which provide "answers" by way of their key vector.

Refining the query vector for a co-occurring class (i.e. asking more "pertinent" questions) does not immediately transfer to an increase in quality for this class's attention. This will only occur if the key vector is also pure (i.e. correlated to the refined query vector). That is, refining the query vector will only lead to improved attention if there is already a good quality and related key vector, which will only occur for a class or patch token if that class is present in the image or in that patch, respectively. If the key vector is impure, then the dot product between the query and key will show they are uncorrelated and the attention will not be impacted.

We hypothesise that for co-occurring classes in the image, the attention mechanism struggles to disambiguate them, and that by transferring information from the auxiliary image's query vector for this co-occurring class, we can refine the current image's query vector by introduce a pure and high-quality query vector from an image that does not suffer from an ambiguous nature (i.e. the auxiliary image does not have both classes present).

Our framework uses a dual-branch structure, where each branch is an identical MCTFormer+ transformer encoder, and two images are passed through the primary and auxiliary branches, respectively. Note that the terms "primary" and "auxiliary" are simply the names given to each branch to distinguish them based on the image they process; the branches are identical and are, in fact, Siamese networks, sharing both architecture and weights. The "primary" image is the image being segmented and the "auxiliary" image is the one from which additional information is taken. During training, we choose a single auxiliary image to enhance the information we have about the primary image; during inference, we simply pass the concerned image through the primary branch without having chosen an auxiliary image. That is, no cross attention is performed during inference.

Information transfer occurs by augmenting the Query vector for select layers in the Primary branch with the Query vector from the corresponding layers in the Auxiliary branch (we call this method Cross Image Attention from the Auxiliary to the Primary branch — CIA-AP). We perform an ablation study in Chapter 5 over possible augmentation methods, such as exchanging the queries between the branches entirely, which we find quite inadequate, and by summing the query vectors. Following concerns about the scaling implication of the summation, we investigate taking the mean of the query vectors, which we find the most beneficial, and so we use it here. The intention here is to artificially improve the quality of the resulting attention for image regions corresponding to the co-occurring classes, thereby promoting their inclusion in the final localisation maps for those classes.

Figure 4.9 provides a graphical overview of our framework. The classification module, PatchCAM module, and attention map refinement procedure are all inherited from MCTFormer+. Note that there are additional CCT and CCT Prime modules included, which are conceptually and intuitively introduced in Section 4.5.2.

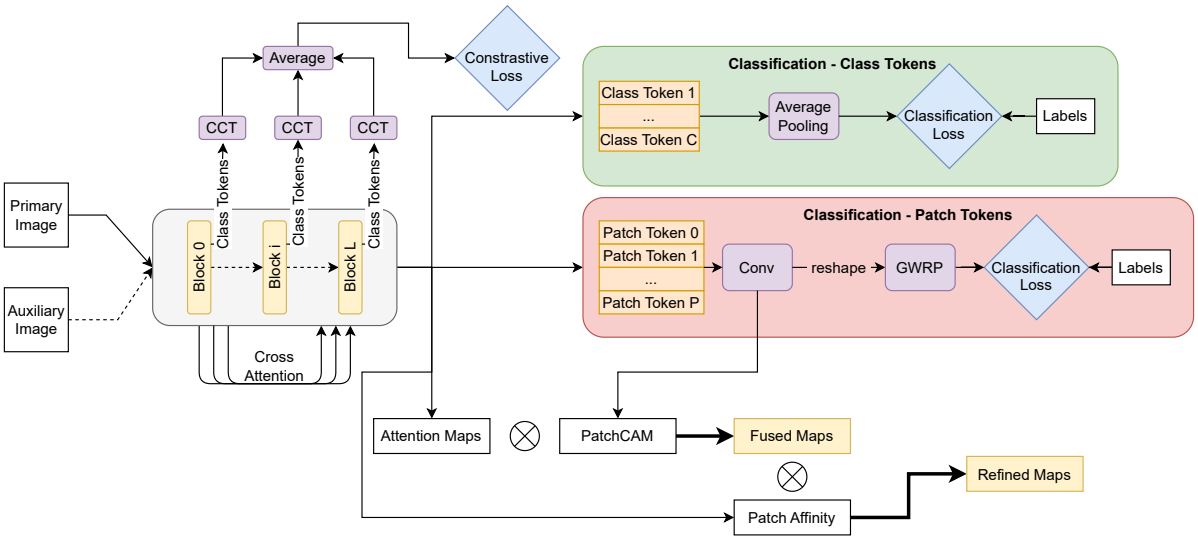


Figure 4.9: High-level Overview of our Dual-Branch model. The Primary (Image 1) and Auxiliary (Image 2) images are passed through different branches of our dual-branch shared-weight transformer encoder (the dotted line represents a different image passing through the same network). Cross Image Attention is applied from the auxiliary branch to the primary branch to enhance the attention for identified Class Co-occurrence pairs in the primary image. CCT (Chapter 6) is applied to the class tokens at each layer of the transformer and the layers' contrastive losses are averaged together to obtain a single contrastive loss value. The class tokens are then passed through a classification module (green) where average pooling is applied to get a single score per class, which is used for training against the image-level labels, after a *sigmoid activation*. Simultaneously, the patch tokens are passed through the PatchCAM module (red), where they are converted from d dimensions to C dimensions (C is the number of classes), before a Global Weighted Ranking Pooling layer is applied to get a single score per class, which is used for training against the image-level labels, after a *sigmoid activation*. The attention maps from the transformer are combined with the CAM from the PatchCAM convolutional layer, before being refined using the patch-to-patch attention as Patch Affinity, to obtain the final class-specific localisation maps.

4.5.1.1 Mechanics

Let x_p and x_a be the primary and auxiliary images, respectively, with corresponding class label sets y_p and y_a. y_p and y_a are further refined to only include pre-determined co-occurring classes, producing z_p and z_a. That is, if i and j are two classes in the dataset, they are considered to be co-occurring if $\text{Cooccurrence}(i, j) \geq t$, for a chosen threshold $t \in [0, 1]$. Only under this condition will such classes present in y_p and y_a be present in z_p and z_a, respectively. That is, for all $i, j \in y_p$, $\text{Cooccurrence}(i, j) \geq t \iff i \in z_p \wedge j \in z_p$. However, constructing z_a from y_a is contingent on z_p: for all $i \in z_p$, $i \in y_a \implies i \in z_a$, meaning that for a class to be in z_a, its co-occurring pair does not have to be in y_a. This is because x_a is chosen such that it has certain properties.

In order to refine the query vector for a class from a co-occurring pair in the primary image, the auxiliary image needs to have a pure version of this class's query vector. That is, if the primary image has both classes from a co-occurring pair, then the auxiliary image must only have one of them. Therefore, x_a is specifically chosen such that for a co-occurring class pair (i, j) in z_p, exactly one is present in z_a. That is, if $i, j \in z_p$, then choose x_a such that $i \in z_a \wedge j \notin z_a$ or $i \notin z_a \wedge j \in z_a$, but not both. We do not restrict which class of the co-occurring pair is present in the auxiliary image.

Let $s \in S$ be the set of co-occurring class indices that appear in both the primary and auxiliary images. That is, $S \in \{0, 1, \cdots, C-1\}$ is the set of all class indices, and $s = z_p \cap z_a$. From this, we know, by definition, $z_p \subseteq y_p \subseteq S$ and $z_a \subseteq y_a \subseteq S$.

Now, let $\mathbf{Q}_p^l$ and $\mathbf{Q}_a^l$ be the learnt query matrices for a particular transformer layer, $l \in \{0, 1, \cdots, L-1\}$, for the primary and auxiliary branches, respectively. In our case, $L = 12$, which is inherited from MCTFormer+. $\mathbf{Q}$ contains the query vectors for all the class and patch tokens in the respective image. Then the primary image's Query matrix is refined according to

$$f(\mathbf{Q}_p^l, s) \leftarrow g\left(f(\mathbf{Q}_p^l, s), f(\mathbf{Q}_a^l, s)\right), \tag{4.12}$$

where $\mathbf{Q}_{\{p,a\}}^l \in \mathbb{R}^{(C+N^2)\times d}$, C is the number of class tokens, N^2 is the number of patch tokens, and d is the dimensionality of the tokens.

g is the query augmentation function used to combine the primary query vectors with the auxiliary query vectors. We find that of the augmentation methods considered during our ablation in Chapter 5, the mean is the most favourable method. Therefore, for two sets of query vectors, A and B,

$$g(A, B) = \frac{1}{2}(A + B), \tag{4.13}$$

where $A + B$ is performed token-wise. That is, the primary query vector for a class or patch token T is added to the auxiliary query vector for token T.

f is a masking function applied to the set of query vectors such that g only operates on certain class or patch query vectors. Let $P = \{\varnothing, \{C, C+1, \cdots, N^2-1\}\}$ be the set of none or all the patch token indices, offset by the number of classes. P can be manually set not to include any patch token indices, which means the cross attention will not be

applied to any patch query vectors. Specifically,

$$f(\mathbf{Q}, s) = \mathbf{Q}|_{s \cup P}, \tag{4.14}$$

where $\mathbf{Q}|_{s \cup P}$ only includes query vectors for class or patch indices in $s \cup P$.

To reiterate, each $\mathbf{Q}$ comprises multiple, stacked query vectors, equal to the total number of class and patch tokens. By default, Equation (4.14) uses both the class and patch tokens for the Query matrix augmentation (i.e. s includes specific class indices and P includes all the patch indices), but we can also use only the class tokens by using $P = \varnothing$. The intuition behind using only the class tokens is that they are the entity that encode class information at the image level, and the patch tokens encode whatever the relevant information in that patch is. Consequently, when we perform the cross image attention, we want to exchange high-level information, and not patch-level information, since the patches in the two images will be different.

4.5.1.2 Mechanics Example

Consider a hypothetical co-occurring pair of classes *mouse* and *keyboard*, where $\mathrm{Cooccurrence}(\mathrm{mouse}, \mathrm{keyboard}) \geq t$, for the chosen $t = 0.45$. Another class, *toaster*, exists, that does not occur with *mouse* or *keyboard*. That is, $\mathrm{Cooccurrence}(\mathrm{mouse}, \mathrm{toaster}) < t$ and $\mathrm{Cooccurrence}(\mathrm{keyboard}, \mathrm{toaster}) < t$.

If we have a primary image, x_p, with all three classes, then $y_p = \{\mathrm{mouse}, \mathrm{keyboard}, \mathrm{toaster}\}$, but $z_p = \{\mathrm{mouse}, \mathrm{keyboard}\}$.

We choose the auxiliary image, x_a, with $y_a = \{\mathrm{mouse}, \mathrm{toaster}\}$ because it only has one of the *mouse* and *keyboard* co-occurring pair. Therefore, $\mathrm{mouse} \in z_p$ and $\mathrm{mouse} \in y_a$ implies that $\mathrm{mouse} \in z_a$.

Consequently, $s = z_p \cap z_a = \{\mathrm{mouse}\}$.

Therefore, when the cross image attention in Equation (4.12) is applied with the mean augmentation method from Equation (4.13), f ensures the cross attention is only applied to the query vector for *mouse* and not for *keyboard* or *toaster*. Because we have not manually set $P = \varnothing$, f also allows the cross attention to be applied to all the patch query vectors.

4.5.2 CCT Prime

An intuitive understanding of two contrastive learning models, CCT and CCT Prime, is developed in this section. This will offer foundational context when these elements are included in the remaining presentation of our final framework, but allows us to present the mechanics and analysis of these methods during their development in Chapter 6.

Contrastive Class Tokens (CCT) from MCTFormer+ [Xu *et al.* 2024], is a contrastive learning module designed to enhance the separation of class tokens within a transformer model. Its core premise is rooted in the assumption that the dataset's classes are distinct and non-hierarchical. Therefore, the class tokens, which encode each class's characteristics, should be entirely disentangled from one another.

Xu *et al.* [2024] observed that the class-specific localisation maps within the model often overlap significantly. They attributed this to the fact that the classification loss primarily influences only the final layers of the transformer, on which classification decisions are most heavily dependent. As a result, the earlier layers might not be adequately trained to distinguish between different classes. To address this, CCT attempts to provide more consistent supervision for the class tokens throughout the network.

The mechanism behind CCT involves constructing a similarity matrix by taking the matrix product of the class tokens with themselves. This operation effectively computes the dot product between all pairs of class tokens, including their self-similarity. The resulting matrix's main diagonal represents the similarity of each class token to itself, while the off-diagonal elements correspond to similarities between different class tokens.

CCT aims to transform this similarity matrix into something resembling an identity matrix: the diagonal elements are encouraged to become 1 (indicating strong similarity with themselves), while the off-diagonal elements are encouraged to become 0 (indicating minimal similarity between different classes). This approach is intended to encourage orthonormality among the class tokens. However, this objective is only applied to the tokens corresponding to classes present in a particular image, achieved through masking.

In practice, however, this orthogonal structure is not strictly achieved. Instead, the class tokens are aligned in broad directional clusters, where tokens corresponding to present classes tend to point in a distinct direction, and tokens for absent classes tend to point in an almost opposite direction. Each class token still points in a distinct direction within these general directional clusters. This creates a conic structure where the directions of class tokens are not necessarily orthogonal but are still separated enough to allow for meaningful classification.

Within our dual-branch framework, CCT is applied independently to each branch, following MCTFormer+. While this does promote a certain level of alignment within each branch, it does not guarantee alignment between the two branches. In other words, the "present" direction in the primary branch may not coincide with the "present" direction in the auxiliary branch. Despite this misalignment, the classification module in each branch can still reach correct classification decisions because, on average, the tokens for present classes will still point in the same general direction.

However, this lack of alignment between branches poses a significant challenge for the cross attention mechanism, which relies on the assumption that the representations are aligned to effectively exchange information between the branches. This is where CCT Prime comes in. CCT Prime is introduced as an additional learning module specifically designed to ensure alignment between the branches. While CCT encourages separation within each branch, CCT Prime focuses on promoting compatibility between the representations of the two branches, making the cross attention mechanism more effective.

By ensuring that the class tokens from the primary and auxiliary branches are aligned during training, CCT Prime provides a more robust foundation for the cross attention mechanism to operate. This alignment is particularly crucial when dealing with

co-occurring classes, where subtle differences between classes must be accurately identified and exchanged between branches.

4.5.3 Loss Function

The overall loss function is largely inherited from MCTFormer+, which has classification loss components for the predictions made by the classification and PatchCAM modules, as well as a contrastive learning loss component from the application of CCT. Seeing as we are using a dual-branch framework, where each branch is an instance of MCTFormer+, these two classification components and the contrastive learning component will be duplicated in that they will be present for both branches. Additionally, we have introduced another contrastive learning module, CCT Prime, conceptually in Section 4.5.2.

The loss from the classification module of each branch, which uses the class tokens to make the predictions, is:

$$\mathcal{L}^{b}_{\text{cls}-\text{class}} = \text{MultiLabelSoftMarginLoss}\left(\mathbf{y}^{b}_{\text{class}-\text{pred}}, \mathbf{y}_{\text{label}}\right), \tag{4.15}$$

where $\mathbf{y}^{b}_{\text{class}-\text{pred}} \in \mathbb{R}^{C}$ is the vector of predicted class scores for branch $b \in \{p, a\}$ (Primary and Auxiliary branches, respectively) and $\mathbf{y}_{\text{label}} \in \{0, 1\}^{C}$ is the vector of class labels. $\mathbf{y}^{b}_{\text{class}-\text{pred}}$ is obtained by passing the class tokens from the final layer of the specified branch's transformer to Equation (4.4).

The loss from the PatchCAM module of each branch, which uses the patch tokens to make the predictions, is:

$$\mathcal{L}^{b}_{\text{cls}-\text{patch}} = \text{MultiLabelSoftMarginLoss}\left(\mathbf{y}^{b}_{\text{patch}-\text{pred}}, \mathbf{y}_{\text{label}}\right), \tag{4.16}$$

where $\mathbf{y}^{b}_{\text{patch}-\text{pred}} \in \mathbb{R}^{C}$ is the vector of predicted class scores for branch $b \in \{p, a\}$ (Primary and Auxiliary branches, respectively) and $\mathbf{y}_{\text{label}} \in \{0, 1\}^{C}$ is the vector of class labels. $\mathbf{y}^{b}_{\text{patch}-\text{pred}}$ is obtained by passing the patch tokens from the final layer of the specified branch's transformer to the PatchCAM module, which passes them through a convolution layer and into a Global Weighted Ranking Pooling layer, as described in Equation (4.5).

The two contrastive learning loss components are from CCT (inherited from MCTFormer+) and an additional module, CCT Prime. The CCT loss is applied to each branch, yielding $\mathcal{L}^{p}_{\text{CCT}}$ and $\mathcal{L}^{a}_{\text{CCT}}$ for the primary and auxiliary branches, respectively. The CCT Prime loss, $\mathcal{L}_{\text{CCTPrime}}$, is applied to both branches simultaneously, and so there is only a single instance of it. The CCT and CCT Prime losses are described in Equation (6.4) and Equation (6.19), respectively.

The total loss is a weighted combination of the classification and contrastive learning loss components:

$$\begin{aligned}\mathcal{L}_{\text{total}} = {} & \alpha\mathcal{L}^{p}_{\text{cls}-\text{class}} + \beta\mathcal{L}^{a}_{\text{cls}-\text{class}} \\ & + \gamma\mathcal{L}^{p}_{\text{cls}-\text{patch}} + \phi\mathcal{L}^{a}_{\text{cls}-\text{patch}} \\ & + \psi\left(\mathcal{L}^{p}_{\text{CCT}} + \mathcal{L}^{a}_{\text{CCT}}\right) + \omega\mathcal{L}_{\text{CCTPrime}},\end{aligned} \tag{4.17}$$

where α, β, γ, and ϕ are the loss weights for the classification components and are set to 1 (which is an empirical setting inherited from MCTFormer+). ψ and ω are the loss weights for the contrastive learning components and are empirically set to 0.01 in Section 6.3.5.

4.5.4 Results

This section presents the pipeline of results using the final model configuration. Two high-performing model configurations are included, since they have merit in different areas. The complete configurations are:

1. Cross Image Attention, as described in Equation (4.12), using mean augmentation, as described in Equation (4.13).
2. In Equation (4.14), when $s = \varnothing$, we set $P = \varnothing$. Briefly, since our cross attention and mining strategies are designed to target co-occurring class pairs, when the primary image does not have a co-occurring pair of classes, then cross attention will not be performed on any class query vectors. By setting $P = \varnothing$, cross attention is not applied to the patch query vectors either.
3. Co-occurrence threshold such that the top 10 class pairs are included. This is 0.0976 for VOC and 0.2491 for COCO.
4. Cross attention applied separately to the two layer groupings 3, 5, 6, and 6, 7, 8. This results in two main main configurations, where all other settings are the same.
5. The disabling of patch cross attention mentioned in Item 2 only applies when no classes have been selected for cross attention. An additional approach is taken for the (3, 5, 6) layer model, where patches are never used for cross attention (i.e. the patches are "excluded"). The (6, 7, 8) layer model still includes them if at least one class is chosen for cross attention.
6. CCT and CCT Prime loss weightings of 0.01.

The particular choices for these parameters are examined in detail through ablation studies presented in Chapter 5 and Chapter 6. Chapter 5 focuses on investigating the effectiveness of our Cross Image Attention (CIA-AP) mechanism, including the influence of different layer groupings, augmentation methods, and patch attention configurations. Chapter 6 explores the impact of our contrastive learning modules, CCT and CCT Prime, including their weighting and compatibility with the cross attention mechanism. These ablation studies aim to provide deeper insight into the contributions of each component to the overall performance of our model.

Initially, we re-conduct the experiment from Section 3.4.2 and compare the performance of pseudo labels of our dual-branch framework to MCTFormer+, for varying levels of co-occurrence. Specifically, we only use the (6, 7, 8) layer model due to resource constraints. Figure 4.10 shows this comparison for the same class pairs chosen previously, which are distributed over high and low original levels of co-occurrence to test whether the trends hold under those different conditions. We previously found that

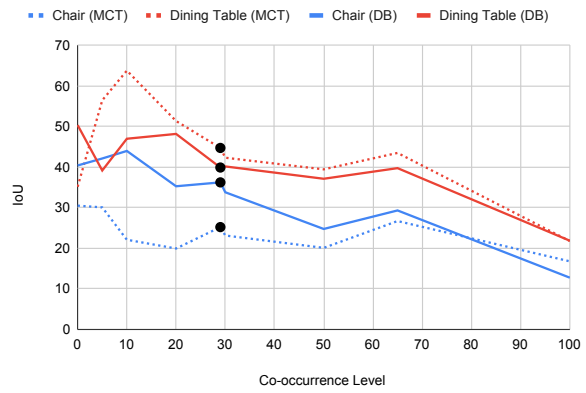
MCTFormer+ generally suffered decreasing class performance as the co-occurrence increased, and find that the same happens for our dual-branch model. However, in most cases for COCO, our model outperforms MCTFormer+ and often does so by a substantial amount. This does not hold for the Bicycle and Motorcycle class pair in Figure 4.10f, where the two models perform similarly (and MCTFormer+ even outperforms our model) for all co-occurrence levels. We conjecture the reason this occurs is due to lack of data: in all the other COCO pairs studied, there are a reasonable number of images containing those pairs which are used, but only 318 images contain both the bicycle and motorcycle pair. In order to keep the number of images constant across the co-occurrence levels for a class pair, we cannot increase this number further, and the sheer lack of images is the primary reason for the degraded performance in both models.

On the other hand, the two pairs studied for VOC generally agree with this trend. The chair class generally outperforms MCTFormer+ by a large amount whilst the dining table slightly underperforms MCTFormer+. The car and person classes perform similarly for both models. We conjecture that the lack of general difficulty and images in VOC mean MCTFormer+ already performs reasonably well, even on the higher co-occurrence class pairs, and there is little room for our targeted improvement.

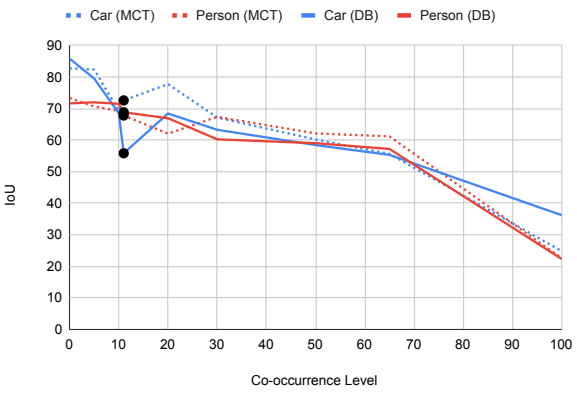
Across all the examined cases, when co-occurrence reaches 100%, the classes never occur without the other in its pair, which means the model cannot learn to effectively classify them (e.g. label flipping occurs), and the model does not have data to disambiguate their features from each other. Hence, the performance for both models is greatly reduced and cannot be reasonably compared.

Scaling up to the complete datasets, Figure 4.11 shows the performance of our two dual-branch models on the co-occurring class pairs of interest (i.e. the class pairs with high co-occurrence in each dataset). Firstly, for both VOC and COCO, our two models perform quite similarly to each other for all the indicated classes and for the overall mIoU performance, with any improvement of one over the other being marginal. For VOC, both our models outperform MCTFormer+ (both with and without CCT) for the co-occurring pair of Chair and Dining Table, but perform similarly to both baselines for the overall mIoU score. For COCO, both our models outperform MCTFormer+ (with CCT) for all the co-occurring classes and often do so by a substantial amount. For example, Baseball Glove and Mouse nearly double their performance over the baseline. Our models also outperform the No CCT MCTFormer+ baseline, except for the (3, 5, 6) layer model on the Knife class, which performs slightly worse, and the (6, 7, 8) layer model on the Knife class and both models on the Fork class, which perform on-par with the baseline. Both our models also perform similarly to the COCO overall mIoU performance.

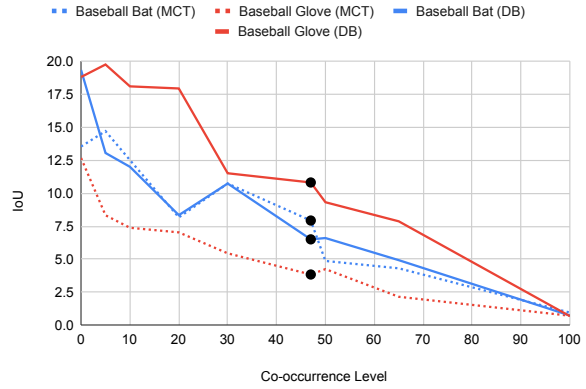
The fact that we achieve these substantial improvements for co-occurring class pairs is a resounding success. However, the mIoU remains on-par with the baseline, which means that for other, "normal" classes in each dataset, we have suffered a performance loss. The complete set of results for all the classes is presented in Table 4.3 and Table 4.4, for VOC and COCO, respectively — this includes the pseudo label performance and the segmentation mask performance. The results over the class are ordered by the



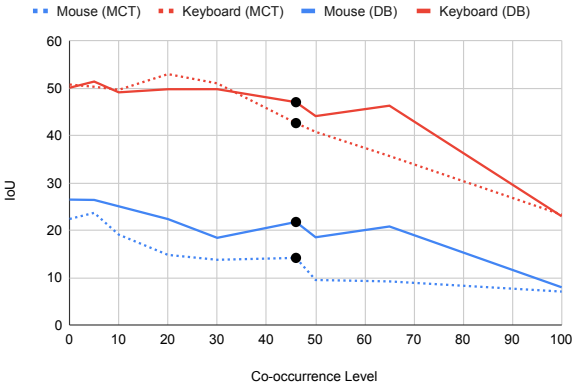
(a) Chair and Dining Table (VOC) — 29.01%



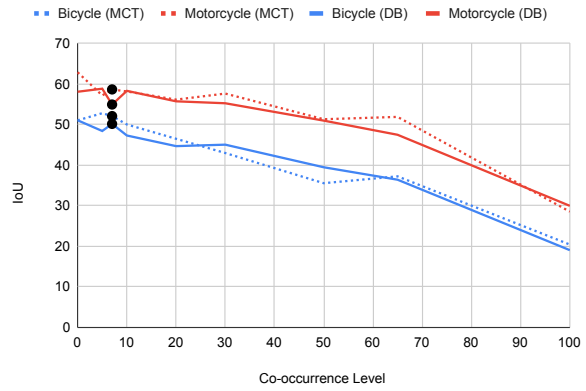
(b) Car and Person (VOC) — 10.96%



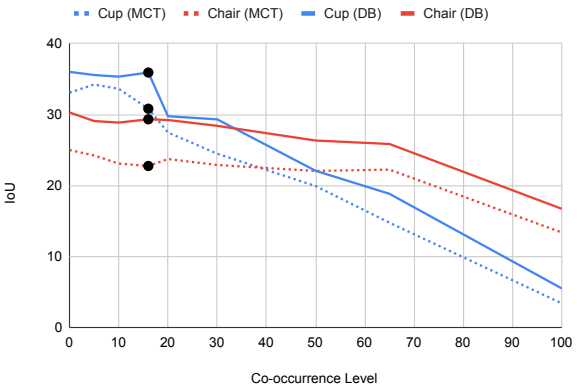
(c) Baseball Bat and Glove (COCO) — 47.23%



(d) Mouse and Keyboard (COCO) — 46.63%

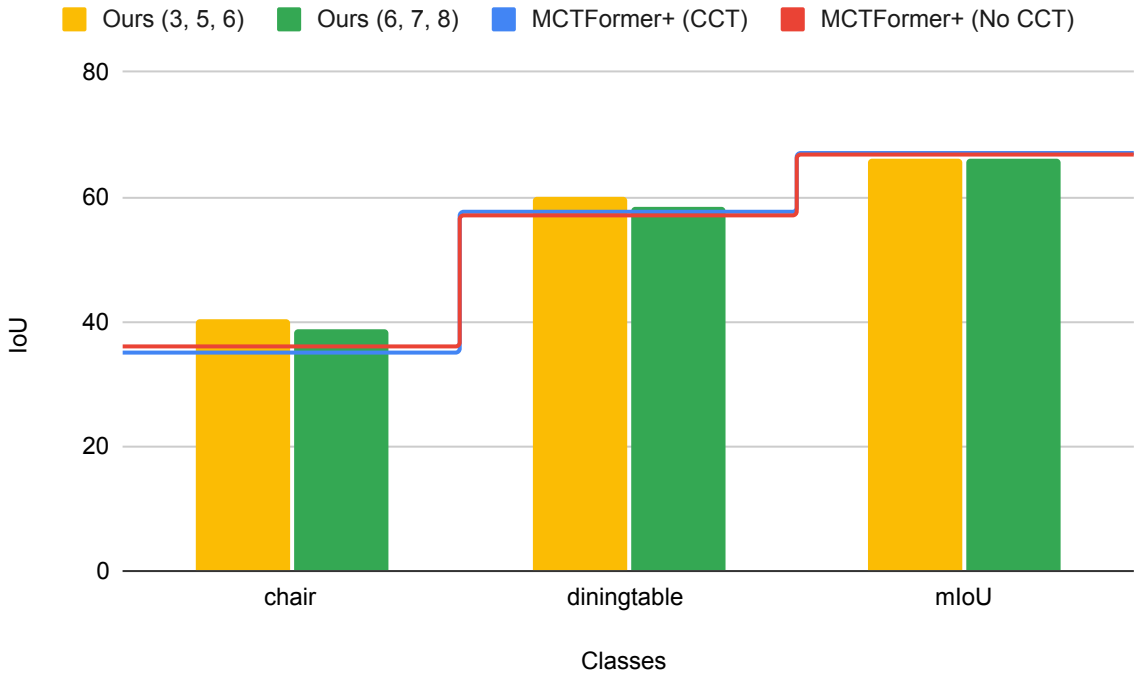


(e) Bicycle and Motorcycle (COCO) — 7.21%

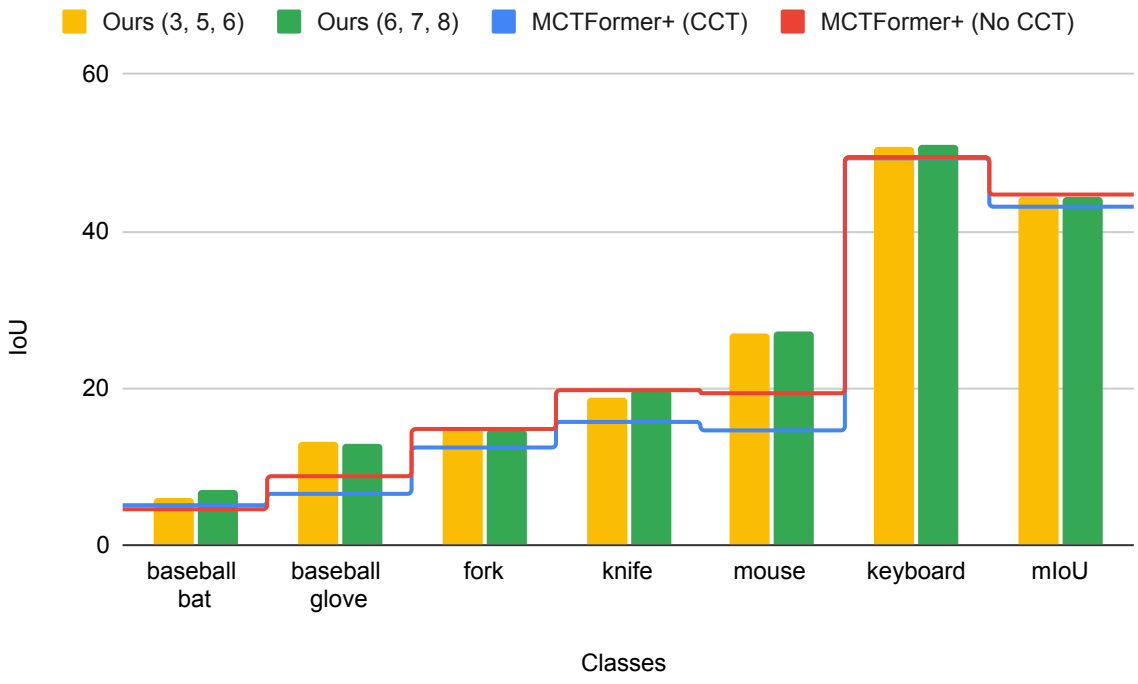


(f) Cup and Chair (COCO) — 15.88%

Figure 4.10: Comparison of MCTFormer+'s performance to our dual-branch model's performance for various pairs of classes over artificially changing co-occurrence levels. Results shown are for the training set pseudo labels. The first row comprises class pairs from VOC and the last two rows comprise class pairs from COCO. The specific class pair and its original level of co-occurrence is stated in each sub-plot's caption. Each class in a pair is assigned a colour for both models (either blue or red). MCTFormer+ results are shown by dotted lines and results for our dual-branch model are shown by solid lines. The black points denote the original level of co-occurrence for reference.



(a) VOC



(b) COCO

Figure 4.11: Comparison of the two best-performing dual-branch models to MCTFormer+ (with and without CCT) for co-occurring class pairs of interest. Results shown are for the training set pseudo labels. The dual-branch models have cross attention applied to layers (3, 5, 6) and (6, 7, 8), respectively. The upper plot is for VOC and the lower one is for COCO.

"Improvement Factor", which is the ratio of our best performing model to MCTFormer+ (with CCT) for the pseudo labels — we use the pseudo labels because they are not affected by anything other than our contributions. When considering the pseudo labels, it is evident that the small mIoU decrease of our model compared to MCTFormer+ (by $\pm 0.2\%$) is primarily due to our performance gain on co-occurring classes being offset by small decreases in performance across the other classes, which then has the cumulative effect of "cancelling out" our gain. However, both our models outperform the reimplemented MCTFormer+, with and without CCT, for the training and validation segmentation masks for COCO, and for the training segmentation masks for VOC, which, combined with the large increase for co-occurring classes, is a notable result. VOC's validation segmentation masks come quite close to outperforming MCTFormer+ (within 0.01%).

Since the results in Table 4.3 and Table 4.4 are ordered by the Improvement Factor, classes higher in the table achieved a larger performance increase using our method. The classes with high co-occurrence (which we consider) have been indicated in the table, and most of these classes are situated close to the top of the table. That is, our method performs better for classes with a naturally high co-occurrence in VOC and COCO.

Additionally, the two tables include the ratio of performance of our models to the scores obtained by training the segmentation model (DeepLab-V1) with the ground-truth masks, thereby giving us an indication of how close our models are to achieving "perfect" performance. The segmentation scores for the training and validation sets for VOC and COCO, learnt using ground-truth supervision, are presented in Table A.1 and Table A.2, respectively. Considering the VOC co-occurring classes in Table 4.3, the "chair" segmentation masks on the validation set actually outperform using the ground truth supervision. This is a phenomenon repeated for the "knife" and "fork" COCO classes in Table 4.4, except that these two classes outperform ground-truth supervision for the pseudo labels and the training set's segmentation masks as well. This is an encouraging result, since it suggests our pseudo labels are able to present co-occurring class information in such a way that the segmentation model finds it more useful than the ground-truth masks.

Possible reasons that could induce the behaviour of co-occurring classes of interest performing much better and other, non-co-occurring classes performing slightly worse are:

1. There is increased supervision for the co-occurring classes, which means the current set of weights are more motivated to solve the co-occurring cases. This could potentially leave less "capacity" in the weights for other classes. This can be confirmed by increasing the size of our models.

2. When there are no co-occurring class pairs in the primary image, then no special mining strategy is used to choose the auxiliary image. Instead, random mining is used, which would bias the auxiliary branch to receive more instances of the more common classes in the dataset. Seeing as this branch still performs classification on this image and updates the same shared set of weights, then the model could

Classes	High Co-occurence	Pseudo Labels (Train Set)				Segmentation Masks (Train Set)				Segmentation Masks (Val Set)				Improvement Factor (ours/MCT'+)
		MCTFormer+		Ours		MCTFormer+		Ours		MCTFormer+		Ours		
		CCT	No CCT	(3, 5, 6)	(6, 7, 8)	CCT	No CCT	(3, 5, 6)	(6, 7, 8)	CCT	No CCT	(3, 5, 6)	(6, 7, 8)	
person		61.20	**71.61**	70.82 (0.84)	68.31 (0.81)	66.08	**80.78**	80.64 (0.96)	79.52 (0.94)	67.09	**80.58**	80.34 (0.97)	79.91 (0.97)	1.157
chair	✓	35.03	36.02	**40.30 (0.93)**	38.71 (0.89)	36.08	34.69	37.56 (0.86)	**38.83 (0.89)**	32.23	29.73	34.12 (1.01)	**34.50 (1.02)**	1.150
boat		50.33	56.03	**56.38 (0.70)**	56.35 (0.70)	59.41	67.02	**70.64 (0.88)**	69.90 (0.87)	54.11	55.63	59.55 (0.88)	**59.79 (0.88)**	1.120
diningtable	✓	57.62	57.04	**59.92 (0.78)**	58.34 (0.76)	67.08	**67.97**	67.83 (0.88)	66.98 (0.87)	55.51	56.29	**56.92 (0.92)**	55.70 (0.90)	1.040
car		71.03	72.75	73.35 (0.87)	**73.56 (0.87)**	**80.45**	79.44	**80.45 (0.95)**	78.71 (0.93)	**80.41**	80.00	80.18 (0.97)	79.11 (0.96)	1.036
sheep		77.89	**79.83**	79.09 (0.86)	79.39 (0.86)	90.54	90.85	91.41 (0.99)	**91.59 (0.99)**	**86.48**	85.69	85.01 (0.95)	86.12 (0.96)	1.019
motorbike		73.56	71.89	72.21 (0.86)	**74.82 (0.89)**	76.21	**78.52**	78.05 (0.93)	77.69 (0.93)	72.96	**75.71**	74.29 (0.91)	75.69 (0.93)	1.017
bicycle		39.38	38.40	**39.87 (0.78)**	39.71 (0.77)	43.17	42.27	**44.41 (0.87)**	43.45 (0.85)	33.52	33.20	**33.70 (0.90)**	32.69 (0.88)	1.012
bird		71.00	**74.45**	70.04 (0.80)	71.65 (0.82)	**88.65**	87.53	83.98 (0.96)	85.98 (0.99)	**88.89**	88.34	88.25 (1.00)	88.46 (1.00)	1.009
aeroplane		64.15	63.96	64.50 (0.73)	**64.55 (0.73)**	**73.91**	72.38	73.61 (0.83)	72.34 (0.81)	**72.70**	70.53	71.97 (0.83)	71.20 (0.82)	1.006
background		87.79	**88.13**	87.79 (0.93)	87.77 (0.93)	90.69	91.20	91.27 (0.97)	**91.28 (0.97)**	90.21	90.59	**90.76 (0.97)**	90.74 (0.97)	1.000
horse		76.87	**76.92**	75.85 (0.84)	76.59 (0.85)	86.83	**87.82**	86.77 (0.96)	86.04 (0.95)	**80.28**	80.26	79.38 (0.96)	78.09 (0.95)	0.996
bus		**78.10**	75.35	76.67 (0.87)	76.81 (0.87)	82.14	82.67	**84.11 (0.95)**	83.76 (0.95)	83.40	82.70	83.31 (0.94)	**83.65 (0.94)**	0.983
train		**67.67**	65.47	66.45 (0.77)	66.54 (0.77)	72.07	72.12	72.64 (0.84)	**73.13 (0.85)**	72.70	73.70	74.43 (0.92)	**74.58 (0.92)**	0.983
cow		**79.48**	77.89	78.00 (0.91)	78.01 (0.91)	**85.08**	84.99	84.28 (0.98)	84.38 (0.98)	**83.39**	81.90	79.92 (0.97)	80.98 (0.98)	0.982
cat		**86.74**	85.65	84.07 (0.90)	84.18 (0.90)	92.41	92.54	**92.73 (0.99)**	92.65 (0.99)	88.19	88.59	88.29 (0.98)	**88.71 (0.99)**	0.970
bottle		65.10	**66.83**	61.36 (0.75)	62.81 (0.77)	78.25	77.36	75.69 (0.93)	**78.74 (0.97)**	**77.74**	75.22	73.70 (0.94)	77.02 (0.99)	0.965
dog		**84.41**	82.20	80.91 (0.87)	81.19 (0.88)	**91.09**	90.69	90.45 (0.98)	90.66 (0.98)	84.46	84.54	84.26 (0.98)	**84.89 (0.99)**	0.962
pottedplant		**52.75**	49.64	49.36 (0.70)	47.83 (0.68)	59.87	**61.42**	58.48 (0.83)	58.42 (0.82)	**51.30**	48.85	49.60 (0.95)	48.85 (0.93)	0.936
sofa		**68.29**	64.64	58.78 (0.99)	59.94 (1.01)	**56.52**	55.20	56.44 (0.95)	56.38 (0.95)	**47.76**	46.78	47.14 (0.98)	46.35 (0.96)	0.878
tvmonitor		**59.57**	48.01	44.74 (0.64)	45.09 (0.65)	**63.86**	49.62	47.94 (0.69)	49.99 (0.72)	**60.54**	44.37	44.45 (0.70)	46.62 (0.74)	0.757
mIoU		**67.04**	66.80	66.21 (0.82)	66.29 (0.83)	73.35	73.67	73.78 (0.92)	**73.83 (0.92)**	**69.71**	69.20	69.50 (0.94)	69.70 (0.94)	0.989

Table 4.3: The performance for each class of our final models on **VOC**. The IoU performance is reported for each class in the dataset, for the training set's pseudo labels, and the training and validation sets' segmentation masks (these form the three major column groupings). The *reimplemented* scores for MCTFormer+, with and without CCT, make up the first two columns in each group, respectively. The final two columns present the scores for our final model, where cross attention is applied to the layer groups 3, 5, 6, and 6, 7, 8, respectively. **For our models' results, the ratio to the performance of the segmentation model trained on the ground truth masks is shown ("∞" refers to segmentation scores of 0 when training with ground truth supervision).** The checkmark in the second column indicates the classes we have generally considered highly co-occurring. The results are ordered by the final column (the Improvement Factor), which measures the ratio of performance for the pseudo labels of our models (the maximum of both models) to MCTFormer+ (with CCT), and a line separates the classes we improved on from the ones we did not. For each of the major groupings, the highest score per class is shown in bold.

Classes	High Co-occurence	Pseudo Labels (Train Set)				Segmentation Masks (Train Set)				Segmentation Masks (Val Set)				Improvement Factor (ours/MCT'+)
		MCTFormer+		Ours		MCTFormer+		Ours		MCTFormer+		Ours		
		CCT	No CCT	(3, 5, 6)	(6, 7, 8)	CCT	No CCT	(3, 5, 6)	(6, 7, 8)	CCT	No CCT	(3, 5, 6)	(6, 7, 8)	
baseball glove	✓	6.53	8.78	**13.24 (0.32)**	12.85 (0.32)	7.23	9.78	**23.83 (0.58)**	18.65 (0.46)	5.63	8.59	**18.46 (0.55)**	14.80 (0.44)	2.028
mouse	✓	14.62	19.36	27.05 (0.55)	**27.28 (0.56)**	16.02	21.56	**38.34 (0.79)**	35.06 (0.72)	12.57	16.67	**31.30 (0.75)**	28.42 (0.68)	1.866
tie		8.82	13.23	**14.78 (0.58)**	12.99 (0.51)	7.77	14.12	**17.35 (0.68)**	13.24 (0.52)	5.81	8.59	**12.37 (0.56)**	8.61 (0.39)	1.676
tennis racket		9.34	**15.07**	13.25 (0.22)	14.56 (0.24)	9.08	19.03	15.65 (0.26)	**19.70 (0.33)**	9.45	19.01	16.62 (0.28)	**20.52 (0.35)**	1.559
skis		6.88	10.21	9.76 (1.29)	**10.27 (1.36)**	8.56	**12.36**	12.31 (1.63)	11.63 (1.54)	9.10	**13.10**	12.44 (1.49)	12.14 (1.46)	1.493
baseball bat	✓	5.08	4.56	6.11 (0.36)	**7.13 (0.41)**	3.55	2.77	6.43 (0.37)	**8.64 (0.50)**	2.62	2.12	4.75 (0.36)	**7.23 (0.55)**	1.404
toaster		28.00	36.74	35.69 (6.44)	**39.16 (7.07)**	1.01	3.06	**8.88 (1.60)**	5.15 (0.93)	0.33	0.94	**2.96 (0.49)**	0.93 (0.16)	1.399
frisbee		32.46	35.80	41.28 (0.70)	**42.35 (0.72)**	57.37	58.82	**62.55 (1.06)**	58.25 (0.99)	54.30	56.11	**59.69 (1.10)**	57.53 (1.06)	1.305
skateboard		14.38	16.81	17.71 (0.43)	**18.49 (0.45)**	13.60	16.01	**20.09 (0.49)**	18.47 (0.45)	13.54	15.44	**19.24 (0.61)**	18.03 (0.57)	1.286
knife	✓	15.70	19.75	18.71 (1.16)	**19.82 (1.23)**	16.37	18.92	**20.56 (1.28)**	18.90 (1.18)	12.93	16.88	**17.49 (1.30)**	17.41 (1.30)	1.262
handbag		15.71	19.08	19.41 (1.83)	**19.77 (1.87)**	6.30	8.49	**10.07 (0.95)**	6.53 (0.62)	3.30	5.27	**5.51 (0.83)**	3.26 (0.49)	1.258
spoon		7.93	9.80	**9.97 (13.47)**	9.36 (12.65)	5.69	6.35	**7.98 (10.78)**	6.93 (9.36)	5.92	7.32	**7.48 (16.62)**	6.91 (15.36)	1.257
kite		26.68	32.19	32.27 (0.66)	**32.89 (0.68)**	29.56	34.80	**38.11 (0.78)**	36.01 (0.74)	29.64	35.22	**37.88 (0.81)**	37.24 (0.79)	1.233
chair		27.62	31.10	**33.90 (1.13)**	33.39 (1.11)	23.50	24.99	**27.97 (0.93)**	24.67 (0.82)	21.78	23.14	**24.71 (0.91)**	22.62 (0.83)	1.227
surfboard		30.95	36.97	**37.52 (0.62)**	36.31 (0.60)	38.63	44.63	**45.50 (0.76)**	42.74 (0.71)	37.32	**43.01**	42.41 (0.76)	40.48 (0.73)	1.212
cup		31.76	35.83	**38.26 (0.94)**	37.96 (0.93)	31.47	38.93	**41.61 (1.02)**	39.62 (0.97)	29.14	35.88	**37.17 (0.99)**	35.73 (0.95)	1.205
fork	✓	12.44	14.80	**14.96 (1.38)**	14.61 (1.34)	18.79	17.78	**19.74 (1.81)**	19.10 (1.76)	18.15	17.53	**18.92 (1.62)**	18.43 (1.58)	1.203
clock		37.27	**44.57**	44.44 (0.72)	42.09 (0.68)	40.86	53.49	**53.70 (0.87)**	49.61 (0.81)	37.34	**49.82**	49.00 (0.87)	45.48 (0.81)	1.192
sink		22.82	23.69	**27.15 (0.64)**	26.71 (0.63)	25.01	26.19	**31.04 (0.74)**	29.85 (0.71)	25.18	26.26	**30.80 (0.79)**	30.01 (0.77)	1.190
toilet		52.98	59.88	**60.10 (0.87)**	58.48 (0.85)	61.90	65.75	**67.28 (0.98)**	65.53 (0.95)	61.18	65.34	**66.39 (0.98)**	65.26 (0.96)	1.134
microwave		42.13	45.84	46.94 (0.85)	**47.78 (0.87)**	39.17	**42.92**	42.44 (0.77)	41.10 (0.75)	37.52	**41.22**	38.09 (0.82)	39.29 (0.84)	1.134
snowboard		24.13	**28.97**	27.18 (0.66)	27.27 (0.66)	26.97	**31.78**	30.24 (0.73)	28.56 (0.69)	29.08	**33.32**	31.02 (0.86)	30.32 (0.84)	1.130
vase		33.25	34.12	**37.50 (0.64)**	37.16 (0.64)	32.03	33.59	**40.67 (0.70)**	38.04 (0.65)	26.76	28.23	**33.58 (0.67)**	32.27 (0.65)	1.128
car		37.51	38.65	**42.10 (0.80)**	41.61 (0.79)	37.87	39.95	42.00 (0.80)	**42.11 (0.80)**	36.06	38.48	39.81 (0.79)	**40.64 (0.80)**	1.122
truck		43.23	**49.72**	46.92 (0.80)	47.27 (0.81)	40.34	**46.92**	43.60 (0.74)	44.93 (0.77)	36.17	**42.00**	37.93 (0.75)	40.09 (0.79)	1.093
potted plant		34.29	35.64	**37.49 (1.59)**	36.98 (1.56)	15.85	18.34	**21.68 (0.92)**	18.05 (0.76)	12.81	14.98	**16.45 (0.88)**	14.69 (0.79)	1.093
boat		36.44	39.45	39.39 (0.75)	**39.69 (0.76)**	40.78	43.91	**44.73 (0.86)**	43.84 (0.84)	38.73	41.24	**41.49 (0.87)**	40.90 (0.86)	1.089
train		59.04	63.83	**63.93 (0.86)**	62.83 (0.84)	64.34	67.45	**68.33 (0.92)**	66.24 (0.89)	63.04	**66.29**	66.08 (0.93)	65.04 (0.91)	1.083
wine glass		34.08	**36.61**	36.57 (0.86)	35.48 (0.84)	31.20	34.35	**35.08 (0.83)**	34.40 (0.81)	36.78	**40.27**	39.05 (0.88)	39.72 (0.89)	1.073
bed		50.36	53.64	**54.00 (0.90)**	53.34 (0.89)	42.71	48.34	**49.72 (0.83)**	47.30 (0.79)	39.94	**45.02**	44.82 (0.81)	43.67 (0.79)	1.072
bottle		33.75	35.92	**35.97 (1.07)**	35.53 (1.05)	33.70	**35.88**	35.63 (1.06)	34.71 (1.03)	33.56	**35.41**	34.61 (1.05)	34.50 (1.05)	1.066
carrot		46.58	**49.54**	48.48 (1.18)	49.47 (1.20)	44.30	44.47	**47.77 (1.16)**	46.37 (1.13)	40.30	39.08	**41.39 (1.20)**	41.09 (1.19)	1.062
remote		34.53	**38.07**	34.63 (0.62)	36.66 (0.66)	48.29	50.08	49.71 (0.90)	**50.66 (0.91)**	41.22	41.38	41.16 (0.91)	**42.33 (0.93)**	1.062
scissors		38.90	**42.32**	41.21 (1.18)	40.73 (1.17)	38.36	38.43	**41.59 (1.19)**	37.66 (1.08)	36.62	**40.56**	40.36 (1.15)	37.07 (1.06)	1.059
toothbrush		29.19	29.50	29.89 (1.28)	**30.82 (1.32)**	25.61	23.79	**31.13 (1.33)**	28.25 (1.21)	15.66	13.54	**20.31 (1.73)**	19.30 (1.65)	1.056
sandwich		45.43	**47.82**	45.84 (0.99)	47.33 (1.03)	38.38	40.43	41.15 (0.89)	**41.19 (0.89)**	34.02	36.69	36.15 (0.92)	**37.60 (0.96)**	1.042
bench		40.56	**42.52**	42.23 (0.94)	41.66 (0.92)	37.20	39.68	**41.42 (0.92)**	39.67 (0.88)	34.63	36.80	**37.83 (0.90)**	37.40 (0.89)	1.041
keyboard	✓	49.31	49.47	50.78 (0.80)	**51.11 (0.80)**	**56.35**	53.99	55.62 (0.87)	54.95 (0.86)	50.48	50.02	**52.27 (0.89)**	52.11 (0.89)	1.037
backpack		26.79	**28.58**	27.55 (1.92)	27.33 (1.91)	13.17	**16.60**	16.39 (1.14)	14.50 (1.01)	13.01	**16.47**	16.17 (1.18)	13.58 (0.99)	1.028
horse		63.91	**66.30**	65.41 (0.87)	65.53 (0.87)	70.07	70.38	**70.52 (0.94)**	70.10 (0.93)	68.70	69.19	**69.86 (0.95)**	69.58 (0.94)	1.025
banana		65.63	**67.66**	66.79 (0.97)	67.13 (0.98)	65.65	67.55	**68.47 (1.00)**	67.62 (0.98)	62.79	63.83	**64.58 (0.98)**	64.46 (0.98)	1.023
bowl		28.16	**30.31**	28.49 (0.76)	28.77 (0.77)	21.27	**23.70**	22.86 (0.61)	22.32 (0.60)	18.18	**21.09**	19.29 (0.64)	19.69 (0.65)	1.022
zebra		73.64	**76.67**	74.95 (0.88)	75.05 (0.89)	78.41	**80.68**	80.46 (0.95)	80.65 (0.95)	78.46	**81.31**	80.57 (0.95)	80.84 (0.95)	1.019
person		57.62	57.74	58.63 (0.75)	**58.66 (0.75)**	64.89	66.00	67.46 (0.86)	**67.51 (0.86)**	64.35	65.35	66.63 (0.86)	**66.76 (0.86)**	1.018
airplane		54.15	**55.99**	55.02 (0.74)	53.94 (0.73)	57.66	60.18	**60.20 (0.81)**	57.44 (0.78)	56.77	59.19	**59.51 (0.81)**	56.60 (0.77)	1.016
couch		45.43	**48.04**	46.15 (0.97)	44.98 (0.94)	34.98	**38.15**	38.02 (0.80)	35.36 (0.74)	31.50	**34.83**	33.77 (0.76)	32.22 (0.73)	1.016
cell phone		42.66	**44.97**	42.44 (0.74)	43.30 (0.76)	53.05	53.04	**54.76 (0.96)**	53.82 (0.94)	53.21	**53.48**	53.41 (0.92)	52.81 (0.91)	1.015
broccoli		49.47	49.70	**50.21 (0.90)**	49.82 (0.89)	**53.80**	53.22	53.79 (0.97)	53.28 (0.96)	54.29	53.56	**54.57 (1.02)**	53.99 (1.01)	1.015
background		79.59	80.25	**80.71 (0.91)**	80.63 (0.91)	82.74	83.43	**83.88 (0.95)**	83.77 (0.94)	82.34	83.03	83.28 (0.95)	**83.33 (0.95)**	1.014
hot dog		55.66	55.14	56.37 (0.99)	**56.44 (0.99)**	51.30	51.91	**54.68 (0.96)**	52.28 (0.92)	45.05	45.61	**46.72 (1.00)**	45.79 (0.98)	1.014
orange		62.51	63.10	62.40 (0.93)	**63.34 (0.95)**	63.42	64.55	65.22 (0.97)	**65.32 (0.98)**	62.53	62.74	62.28 (0.97)	**63.22 (0.99)**	1.013
book		38.58	**39.70**	38.82 (1.16)	38.05 (1.13)	32.33	33.16	**33.89 (1.01)**	31.64 (0.94)	30.36	**31.16**	30.52 (1.07)	29.84 (1.04)	1.006
traffic light		26.59	**27.38**	26.74 (0.53)	26.51 (0.52)	46.59	46.46	**49.67 (0.98)**	48.07 (0.95)	46.88	47.01	**48.71 (0.96)**	48.45 (0.96)	1.006
apple		51.23	**53.07**	50.91 (0.99)	51.48 (1.00)	48.39	**48.85**	48.00 (0.93)	46.86 (0.91)	51.10	**51.25**	48.25 (0.92)	49.02 (0.93)	1.005
parking meter		62.68	**64.66**	62.16 (0.85)	62.98 (0.86)	67.90	69.92	**72.14 (0.99)**	68.97 (0.94)	67.45	68.89	**69.38 (1.06)**	69.01 (1.06)	1.005
cake		50.07	**50.85**	49.57 (0.91)	50.24 (0.93)	48.35	48.87	**51.74 (0.95)**	49.29 (0.91)	46.26	46.86	**48.24 (0.97)**	46.79 (0.94)	1.003
dog		69.08	**69.86**	69.11 (0.92)	68.08 (0.91)	71.53	71.89	**72.02 (0.96)**	70.66 (0.94)	69.30	**70.09**	69.44 (0.97)	69.02 (0.97)	1.000
bus		**65.25**	64.07	63.57 (0.82)	64.82 (0.83)	68.58	69.67	69.38 (0.89)	**70.75 (0.91)**	64.69	65.63	63.85 (0.86)	**66.87 (0.90)**	0.993
teddy bear		**60.63**	60.11	60.17 (0.88)	59.51 (0.87)	61.07	61.39	**63.08 (0.93)**	61.56 (0.90)	61.02	61.80	**62.28 (0.94)**	62.27 (0.94)	0.992
cow		**68.31**	67.33	67.72 (0.86)	66.79 (0.85)	72.84	73.07	**74.73 (0.95)**	74.05 (0.94)	69.04	69.75	70.40 (0.94)	**70.41 (0.94)**	0.991
tv		48.26	**49.30**	47.70 (0.79)	47.83 (0.80)	**50.49**	48.40	48.71 (0.81)	48.28 (0.80)	**48.37**	46.72	46.28 (0.83)	46.70 (0.84)	0.991
donut		57.45	**57.55**	56.05 (0.95)	56.88 (0.96)	54.37	55.19	**57.78 (0.98)**	55.36 (0.94)	56.78	58.57	**59.27 (0.99)**	59.13 (0.98)	0.990
bird		58.73	**59.20**	57.38 (0.81)	57.98 (0.82)	67.03	**68.45**	67.87 (0.96)	**68.45 (0.97)**	62.57	**64.31**	63.60 (0.99)	64.29 (1.00)	0.987
bicycle		**52.18**	51.52	51.46 (0.99)	51.49 (0.99)	50.00	**51.37**	51.36 (0.99)	50.97 (0.98)	48.20	**49.32**	49.21 (0.98)	49.30 (0.98)	0.987
motorcycle		**65.32**	64.57	64.41 (0.89)	64.20 (0.89)	65.46	65.23	**66.66 (0.92)**	65.54 (0.90)	64.96	64.88	**65.83 (0.93)**	65.30 (0.92)	0.986
umbrella		63.66	**64.83**	62.76 (0.94)	62.10 (0.93)	62.26	62.93	**64.34 (0.97)**	62.07 (0.93)	63.51	**64.90**	64.28 (0.94)	63.90 (0.94)	0.986
elephant		**79.01**	78.22	77.51 (0.91)	76.91 (0.91)	81.56	81.12	**81.93 (0.97)**	81.23 (0.96)	**81.04**	80.38	80.91 (0.96)	80.51 (0.96)	0.981
suitcase		**52.62**	52.54	51.62 (0.90)	51.13 (0.89)	50.38	50.96	**52.19 (0.91)**	50.40 (0.88)	45.68	**47.24**	46.44 (0.89)	46.46 (0.89)	0.981
giraffe		69.79	**70.05**	68.30 (0.85)	68.25 (0.85)	74.81	**75.89**	75.17 (0.94)	74.69 (0.93)	74.71	**75.79**	75.48 (0.95)	75.12 (0.94)	0.979
bear		**80.39**	76.42	78.61 (0.94)	77.92 (0.93)	81.93	81.57	**82.59 (0.99)**	81.89 (0.98)	79.52	80.04	**80.46 (1.00)**	80.32 (1.00)	0.978
fire hydrant		**70.69**	69.98	68.61 (0.88)	68.80 (0.88)	74.05	75.03	**76.40 (0.98)**	74.99 (0.96)	74.44	73.83	**74.97 (0.99)**	74.81 (0.99)	0.973
stop sign		**73.11**	72.51	70.74 (0.89)	70.27 (0.88)	76.28	77.91	**78.42 (0.98)**	77.52 (0.97)	78.53	**79.27**	78.32 (0.98)	78.33 (0.98)	0.968
sheep		**65.41**	65.29	62.21 (0.82)	61.69 (0.81)	**71.63**	71.38	71.41 (0.94)	70.33 (0.93)	**71.25**	71.17	70.29 (0.94)	70.33 (0.94)	0.951
refrigerator		45.72	**46.80**	43.38 (0.71)	43.00 (0.70)	**46.33**	46.16	43.34 (0.71)	39.58 (0.65)	**43.19**	42.76	37.88 (0.65)	35.81 (0.62)	0.949
pizza		**64.95**	59.13	60.98 (0.85)	61.62 (0.86)	**65.89**	62.07	63.37 (0.89)	63.47 (0.89)	**64.76**	60.99	62.09 (0.89)	62.29 (0.89)	0.949
cat		**71.14**	64.46	65.31 (0.85)	66.64 (0.87)	**75.23**	71.67	72.42 (0.95)	74.28 (0.97)	**74.77**	71.71	71.83 (0.96)	73.55 (0.98)	0.937
sports ball		**12.17**	11.25	9.71 (0.28)	11.38 (0.33)	**35.21**	33.85	31.64 (0.92)	33.84 (0.98)	30.41	**30.57**	24.01 (0.84)	28.68 (1.00)	0.935
oven		**37.92**	35.72	34.58 (0.72)	34.01 (0.71)	**36.62**	34.32	34.18 (0.71)	30.02 (0.62)	**34.60**	33.14	31.00 (0.72)	29.49 (0.68)	0.912
laptop		**48.48**	41.59	42.88 (0.68)	42.60 (0.68)	**50.44**	37.43	43.39 (0.69)	41.82 (0.66)	**49.21**	36.94	41.42 (0.69)	40.45 (0.68)	0.884
dining table		**30.05**	27.81	24.46 (0.59)	24.15 (0.58)	**20.44**	19.25	13.83 (0.33)	13.13 (0.32)	**20.03**	18.45	12.40 (0.32)	12.28 (0.32)	0.814
hair drier		29.02	**29.73**	22.32 (∞)	22.01 (∞)	0.92	0.00	**12.35 (∞)**	0.00 (∞)	0.48	0.09	**4.18 (∞)**	0.00 (∞)	0.769
mIoU		43.14	**44.69**	44.46 (0.85)	44.49 (0.85)	43.64	45.12	**46.82 (0.89)**	45.31 (0.87)	42.01	43.51	**44.07 (0.90)**	43.59 (0.89)	1.031

Table 4.4: The performance for each class of our final models on **COCO**. The IoU performance is reported for each class in the dataset, for the training set's pseudo labels, and the training and validation sets' segmentation masks (these form the three major column groupings). The *reimplemented* scores for MCTFormer+, with and without CCT, make up the first two columns in each group, respectively. The final two columns present the scores for our final model, where cross attention is applied to the layer groups 3, 5, 6, and 6, 7, 8, respectively. **For our models' results, the ratio to the performance of the segmentation model trained on the ground truth masks is shown ("∞" refers to segmentation scores of 0 when training with ground truth supervision).** The checkmark in the second column indicates the classes we have generally considered highly co-occurring. The results are ordered by the final column (the Improvement Factor), which measures the ratio of performance for the pseudo labels of our models (the maximum of both models) to MCTFormer+ (with CCT), and a line separates the classes we improved on from the ones we did not. For each of the major groupings, the highest score per class is shown in bold.

be biased to more common classes as opposed to the rarely-occurring ones. This could be confirmed by assigning probabilistic weights to each image based on their classes in order to make the random choice of a class more uniform.

We have identified two possible reasons for this behaviour and have suggested possible experiments to accept or reject them, but it is clear more work is required to identify the precise cause and rectify it.

One major concern is the collapse of segmentation performance (for both training and validation sets) of the Toaster and Hair Drier classes, for COCO, compared to the pseudo label performance. Figure 4.12 provides examples for the Hair Drier class. The pseudo labels for both our models clearly show the hair drier being identified and correctly located with minimal erroneous pixel activations. However, the segmentation masks for the segmentation model trained with each set of pseudo labels clearly show a complete lack of identification of the hair drier. Notably, these classes obtain similar segmentation mask scores even when the segmentation model is trained using the ground truth segmentation masks — these results are presented in Table A.2. This suggests that the reason for the collapse is not due to our models producing bad-quality pseudo labels, but rather because the segmentation model is not effectively learning to recognise these classes, perhaps because the DeepLab-V1 model has reached its learning capacity. We leave an analysis of more modern segmentation models to future work, as the collapse in segmentation mask quality for these classes is clearly due to the segmentation model itself, and not our pseudo label quality.

Figure 4.13 and Figure 4.14 present examples of pseudo labels generated using our method for VOC and COCO, respectively, with Figure 4.15 presenting examples of our method failing.

Figure 4.13 initially presents an example of a general, non-co-occurring class ("tvmonitor") in row (a), largely maintaining the performance of MCTFormer+ — this is a more widely observed trend for non-co-occurring classes. Even though the first model (3, 5, 6 layer model) increased the activated region both above and below (spatially) MCTFormer+'s activated region, the second model (6, 7, 8 layer model) was able to constrain the activation below MCTFormer+'s more accurately to the object. For the row (b) (for the "chair" and "dining table" co-occurring pair), we are able to extensively constrain the over-activation and misclassification of the "chair" in particular, to more finely correspond to the original image. Row (c) shows that we are able to achieve better coverage for instances including this co-occurring pair, with further "chair" misclassification being prevented. Row (d)'s example, whilst introducing extensive over-activations for chairs above the table, is in fact able to identify that chairs exist above the table, which is something that MCTFormer+ was not able to do. The example in row (e) only prevents the erroneous activation for the "chair" towards the bottom of the image; it maintains similar performance to MCTFormer+ for the over-activation above the table. Overall, we still struggle to finely segment intricate structures, like the backs of the chairs, which are not single and solid areas.

Figure 4.14 presents examples of successfully-improved pseudo labels for COCO. The general images for rows (a) and (b) show that on normal, non-co-occurring classes

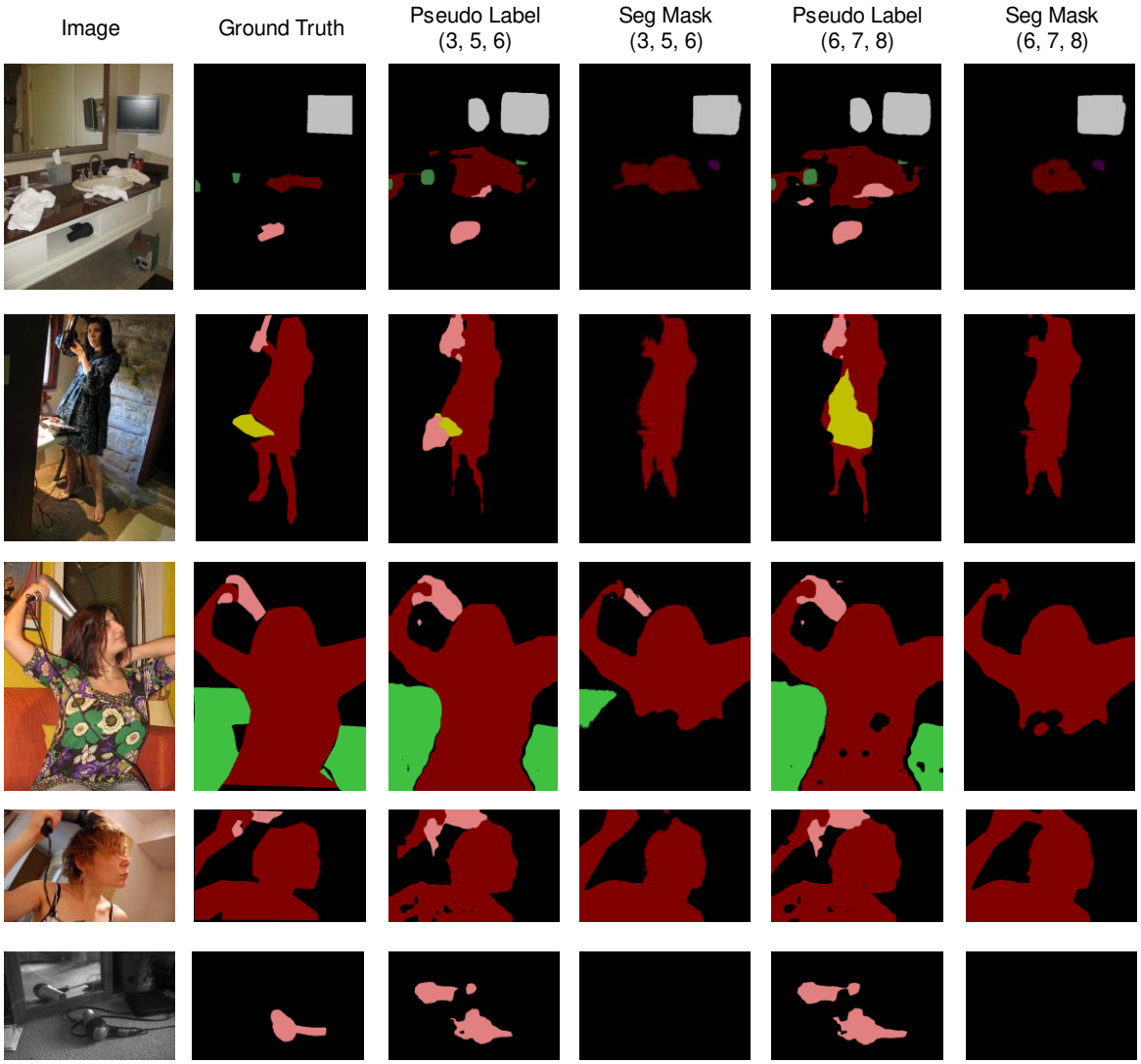


Figure 4.12: Examples of pseudo labels and segmentation masks including a Hair Drier (pink), from COCO. The first two columns are the original image and ground truth segmentation mask, respectively. The last four columns are arranged in two groups of two, where the first column in each group is the Pseudo Label predicted by one of our models and the second column is the Segmentation Mask predicted by the segmentation model trained with this model's pseudo labels. The first group is the model using cross attention on layers 3, 5, and 6, and the second group is based on the model using layers 6, 7, and 8.

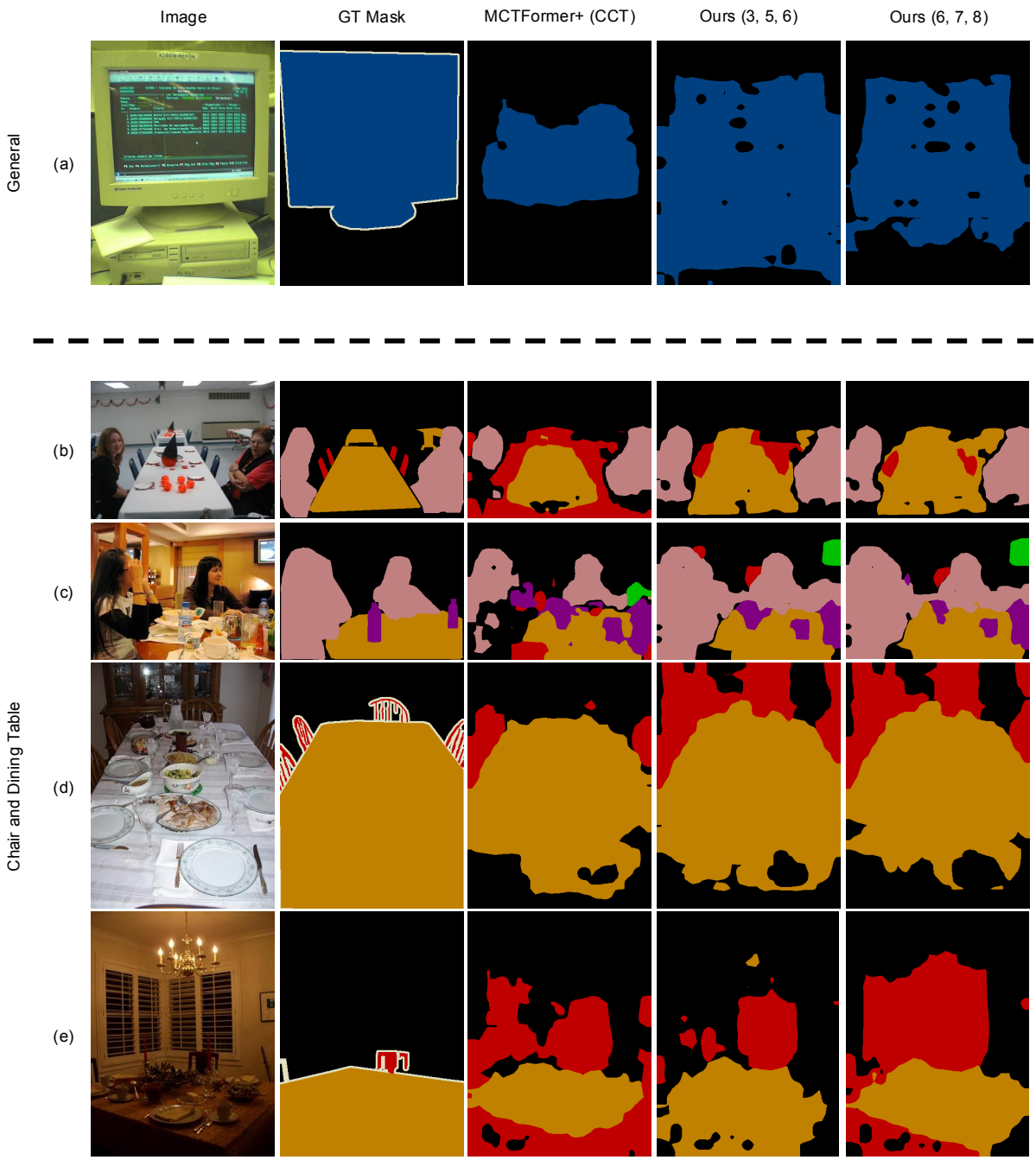


Figure 4.13: Visualisation of pseudo labels generated for **VOC** images, by our two best dual-branch models (using cross image attention for layers 3, 5, 6, and 6, 7, 8, compared to the ground-truth segmentation masks and the pseudo labels generated by MCTFormer+. An example of a general class is shown ("tvmonitor") and several examples for the co-occurring class pair of "chair" and "dining table" are shown.

("bear" and "elephant"), we largely maintain the performance of MCTFormer+ — we do not improve the coverage of these objects, but crucially, we do not degrade them either. The coverage is maintained and we do not introduce erroneous activations. Similarly, we show in the "fork" example in row (c) that our performance is largely similar to the MCTFormer+ baseline. Despite Table 4.4 showing that our performance improved for this class, the general nature of objects of this class is to be small-area objects, meaning that the majority of the improvement will have come from fine boundary corrections. We are not able to segment the intricate structures of the fork prongs present in the ground-truth mask, but we are able to estimate the object coverage more effectively. The last two groups of images are for the highly co-occurring class pairs of "baseball bat" & "baseball glove" (rows (d) to (f)) and "mouse" & "keyboard" (rows (g) to (i)). In both sets of examples, we are clearly able to substantially reduce erroneous activations and over-activations, constraining the regions of identified pixels for these classes far more accurately to their object boundaries. We also prevent the misclassification of similar looking, but inaccurate background regions. This is a direct success of using cross-image context in our dual-branch framework. Ironically, for the example in row (e), the "diagonal bar" actually looks like a baseball bat being held by a person (bottom left corner has what appears to be a hand) — both our models and MCTFormer+ are largely able to detect this, even though the ground truth mask does not.

Figure 4.15 presents a number of examples from VOC and COCO where our efforts to improve co-occurrence class segmentation quality through CIA-AP have been largely over exaggerated. For COCO, the example in row (a) shows that while we are better able to detect the location of the baseball bat compared to MCTFormer+ (due to the improved attention), we do so at the cost of reduced coverage of the person. In contrast, the VOC example in row (d) shows that the table has been localised almost entirely incorrectly, with erroneous activations and misclassification for the candles drawing attention away from increased coverage of the table itself. Whilst the attention refinement generally helps, it adversely affects the second and third COCO examples (rows (b) and (c)) and the last VOC example (row (e)). In these cases, in its desire to promote the co-occurring classes, our models have over-estimated the extent of these objects and induces erroneous activations. A general pattern is that when the co-occurring object in an image is quite small, the models tend to overcompensate and classify many more pixels as belonging to this class than is necessary, almost as if the model knows the object should be present, but isn't confident about its location, and so tries to 'place' it in multiple potential locations.

4.5.5 Comparison to State-of-the-Art

Table 4.5 compares the performance of our two best dual-branch models (using layers 3, 5, 6, and 6, 7, 8, respectively) to other high performing methods. They are split into two broad categories: single and multi-staged methods. Single-stage methods directly predict the final segmentation mask, often predicting an intermediate pseudo label along the way to help with pixel-level supervision. Multi-stage models use two separate models, one for predicting the pseudo label (and this is where the major con-

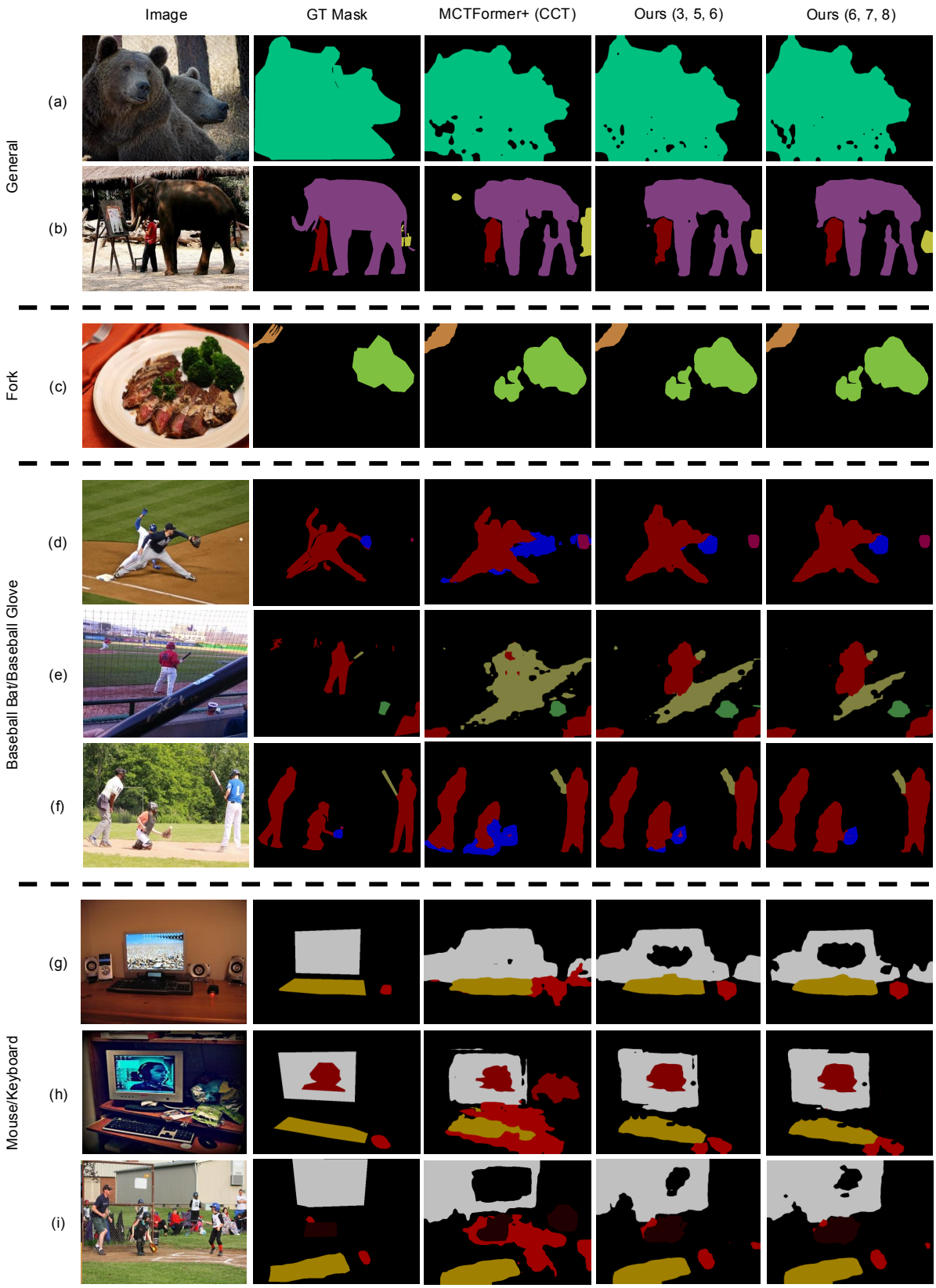


Figure 4.14: Visualisation of pseudo labels generated for **COCO** images, by our two best dual-branch models (using cross image attention for layers 3, 5, 6, and 6, 7, 8, compared to the ground-truth segmentation masks and the pseudo labels generated by MCTFormer+. Two examples of general classes ("bear" and "elephant") are shown, a single example of the co-occurring class "fork", and three examples each for the co-occurring class pairs "baseball bat" & "baseball glove", and "mouse" & "keyboard".

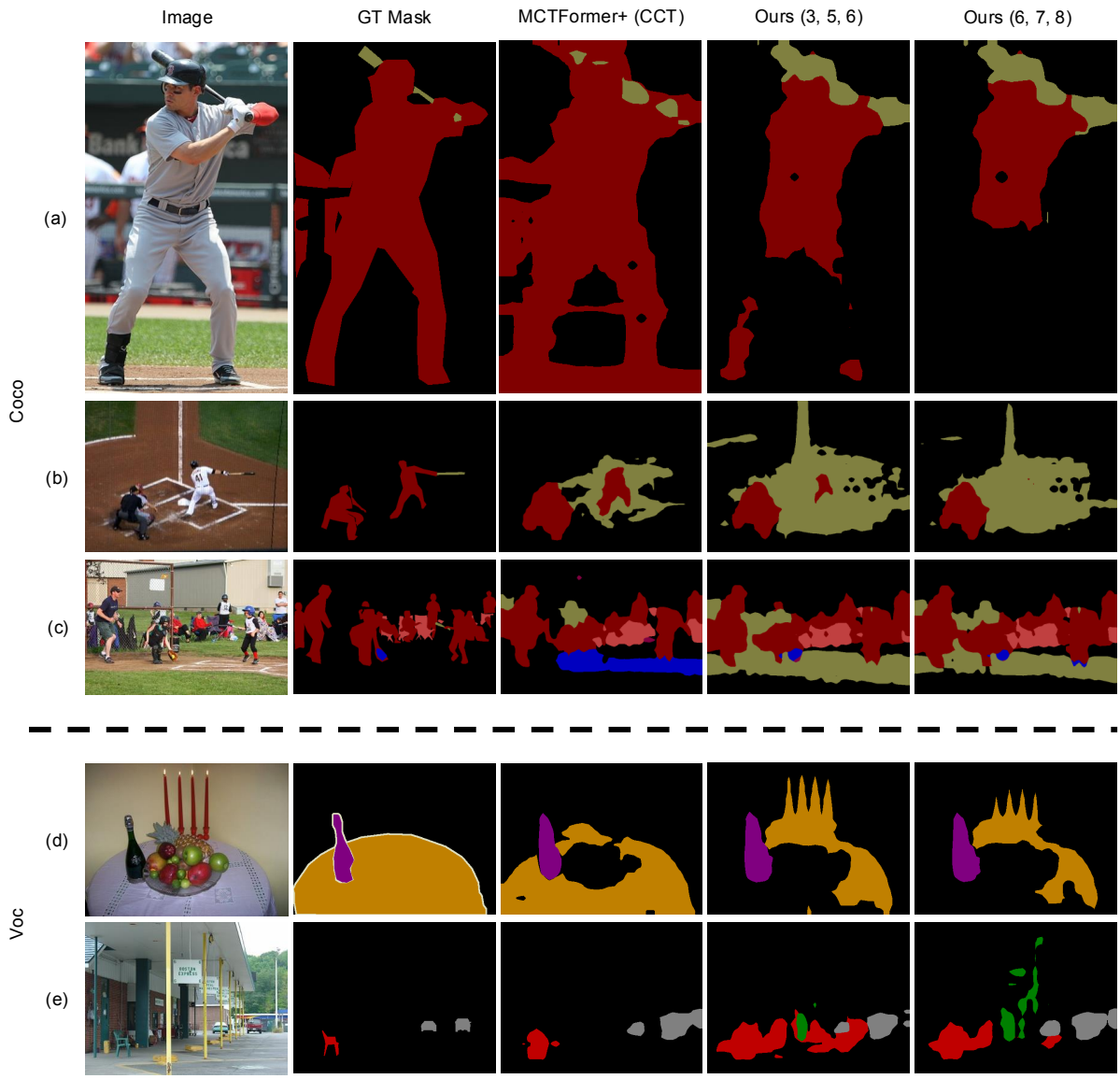


Figure 4.15: Visualisation of pseudo label **failure cases** generated for VOC and COCO images, by our two best dual-branch models (using cross image attention for layers 3, 5, 6, and 6, 7, 8), compared to the ground-truth segmentation masks and the pseudo labels generated by MCTFormer+. The first three images are from COCO and primarily correspond to the "baseball bat" and "baseball glove" co-occurring class pair. The last two images are from VOC and primarily correspond to the "chair" and "dining table" co-occurring class pair.

tributions are to WSSS for multi-stage models), and the second is generally an off-the-shelf segmentation model which is trained using the pseudo label as the ground truth. Our method is a multi-stage method.

The prior work in Table 4.5 only report comparison scores for training set pseudo labels on VOC, and mainly report the final segmentation scores for the validation sets. The works often use the same segmentation backbone when using VOC and COCO, however, sometimes they do change the backbone — this is indicated by the backbone being reported for both datasets.

MCTFormer+ has the best validation score on VOC, of which we fall 4.3% short. However, our reimplementation of MCTFormer+ was not able to match this performance and our best model does match the reimplemented score. We cannot draw direct comparison to SeCo [Yang *et al.* 2024], which is the current state-of-the-art across the board, since that is a single-stage model and won't be susceptible to the same flaws MCTFormer+ and our dual-branch models are that lead to the 4.3% discrepancy to the reported scores (i.e. in the segmentation model). In terms of the VOC pseudo label quality, we underperform all the other methods by up to 10.2% when comparing to SeCo, which is concerning. However, it is unclear whether the segmentation model is able to effectively recover the discrepancy, since it is only able to match the reimplemented MCTFormer+ scores, which itself suggests that we retain the competitiveness of MCTFormer+. Seeing as COCO is where we put most of our focus, the suggestion that we remain competitive on VOC is sufficient.

For COCO, our validation mask scores are competitive to the other methods, falling only 2.6% short of the current state-of-the-art, SeCo [Yang *et al.* 2024]. Interestingly, we do outperform (by 2.1%) the reimplemented MCTFormer+. Seeing as we were only able to match the reimplemented performance on VOC, this suggests that our approach is more effective on COCO — this aligns with our view that COCO is substantially harder than VOC in terms of class co-occurrence, and that our approach directly targets this co-occurrence. However, we cannot draw comparisons to our co-occurring classes of interest, which are our main focus, since the other works do not report class scores for COCO, only the final mIoU scores. Since this is where we saw the most improvement, we cannot draw an effective comparison using overall mIoU data.

4.6 Summary

Section 4.3 provides an in-depth analysis of MCTFormer+ [Xu *et al.* 2024], a high-performing transformer-based model that employs multiple class tokens and a novel approach to refining transformer attention maps. Our reimplementation confirmed their findings that contrastive learning (CCT) improves performance on PASCAL VOC 2012 but not on MS COCO 2014.

Our final model builds upon MCTFormer+ [Xu *et al.* 2024] by integrating our Cross Image Attention from the Auxiliary to the Primary branch (CIA-AP) and a dual-branch contrastive learning scheme involving Contrastive Class Tokens (CCT) and CCT Prime. The overall architecture consists of two parallel transformer encoder branches: the Pri-

Methods	Single/Multi Stage	Supervision	VOC			COCO	
			Backbone	Pseudo Label	Val Seg Mask	Backbone	Val Seg Mask
1Stage [Araslanov and Roth 2020]	$\mathcal{S}$	$\mathcal{I}$	ResNet38	66.9	62.7	—	—
AFA [Ru *et al.* 2022]	$\mathcal{S}$	$\mathcal{I}$	MiT-B	68.7	66.0	MiT-B	38.9
ViT-PCM [Rossetti *et al.* 2022]	$\mathcal{S}$	$\mathcal{I}$	ViT-B	71.4	70.3	—	—
ToCo [Ru *et al.* 2023]	$\mathcal{S}$	$\mathcal{I}$	ViT-B	73.6	71.1	ViT-B	42.3
SeCo [Yang *et al.* 2024]	$\mathcal{S}$	$\mathcal{I}$	ViT-B	**76.5**	**74.0**	ViT-B	**46.7**
EPS [Lee *et al.* 2021b]	$\mathcal{M}$	$\mathcal{I}+\mathcal{SM}$	ResNet101	71.6	71.0	VGG16	35.7
RCA [Zhou *et al.* 2022]	$\mathcal{M}$	$\mathcal{I}+\mathcal{SM}$	ResNet38	74.1	72.2	VGG16	36.8
L2G [Jiang *et al.* 2022]	$\mathcal{M}$	$\mathcal{I}+\mathcal{SM}$	ResNet38	71.9	72.1	ResNet101	44.2
CDA [Su *et al.* 2021]	$\mathcal{M}$	$\mathcal{I}+\mathcal{E}$	ResNet38	66.4	66.1	ResNet38	33.2
CLIP-ES [Lin *et al.* 2023]	$\mathcal{M}$	$\mathcal{I}+\mathcal{E}$	ResNet101	75.0	73.8	ResNet101	45.4
AdvCAM [Lee *et al.* 2021a]	$\mathcal{M}$	$\mathcal{I}$	ResNet101	68.0	68.1	—	—
W-OoD [Lee *et al.* 2022]	$\mathcal{M}$	$\mathcal{I}$	ResNet38	72.1	70.7	—	—
CLIMS [Xie *et al.* 2022]	$\mathcal{M}$	$\mathcal{I}$	ResNet50	70.5	70.4	—	—
FPR [Chen *et al.* 2023b]	$\mathcal{M}$	$\mathcal{I}$	ResNet101	66.4	70.3	ResNet101	43.9
OCR [Cheng *et al.* 2023]	$\mathcal{M}$	$\mathcal{I}$	ResNet38	69.1	72.7	ResNet38	42.5
MCTFormer [Xu *et al.* 2022]				61.7	71.9		42.0
MCTFormer+ [Xu *et al.* 2024]	$\mathcal{M}$	$\mathcal{I}$	ResNet38	68.8	**74.0**	ResNet38	45.2
MCTFormer+*				67.0	69.7		42.0
Ours (3, 5, 6)	$\mathcal{M}$	$\mathcal{I}$	ResNet38	66.2	69.5	ResNet38	44.1
Ours (6, 7, 8)				66.3	69.7		43.6

Table 4.5: Comparisons with SOTAs in mIoU (%). The pseudo labels are reported for the training VOC set only. Validation segmentation masks are reported for both VOC and COCO. "—" denotes no reported score. "*" refers to our reimplemented version of the method. $\mathcal{S}$: single-staged, $\mathcal{M}$: multi-staged, $\mathcal{I}$ image labels, $\mathcal{SM}$: saliency maps, $\mathcal{E}$: external data.

mary Branch (processing the image being segmented) and the Auxiliary Branch (providing additional context). These branches are Siamese networks, sharing the same architecture and weights.

Cross Image Attention (CIA-AP): Our Cross Image Attention mechanism aims to address the class co-occurrence problem by enhancing the attention maps corresponding to co-occurring classes. Specifically, the query vectors of the Primary Branch are augmented with those from the Auxiliary Branch by averaging them, which allows for the introduction of high-quality query vectors from the auxiliary image. To ensure these vectors are pure and unambiguous, the auxiliary image is chosen such that it contains only one of the co-occurring classes present in the primary image. This selective pairing allows the auxiliary image to provide a clean, high-quality, and unambiguous query vector that enhances the representation of the relevant co-occurring class in the primary image. This improvement in attention quality allows for effective enhancement of attention maps corresponding to challenging co-occurring classes.

CCT and CCT Prime: The CCT module, inherited from MCTFormer+, aims to enhance class separation by encouraging the class tokens' similarity matrix to approximate an identity matrix. This encourages tokens for different classes to occupy distinct regions of the representation space. However, CCT only promotes separation within each branch independently, which can be insufficient for effective cross-image attention.

To address this limitation, we introduce CCT Prime, a contrastive learning module designed to align the representations of the Primary and Auxiliary branches. While CCT

encourages separation within each branch, CCT Prime focuses on promoting compatibility between the representations of the two branches, enabling more effective information exchange during cross attention. This alignment is particularly crucial for resolving class co-occurrence, ensuring that the auxiliary image provides meaningful guidance to the primary image.

Pipeline Overview: The model pipeline consists of several components working together to achieve improved weakly supervised segmentation.

1. First, two images are passed through the Primary and Auxiliary branches of the transformer encoder. These branches are identical and share weights, forming a dual-branch architecture.
2. Cross Image Attention (CIA-AP) is applied from the auxiliary to the primary branch by averaging the query vectors. This operation enhances the attention maps corresponding to co-occurring classes by introducing pure query vectors from the auxiliary image, which is selected to contain only one of the co-occurring classes in the primary image.
3. CCT is applied independently to each branch to promote intra-branch separation of class tokens. However, since this does not guarantee alignment between branches, CCT Prime is introduced to promote compatibility between the Primary and Auxiliary branches, allowing for more effective information exchange during cross attention.
4. The class tokens are then passed through a classification module to obtain image-level classification scores. Simultaneously, patch tokens are processed through the PatchCAM module, which refines class-specific localisation maps via Patch Affinity. This is performed for both the primary and auxiliary images.
5. During training, CIA-AP utilises auxiliary images to enhance processing of the primary image. During inference, only the Primary Branch is used, with no cross attention performed.

This framework effectively addresses the detrimental impact of class co-occurrence on weakly supervised segmentation by combining cross-image attention with contrastive learning. By selectively enhancing attention for the targeted classes, our approach successfully improved performance for challenging co-occurring pairs such as Baseball Bat and Baseball Glove, Mouse and Keyboard, and Knife and Fork, while roughly maintaining overall mIoU performance.

Despite these improvements, our method does not outperform the current state-of-the-art but remains competitive while effectively mitigating the co-occurrence issue.

Chapter 5

Cross Image Attention Ablation

This chapter presents an extensive series of ablation studies aimed at thoroughly examining the impact of various design choices related to Cross Image Attention, and the ultimate formulation of Cross Image Attention from the Auxiliary to the Primary branch (CIA-AP) that is presented as part of the final model in Chapter 4. The analysis focuses on understanding how different configurations contribute to addressing the class co-occurrence problem and improving segmentation performance. Given that the COCO dataset exhibits more severe co-occurrence issues compared to VOC, most of the analysis is centred around the worst-performing class pairs within COCO.

In particular, we investigate several key components of our model: the Cross Image Attention mechanism, including different layer groupings, augmentation methods, and patch attention configurations. The decisions made for these configurations are informed by the findings of these ablation studies, which provide deeper insight into the contributions of each component to the overall performance.

Specifically, Section 5.1 argues for the use of Siamese networks for our dual-branch framework, where each branch is identical in architecture and shares weights. Section 5.2 broadly presents the iterative development that results in the final CIA-AP cross attention scheme, with Sections 5.2.1 to 5.2.4 presenting the conventional query-swap cross attention scheme and subsequent refinements. Section 5.2.5 critically analyses the impact of the conventional cross attention schemes and finds them to be inadequate for addressing the co-occurrence problem, although they are able to exchange information between the primary and auxiliary branches.

Building on this, Section 5.2.6 presents a new, principled approach to applying cross image attention, namely the Cross Image Attention from the Auxiliary to Primary branch (CIA-AP) that we use in our final model. As part of CIA-AP, Section 5.2.7 examines the effects of various augmentation methods for combining the query matrices from the primary and auxiliary branches, before Section 5.2.8 investigates the impact of cross attention being applied to different groups of layers.

Finally, Section 5.3 presents mathematical motivation for our decision not to use cross attention (or the auxiliary branch) during inference.

5.1 Siamese Weights

Section 4.5 presented our final dual-branch framework, which briefly entails passing two images, one primary and one auxiliary, through each branch respectively, and performing cross image attention between these two branches during the forward pass. These two branches are architecturally identical and are chosen to share their weights. Additionally, the corresponding classification and PatchCAM modules are also chosen to share their weights.

This was a decision made early in the framework's development and so was performed on the VOC dataset only. Figure 5.1 compares the overall mIoU performance of the pseudo labels for all combinations of the following naive training and inference mining strategies (i.e. the strategy with which the auxiliary image is chosen):

1. Training Mining Strategies:

   (a) Random: the auxiliary image is randomly chosen from the entire training image set, with the sole provision that the primary image is not sampled. The primary and auxiliary images may have any degree of overlap for the classes present. Random mining is included as a default baseline.

   (b) Exactly 1 Overlap: the auxiliary image is randomly chosen from the entire training image set, such that it has exactly 1 class in common with the primary image, and that it is not the primary image. The primary and auxiliary images may have any other classes present, provided there is no further overlap. The intention with this strategy is to promote disambiguation of class pairs — by having an auxiliary image with only one of a pair of classes, the model is exposed to the appearance of only one of the classes, thereby being able to tell the classes apart. Note that there is no co-occurrence-based restriction on what class pairs are allowed to overlap — this means that overlap could occur on a non-co-occurring class, but this is an initial naive method which is later refined.

2. Inference Mining Strategies:

   (a) Same: the primary image is also used as the auxiliary image. Each image will have a distinct set of tokens.

   (b) Exactly 0 Overlap: the auxiliary image is randomly chosen such that it has no classes in common with the primary image. The primary and auxiliary images may have any other classes present, provided they are not present in both images.

   (c) Exactly 1 Overlap: same as the corresponding training mining strategy.

   (d) At Least 1 Overlap: the auxiliary image is randomly chosen such that it has at least 1 class in common with the primary image, and that it is not the primary image. The primary and auxiliary images may have any other classes present.

The complete description of these mining strategies is not relevant to this sub-section. However, they are used in the model configurations for this section, and are pertinent later on in this ablation.

Figure 5.1 shows that the cases where the two branches share weights outperforms the corresponding configuration where the two branches have their own set of weights. Consequently, we decided to exclusively use shared weights for the transformer branches and their attached classification and PatchCAM modules.

Even though Figure 5.1 justifies the use of shared weights for the VOC dataset, we do present evidence that this is still valid for use with the COCO dataset: Figure 5.2 presents the overall training and validation losses for our final two models, over the complete training period of 45 epochs. The training curves both continue to decrease until the end of training, suggesting that the model with shared weights has capacity to learn over the entirety of its training. The validation curves do not increase after the initial spike, which reinforces that our model is not overfitting, but rather learning meaningfully.

Consequently, the use of shared weights for the two branches' transformers and attached classification and PatchCAM modules is preferable to the use of separate weights. Initial experiments on VOC showed that shared weights outperform the counterpart model configuration for separate weights, and later experiments showed that even with the harder COCO dataset, the dual-branch models are still able to meaningfully learn.

Additionally, the intuitive appeal of using shared weights lies in their ability to enhance consistency between the branches. Since our overarching goal is to provide more stable and reliable query vectors to the primary image, it is crucial that both branches generate queries in a consistent manner. By sharing weights, we ensure that the query vectors produced by the auxiliary branch are constructed using the same underlying feature extraction process as those in the primary branch. This alignment is particularly beneficial when transferring information via cross attention, as it promotes compatibility between the query vectors from both branches, thereby enhancing the overall effectiveness of the information exchange.

5.2 Evolution of Cross Image Attention and Auxiliary Image Mining

The core part of this work is determining the correct form for applying Cross Image Attention to effectively share information between the two branches. A parallel problem is to choose the most effective auxiliary image, such that the "correct" information is shared between the branches. As decisions are made about the implementation of the cross image attention, the auxiliary image has to be mined in a complementary way. Consequently, these two conceptual streams develop together in an interlinked manner, and so will be presented together.

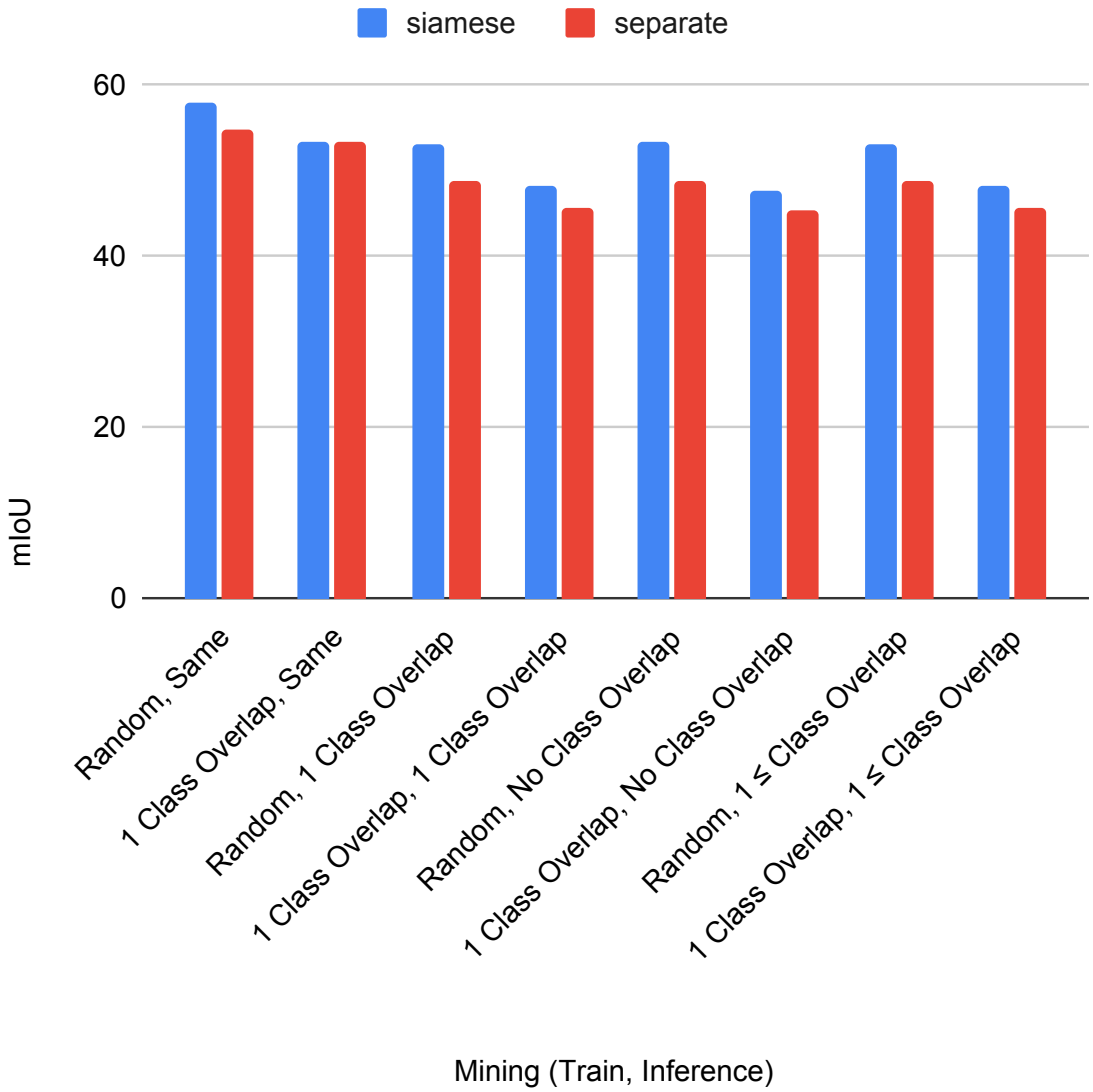


Figure 5.1: Comparison of overall pseudo label quality (mIoU) for different training mining strategies and different inference mining strategies, for VOC. The x-axis is labelled as *"(train strategy, infer strategy)"*.

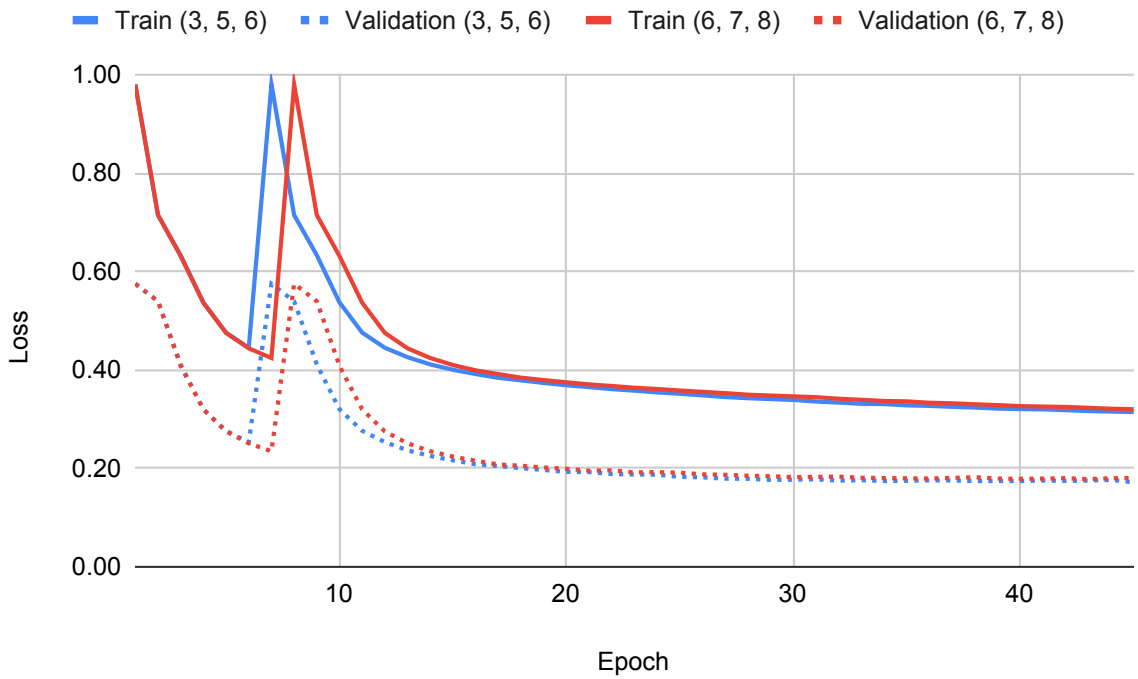


Figure 5.2: Graph of the overall train (solid) and validation (dashed) losses, for our final two models, for COCO. The normal COCO train set was further split into a random 80/20 training and validation split to conduct this test.

This section will initially investigate traditional methods of applying cross attention, mainly focusing on VOC, before taking a more principled approach, with COCO, to solving the class co-occurrence problem.

Additionally, although other cross attention methods exist [Chen *et al.* 2021; Lin *et al.* 2022], we choose to use the method of exchanging the Query matrices between self-attention modules due to its simple and intuitive interpretation for solving the class co-occurrence problem. This intuition was discussed in the final presentation of the model in Section 4.5.1.

5.2.1 Conventional Cross Attention

The initial Cross Attention scheme operated by replacing one Query matrix with a second one, from a different source, and achieved remarkable success [Vaswani *et al.* 2017], making it a natural starting point. Instead of combining the Query matrices from the primary and auxiliary branches, as we finally do in Equation (4.12), the matrices are swapped:

$$\begin{aligned} \hat{\mathbf{Q}}_p^l &= \mathbf{Q}_a^l \\ \hat{\mathbf{Q}}_a^l &= \mathbf{Q}_p^l \end{aligned}, \tag{5.1}$$

where $\hat{\mathbf{Q}}_{\{p,a\}}^l, \mathbf{Q}_{\{p,a\}}^l \in \mathbb{R}^{(C+N^2)\times d}$, C is the number of class tokens, N^2 is the number of patch tokens, and d is the dimensionality of the tokens. $l = 2i, i \in [0, \frac{L-1}{2}], i \in \mathbb{Z}$ is the

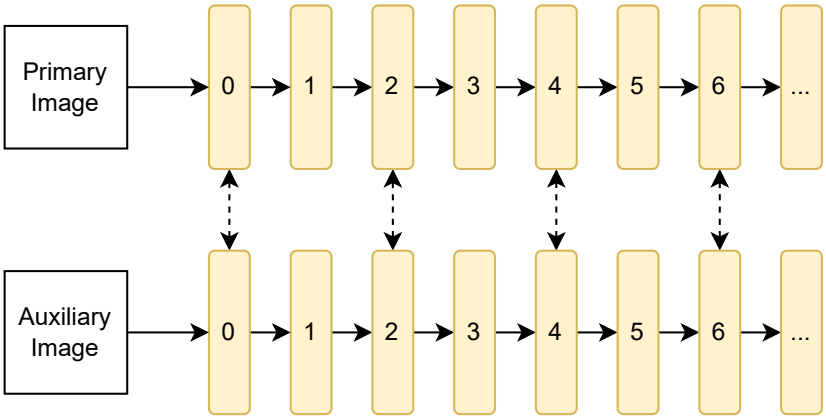


Figure 5.3: Diagram showing the application of **conventional cross attention** at alternating transformer layers, specifically, even-indexed layers.

even-indexed transformer layer, where L is the total number of layers — we alternate the layers at which we apply cross attention. This is graphically depicted in Figure 5.3.

This is the cross attention method used in Section 5.1, and seeing as this is the initial cross attention attempt and will serve as a building block, we sample the auxiliary image in a number of simple, but naive ways, which change depending on whether training or inference is being performed (these are the same mining strategies introduced in Section 5.1, briefly re-stated here):

1. Training Mining Strategies:
   (a) Random: the auxiliary image is randomly chosen from the entire training image set, with the sole provision that the primary image is not sampled.
   (b) Exactly 1 Overlap: the auxiliary image is randomly chosen from the entire training image set, such that it has exactly 1 class in common with the primary image, and that it is not the primary image.
2. Inference Mining Strategies:
   (a) Same: the primary image is also used as the auxiliary image. Each image will have a distinct set of tokens.
   (b) Exactly 0 Overlap: the auxiliary image is randomly chosen such that it has no classes in common with the primary image.
   (c) Exactly 1 Overlap: same as the corresponding training mining strategy.
   (d) At Least 1 Overlap: the auxiliary image is randomly chosen such that it has at least 1 class in common with the primary image, and that it is not the primary image.

Revisiting the VOC experiments from Section 5.1, Figure 5.4 compares the inference mining strategies for the two training mining strategies. Evidently, using the primary image as the auxiliary image as well ("Same" strategy) is the most beneficial, thus we use it henceforth. Seeing as the two branches share their weights, and the class tokens

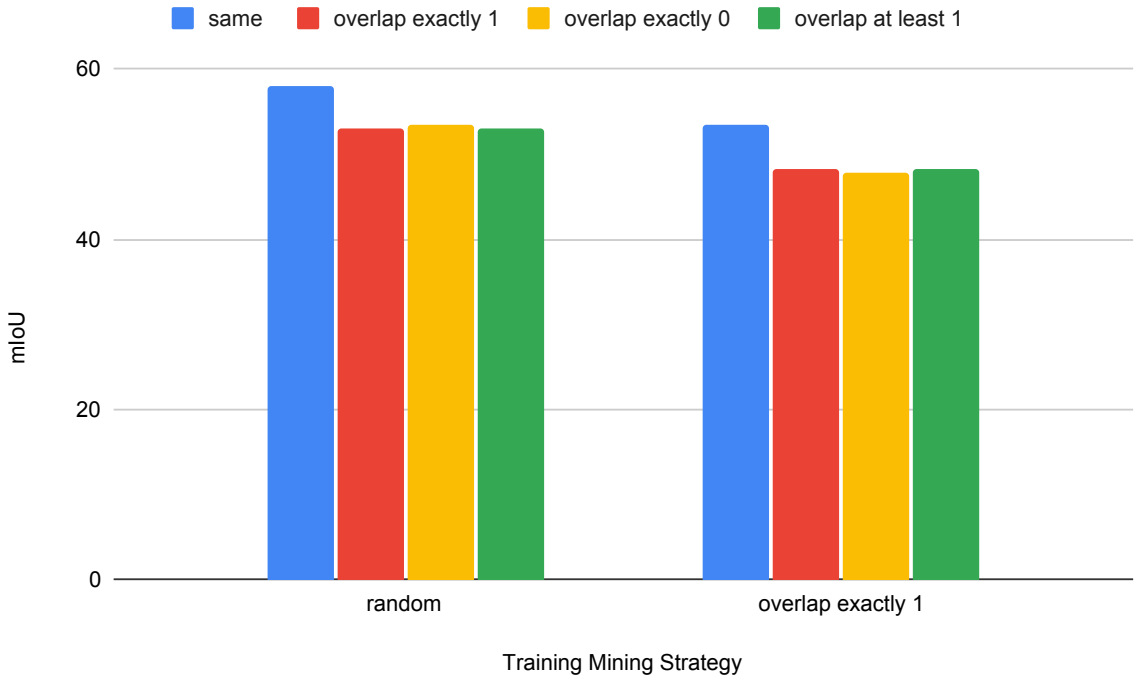


Figure 5.4: Comparison of overall pseudo label quality (mIoU) for different training mining strategies and different inference mining strategies, for VOC. The results are grouped according to the training mining strategy.

are initialised to the same pretrained DeiT-S class token (as mentioned in Section 4.4), this is tantamount to no cross attention being performed during inference. Cross Attention will still impact the training, but this inference strategy will just exchange the same information between the two branches.

Additionally, we introduce asymmetry to the two branches. The primary branch will perform the classification as normal (predict the classes present in the image). The auxiliary branch is trained to predict either the union or the intersection of the classes in the primary and auxiliary images (hereafter called the Overlap task) — the information to enable this task is made available through the cross image attention. The intention behind this asymmetry is to encourage the model to use the shared information instead of potentially learning to ignore it. An important addition is that of a multitask prompt [Sanh *et al.* 2022] to the class and patch tokens for each branch. The class tokens are initialised identically and there are no unique and learnable parameters for either branch, so the gradient path for both branches is the same. Therefore, multitask prompting is the only way for the model to distinguish between having to perform the Classification and Overlap tasks. Specifically, the first dimension of the tokens for the primary and auxiliary branches are set to 0 and 1, respectively.

The final model variation we introduce is to only apply the cross image attention in one direction, from the primary branch to the auxiliary branch (called Explicit Cross

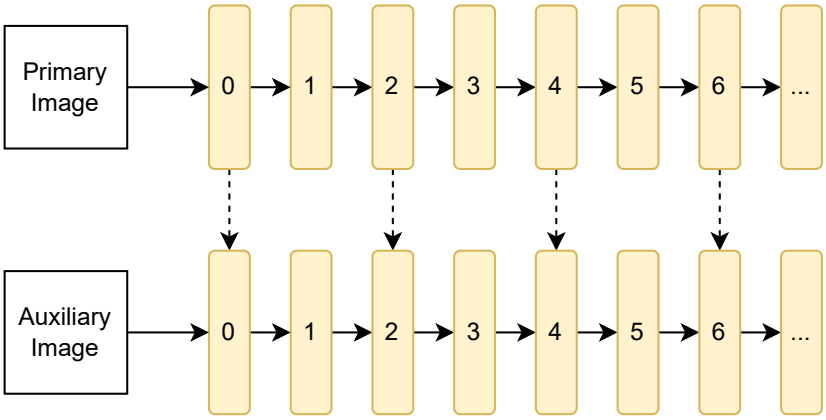


Figure 5.5: Diagram showing the **explicit** application of cross attention at alternating transformer layers, specifically, even-indexed layers, from the primary to the auxiliary branches.

Attention):

$$\begin{aligned} \hat{\mathbf{Q}}_p^l &= \mathbf{Q}_p^l \\ \hat{\mathbf{Q}}_a^l &= \mathbf{Q}_p^{l\,\prime} \end{aligned} \tag{5.2}$$

where the shapes, subscripts, and superscripts are the same as in Equation (5.1) — we alternate the layers at which we apply cross attention. This is graphically depicted in Figure 5.5. The auxiliary branch, performing the Overlap task, is the one that actually requires the shared information, so we can be more explicit with the application of the cross image attention. The fact that the weights between the branches are shared means that the primary branch will indirectly use the exchanged information as well.

Figure 5.6 displays the results of testing Conventional and Explicit Cross Image Attention, for the two training mining strategies, and for the two overlap tasks. Results are presented for the overall performance as well as for the most prominent class co-occurrence pair in VOC, the chair and dining table, since this is the case we primarily intend to improve. In all these cases, the explicit cross attention scheme (in blue shades) outperforms the conventional scheme (in red shades), but still falls short of the baseline performance — adjacent blue and red bars are to be compared. Nevertheless, a targeted approach to applying the cross image attention is evidently preferable to the conventional approach. There is no distinguishing between taking the intersection or the union for the overlap task, so we shall continue to test both.

5.2.2 Impact of Overlap Task

As shown in Figure 5.6, whilst there is little distinction between the performance of the union and intersection tasks, the mere introduction of the Overlap task leads to degraded performance. We initially introduced the Overlap task in an attempt to encourage the model to use the exchanged information and not just learn to ignore it. However, the worsened performance requires us to ascertain the impact the Overlap task is actually having.

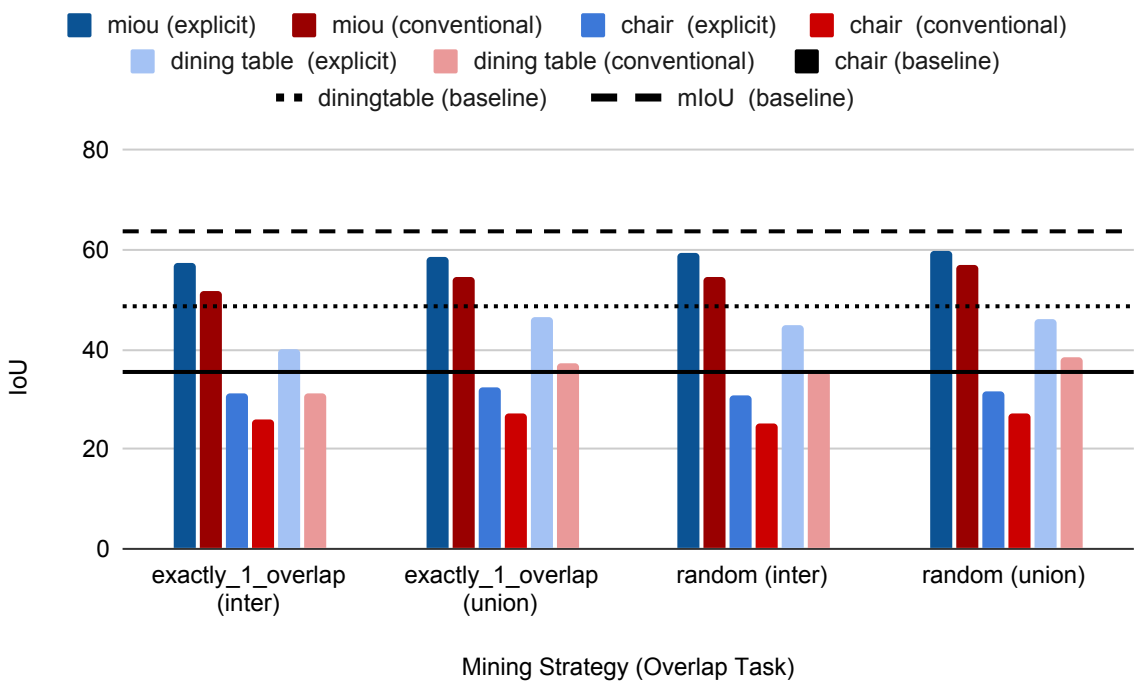


Figure 5.6: Comparison of overall pseudo label quality (mIoU) for Conventional (red shades) and Explicit Cross Image Attention (blue shades), for VOC. The results are grouped according to their training mining strategy and the overlap task. The results for the most prominent class co-occurrence pair (chair and dining table) are displayed, along with the overall mIoU performance — the different colour shades reference these classes. The baseline comparison to MCTFormer+ is included as the black lines.

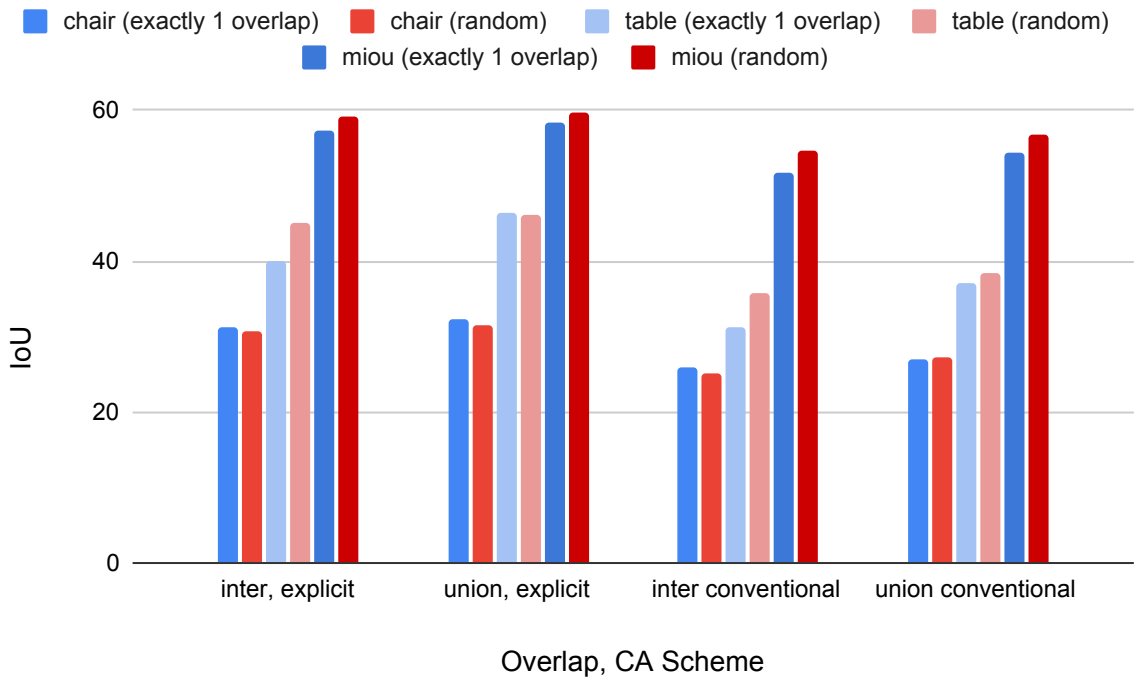


Figure 5.7: Graph comparing the performance of different training mining strategies for different configurations of the Overlap task and cross attention scheme (for VOC). Random mining models are in red, Exactly 1 Overlap models are in blue, and the shade of colour references the class — each pair of columns compares class performance for the two mining strategies. Results are shown for the most prominent co-occurring class pair (chair and table) and for the overall performance (mIoU). This is a different view of the results in Figure 5.6.

From the investigation of explicitly applying cross attention (Figure 5.6) we reformat the results to compare the efficacy of the two training mining strategies: Random and Exactly 1 Overlap. In all otherwise-equivalent configurations, Figure 5.7 shows the models using Random mining outperform their counterparts. We also previously found that explicit cross attention was preferable to conventional cross attention. So we choose a particular model configuration to investigate the impact of the Overlap task: Explicit Cross Image Attention, with the intersection as the Overlap task, trained using random mining.

The Classification and Overlap tasks are performed by the primary and auxiliary branches respectively, and originally had equal weighting to the overall loss (i.e. each branch's loss component had a coefficient of 1). By changing the coefficient of the Overlap task (the auxiliary branch), we gain insight into its overall effect.

Figure 5.8 shows that the Overlap task itself is a source of confusion to the model. When the Overlap task is turned off (0 coefficient), the performance is on-par with MCTFormer+, which is expected since this reduces the dual-branch model to the MCTFormer+ case. Even a marginal contribution (0.1 coefficient) leads to degraded performance, a trend which is exacerbated as the contribution of the Overlap task increases.

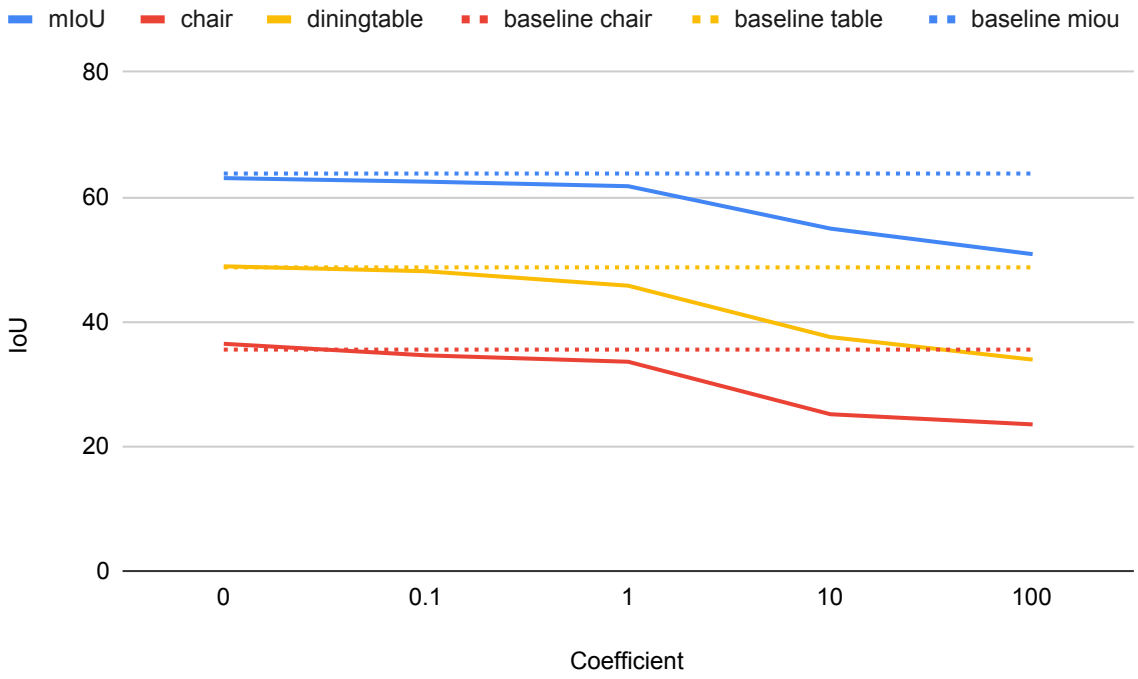


Figure 5.8: Graph comparing the performance of different weightings of the Overlap task. Results are shown for the most prominent co-occurring class pair (chair and table) and for the overall performance (mIoU), for VOC. Baseline performance from MCTFormer+ is included (dotted lines).

This raises significant questions about the semantics of applying cross image attention in this way. Explicit Cross Attention outperforms Conventional Cross Attention: by manually removing an aspect of the information sharing, performance improved. Therefore, it is likely that these two cross attention schemes are introducing internal noise to the model, which it is not able to learn to overcome.

5.2.3 Explicit Cross Image Attention at a Single Layer

Following the investigations of Section 5.2.1 and Section 5.2.2, we expect that the Conventional and Explicit Cross Image Attention schemes are introducing noise to the model. By manually removing an aspect of this noise when halving the amount of cross attention being performed (by going from conventional to explicit schemes), performance improved. Thus an even more targeted approach to the cross attention scheme is necessary.

Akin to later convolutional layers in CNNs learning more and more abstracted features [Zeiler and Fergus 2014], we make the assumption that early layers in the transformer learn low-level information, which gets progressively more abstract deeper in the model (this assumption is challenged in Section 5.2.8). Therefore, we perform a sweep, where the Query matrix from the primary branch is swapped into the auxiliary branch (i.e. similarly to the explicit cross attention), for a single layer at a time, for the second

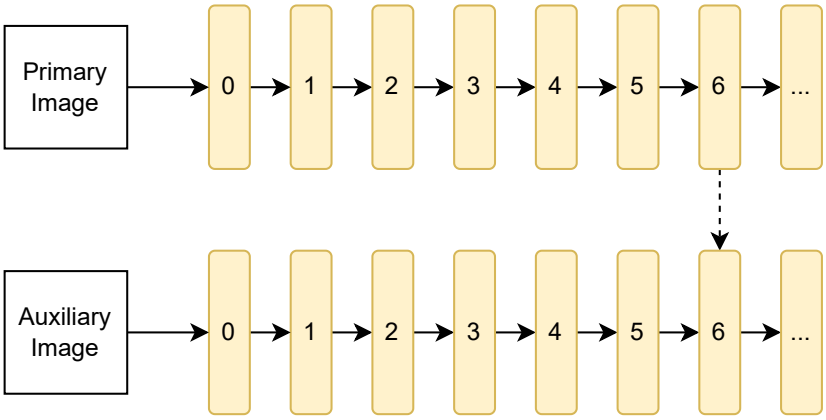


Figure 5.9: Diagram showing the **explicit** application of cross attention at a **single** transformer layer, specifically, layers in the second half of the transformer, from the primary to the auxiliary branches.

half of the transformer. Mathematically, this is identical to Equation (5.2), but where $l \in [\frac{L}{2}, L-1], l \in \mathbb{Z}$, and l takes a single value per model. This scheme is depicted graphically in Figure 5.9.

Following the finding in Section 5.2.2 that using random mining during training outperforms the other mining strategies, the models here will perform random training mining only. They will also use the Overlap task.

Figure 5.10 presents the results of applying this Single Layer Explicit Cross Image Attention to the second half of the transformer, using VOC. Noticeably, the union task seems to be outperforming the intersection task on our classes interest and for the overall performance, even going so far as to outperform the baseline for the chair and dining table classes for some configurations, namely when applied to Layers 7 or 11 (12 0-indexed layers in total). This is a notable improvement, for at least the Union models, over the Explicit Cross Image Attention scheme (the square line, which was presented in Section 5.2.1) which fell quite short of the baseline performance. Even though the overall mIoU score decreased compared to the baseline, the fact that the co-occurring class pairs experienced an improvement is important.

Now that we have a slight improvement for the worst pair of co-occurring classes in VOC, it is prudent to conduct further experiments using the much harder COCO dataset. Figure 5.11 establishes results using COCO for the same configurations that achieved high performance on VOC: included are intersection and union tasks using explicit cross attention, the intersection task applied to Layer 8 using single layer explicit cross attention, and the union task applied to Layers 7 and 11 using single layer explicit cross attention. Firstly, knife and fork have a model (union task, single layer cross attention applied to Layer 11) which outperforms the baseline by a substantial amount. Of the remaining two pairs of classes, baseball bat, baseball glove, and mouse each have one of the models slightly outperforming the baseline, and keyboard does not outperform the baseline for any of the models. Notice that the single-layer scheme generally outperforms the explicit cross attention scheme, which aligns with our findings on VOC.

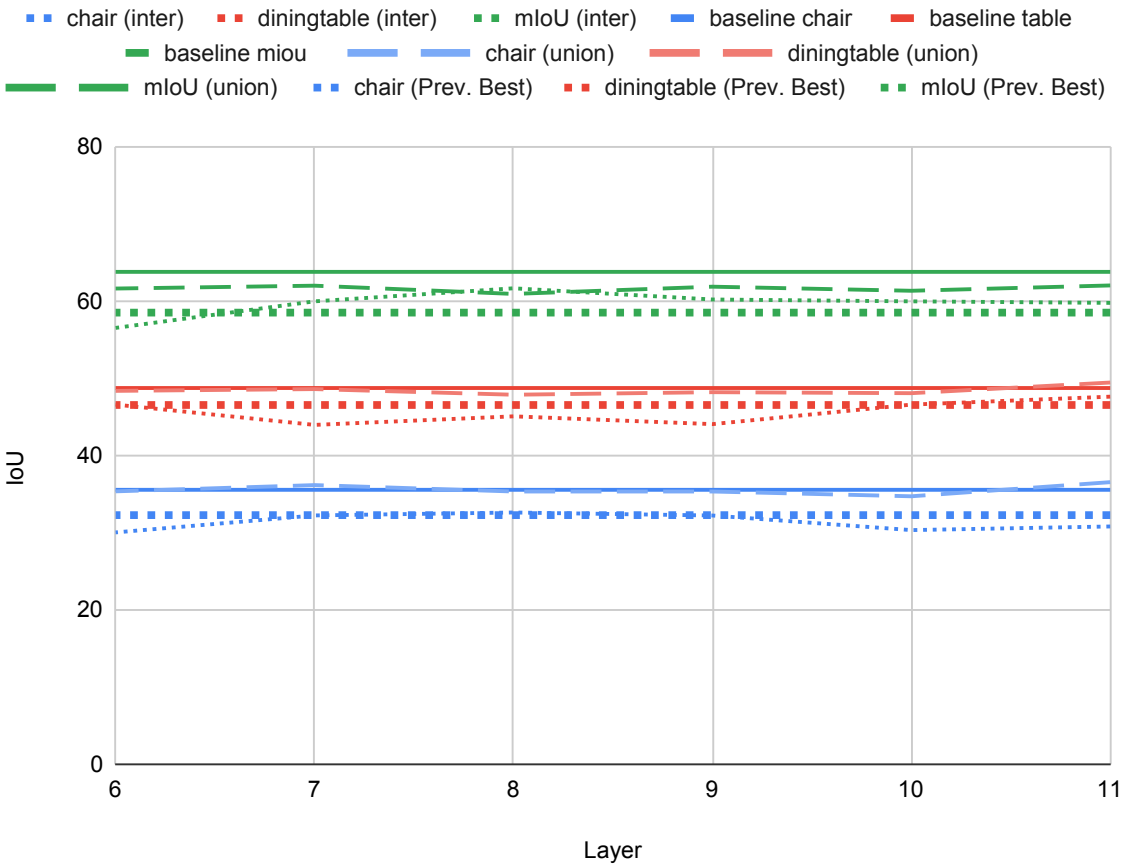


Figure 5.10: Graph showing the performance when the primary Query matrix explicitly replaces the auxiliary Query matrix for a single layer, for various layers, for VOC. Measured are the most prominent class co-occurring pair (table, chair) and the overall performance (mIoU). Both the overlap tasks, intersection and union, are presented, as well as baseline comparisons to MCTFormer+ and the single best explicit cross attention model (Exactly 1 Overlap training mining with the union overlap task). The colour legend is: green for mIoU, red for dining table, blue for chair. The line shape legend is: "—" (solid) for MCTFormer+ baseline, "- -" (dashed) is for the union models, "· · ·" (dotted) is for the intersection models, "■ ■" (squares) is for the previous best model.

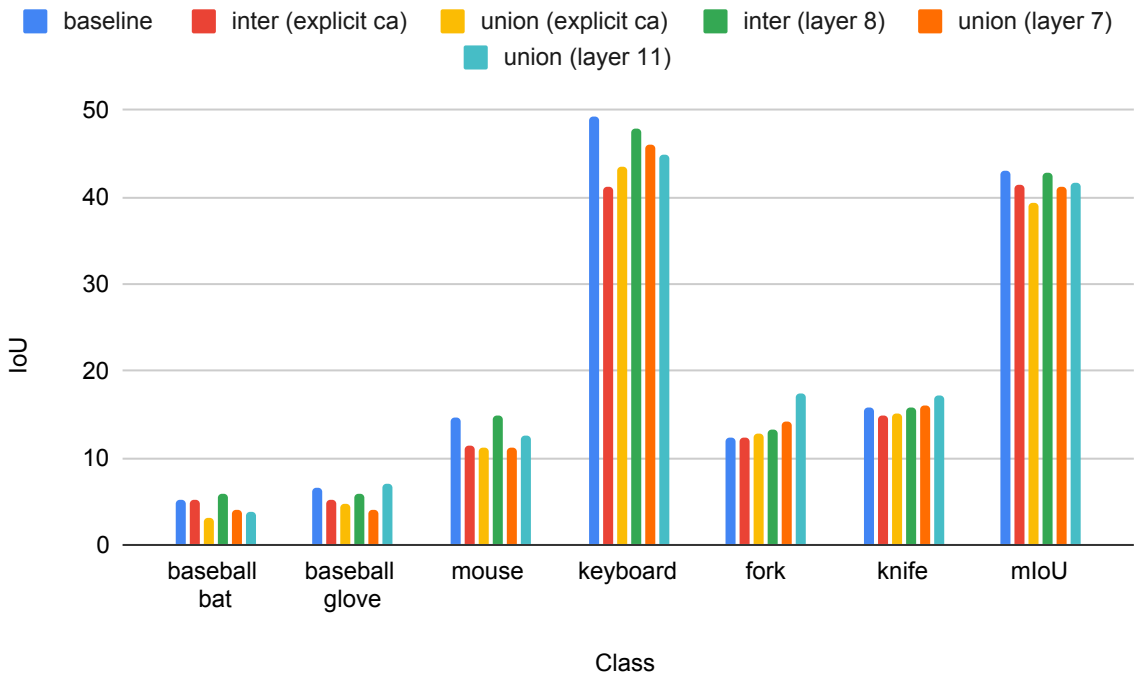


Figure 5.11: Graph showing the performance for Explicit Cross Image Attention and Single Layer Cross Image Attention for select configurations, for COCO. The models presented are COCO comparisons to earlier, equivalent, and high-performing models on VOC. Measured are the most prominent class co-occurring pairs (Baseball Bat & Baseball Glove, Mouse & Keyboard, Knife & Fork) and the overall performance (mIoU). A baseline comparison to MCTFormer+ is included.

There is no single model which achieves these increased scores, but rather the Intersection applied to Layer 8 and the Union applied to Layer 11 which share them. However, all the models perform slightly worse than the baseline overall, thus there is still room for improvement.

5.2.4 Explicit Cross Image Attention at Multiple Layers

Section 5.2.1 and Section 5.2.2 showed that the Explicit Cross Image Attention scheme, which alternates the layers at which the Query matrix is replaced, was preferable to Conventional Cross Attention, leading us to surmise that the reduction in cross attention reduced some of the internal noise. Section 5.2.3 showed that the Single Layer Explicit Cross Image Attention, which replaces the Query matrix at a single layer, was even more beneficial. However, it does represent an extreme reduction to the degree of cross attention being applied. Therefore, it stands to reason that there is some middle ground, between applying cross attention at half the layers (i.e. alternating the layers) and at only one of the layers, where performance will peak.

We choose the halfway point of 3 layers, and choose to use consecutive layers — there is no particular reason for using consecutive layers, and Section 5.2.8 shows that there

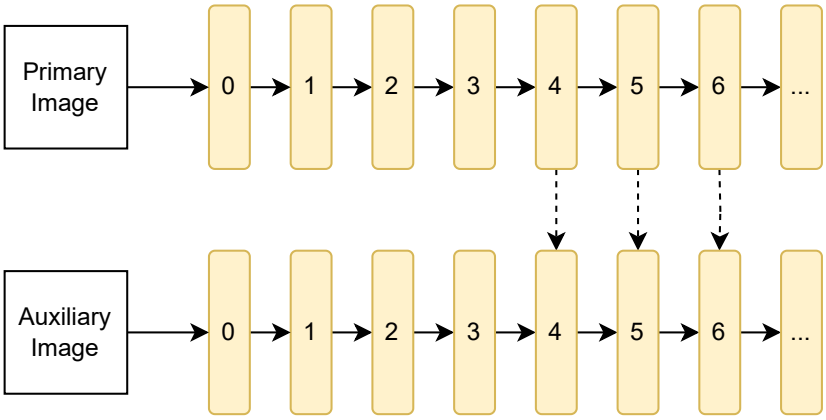


Figure 5.12: Diagram showing the **explicit** application of cross attention at **multiple, consecutive** transformer layers, specifically, layers in the second half of the transformer, from the primary to the auxiliary branches.

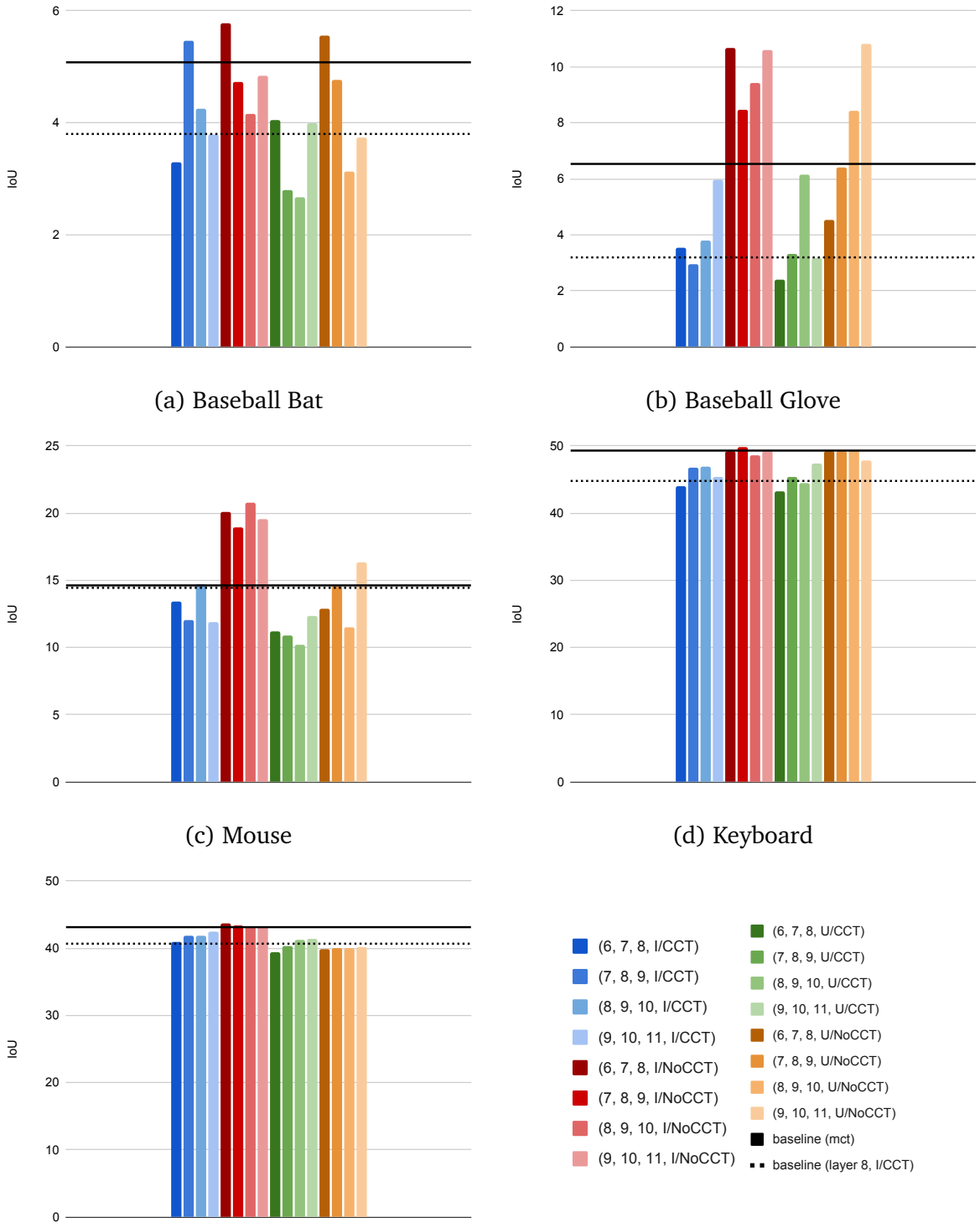
can be variability in these choices with minimal impact to the performance. This is depicted graphically in Figure 5.12.

Figure 5.13 shows the results of this multi-layer cross attention scheme, presenting model configurations for the intersection or union tasks, and the inclusion or exclusion of contrastive learning (CCT) from both branches — to briefly recap the CCT intuition from Section 4.5.2, it is a contrastive learning module which drives the class tokens to learn distinct representations, by encouraging them to be orthonormal. We include the configurations without contrastive learning since initial results from the reimplementation of MCTFormer+ in Section 4.3.5 suggest that in its current form, CCT degrades the performance for co-occurring class pairs.

All the models using the intersection Overlap task outperformed their union counterparts (blue vs green and red vs orange, where the former colour is the intersection task). This is not something earlier experiments displayed, so it is an interesting development considering the attractiveness of the intersection task: the intersection task encourages the model to disambiguate classes that only occur in one of the primary and auxiliary images, whereas the union task, by its very design, does not since it learns to predict all classes present in both images.

Note that, for the most part, the results excluding contrastive learning (red vs blue and orange vs green, where the former colour excludes CCT) either maintain or improve the performance of co-occurring classes of interest and the overall mIoU performance. This phenomenon is further investigated in Chapter 6, but for the remaining analysis in this section, the best performing model configuration is used: trained using the intersection task and no contrastive learning.

The major result, shown in Figure 5.14, is substantial improvement in the co-occurring class pairs of interest, whilst more-or-less maintaining the overall mIoU performance. Although, not all the layer groups improve all these co-occurring classes. Figure 5.14 shows the percentage improvement over MCTFormer+: the baseball glove and mouse classes achieve a consistent improvement of more than 25%, with the baseball glove



(a) Baseball Bat

(b) Baseball Glove

(c) Mouse

(d) Keyboard

(e) Overall Performance (mIoU)

(f) Legend

Figure 5.13: A grid of graphs comparing the performance of Multi-Layer Cross Image Attention for different groups of layers, for COCO. The layers chosen are in groups of 3, from the second half of the transformer. Each graph shows results for a prominent class from a co-occurrence pair (baseball bat & baseball glove, mouse & keyboard) and the overall performance (mIoU). The following model configurations are used:
(a) intersection overlap task and CCT applied to both branches (blue, I/CCT),
(b) intersection overlap task and no CCT applied to both branches (red, I/NoCCT),
(c) union overlap task and CCT applied to both branches (green, U/CCT),
(d) union overlap task and no CCT applied to both branches (orange, U/NoCCT)
(e) baselines of MCTFormer+ (solid black) and intersection overlap task with overlap task applied to layer 8 (dotted black).

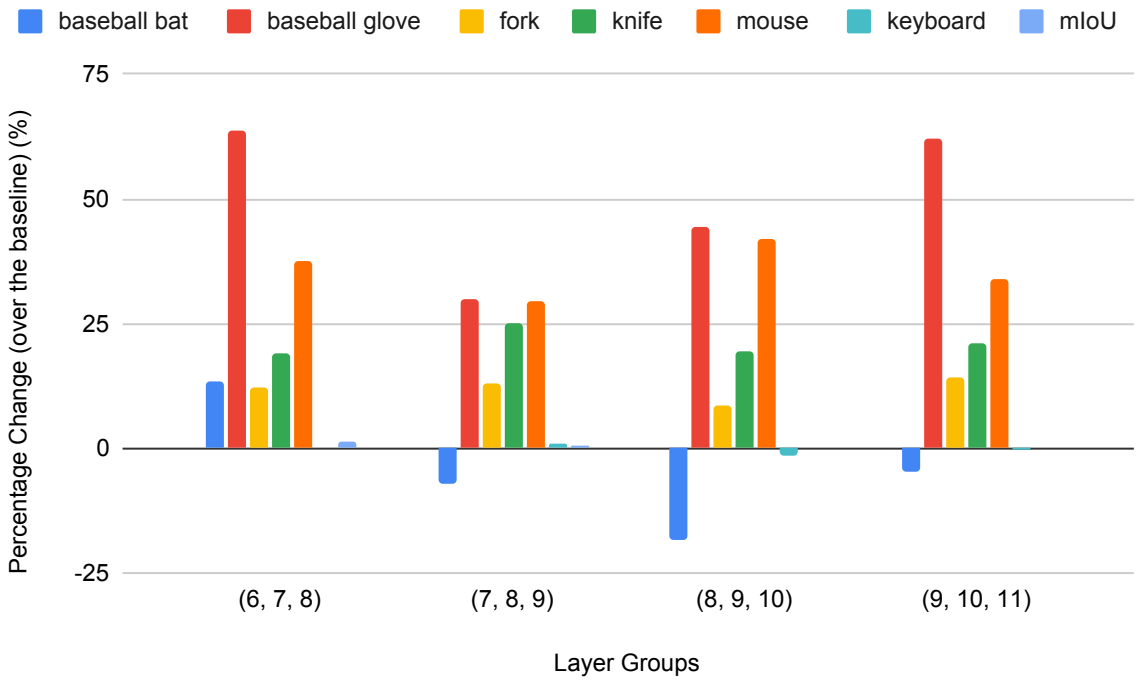


Figure 5.14: Diagram showing the percentage increase in IoU performance for different groups of layers compared to MCTFormer+, when applying multi-layer cross attention, for COCO (values of 0 are unchanged, < 0 have worsened, and > 0 have improved). The models used are trained with the Intersection overlap task and no contrastive learning (these are initially shown in Figure 5.13 and reformatted for display here).

achieving greater than 50% improvement for two layer groupings; the knife and fork classes also achieve large improvements, but not greater than 25%. The baseball bat only improves for one layer grouping and worsens for the others, which is troubling, but the overall performance (mIoU) is maintained for all layer groupings, which is a positive result.

Consequently, applying explicit cross attention to multiple (i.e. 3), consecutive layers is beneficial. However, further investigation is required to ascertain why some cases (i.e. baseball bat) deteriorate under this cross attention scheme.

5.2.5 Critical Analysis of Cross Image Attention

In order to understand how the dual-branch model is actually behaving with the cross image attention, we need to inspect the attention maps at each layer of the transformer for some chosen images.

We use the model trained with the Intersection overlap task and explicit cross image attention applied at layer 8 (i.e. a single layer). This single-layer cross attention model performed well for co-occurring classes in COCO and is easier to inspect than the multi-layer cross attention models. Figure 5.15 presents a low co-occurrence class pair case

(elephant and bear) and Figure 5.16 presents a moderate co-occurrence class pair case (sports ball and baseball bat), both from COCO. These two figures only show the refined attention maps for 4 layers, which gives a good impression of the behaviour of the attention maps: an early layer (Layer 4), the ones immediately before and after cross attention is applied (Layers 7 and 8), and the last layer (Layer 11). The former case allows us to see how the attention maps evolve for a case that the model is already capable of solving. This means there will not be noise in the attention maps from an inability to perform its training goal, but rather that any noise (or artefacts) will solely be from the cross attention. Similarly, the latter case presents a slightly harder comparison to understand how cross attention behaves even when the model is less capable of solving the training goal.

Noticeably, the layer at which the cross attention is applied suffers from severe confusion (in the auxiliary branch, since this is the direction in which the cross attention is applied): prior layers had general attention around the objects of interest, but the layers afterwards seemed to lose all semblance to what they were meant to be attending. An interesting phenomenon is that at the cross attention layer, an imprint of objects in the primary image can be seen in the attention of the auxiliary image. This means that information transfer is definitely happening, but that either the cross attention's current form or the auxiliary branch's training task (i.e. the overlap tasks) are deficient and the model is not able to learn an effective way of utilising the shared information.

The class classification score indicates the model's ability to recognise that the classes *exist* in the image (i.e. the classification task). The value is on a scale from 0 to 1, and is passed through a sigmoid activation function before being compared to the ground-truth labels. For the harder case in Figure 5.16, the model struggled to classify the objects. For example, the model correctly classifies the baseball bat in the primary image with decent attention coverage, but only attends to a small region of the sports balls and even misclassifies these pixels. This has a knock-on effect, since the bad attention for the sports balls in the primary image causes the model to be uncertain about their presence in the auxiliary image (when previously it was confident), thereby causing the classification score to fall and the sports ball attention to become quite poor. The model is able to recover the auxiliary sports ball attention, but loses a lot of effort in the process.

Consequently, we have gained a general understanding that the cross attention is not performing as intended, despite the favourable results obtained in Section 5.2.4. However, we have confirmed that cross image attention is a valid, and capable, method of information sharing between the two branches. Therefore, a more principled approach is required, which is discussed in Section 5.2.6.

5.2.6 Principled Revision of Cross Image Attention

Following the critical analysis of current cross image attention schemes in Section 5.2.5, a more principled approach to the development of an effective cross image attention scheme is required. We observed that cross image attention is quite capable of transferring information from one branch to the other, but that the particular way it was

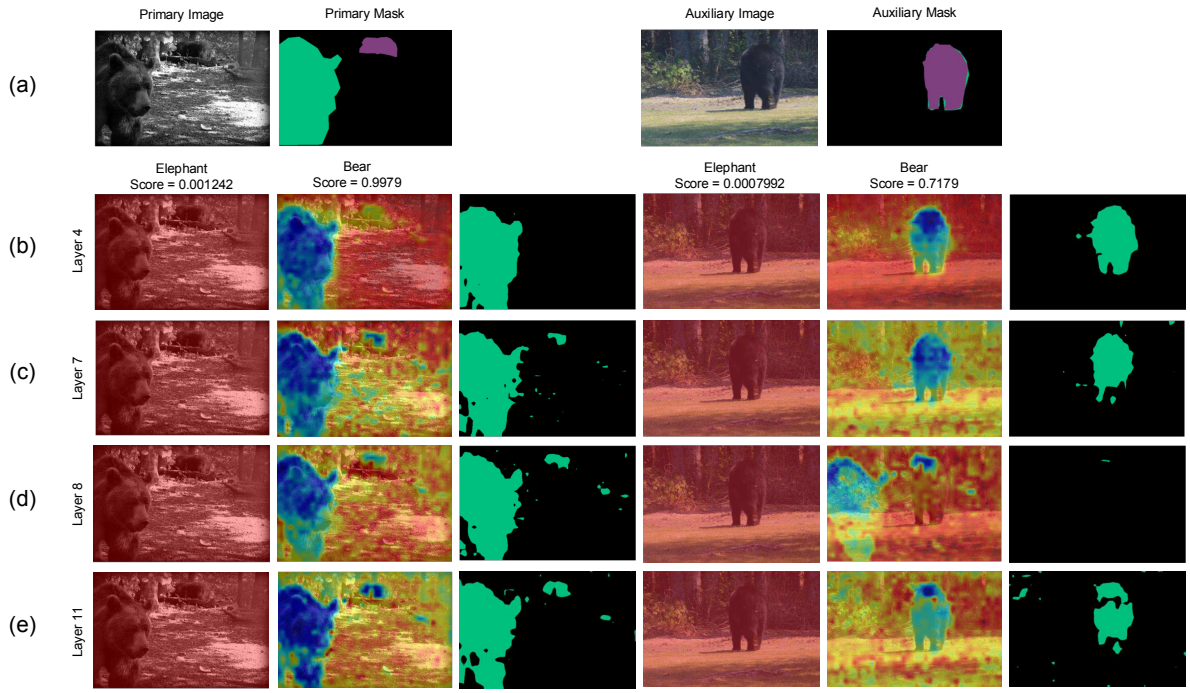


Figure 5.15: **Not Co-occurring** Case (0.18%). Diagram showing the attention maps for the primary (left) and auxiliary (right) images, when Explicit Cross Image Attention is applied just before the Layer 8 row (using the Intersection Overlap task). The top row shows the original images and corresponding ground-truth pixel masks, which are taken from COCO. There are only two classes present, bear and elephant, so we show the attention maps for these classes only. Each image has three columns, where the first two columns are the final refined attention maps (according to Section 4.3.4) from the specified transformer layer and for the specified class. The classification score ($\in [0, 1]$) for the specified class is included as the header for the column. The final column for each image is what the pseudo label would be if the fused attention maps from that layer were used. The classes involved for this case are not co-occurring. **The attention or activation heat map is scaled from red (low) to blue (high).**

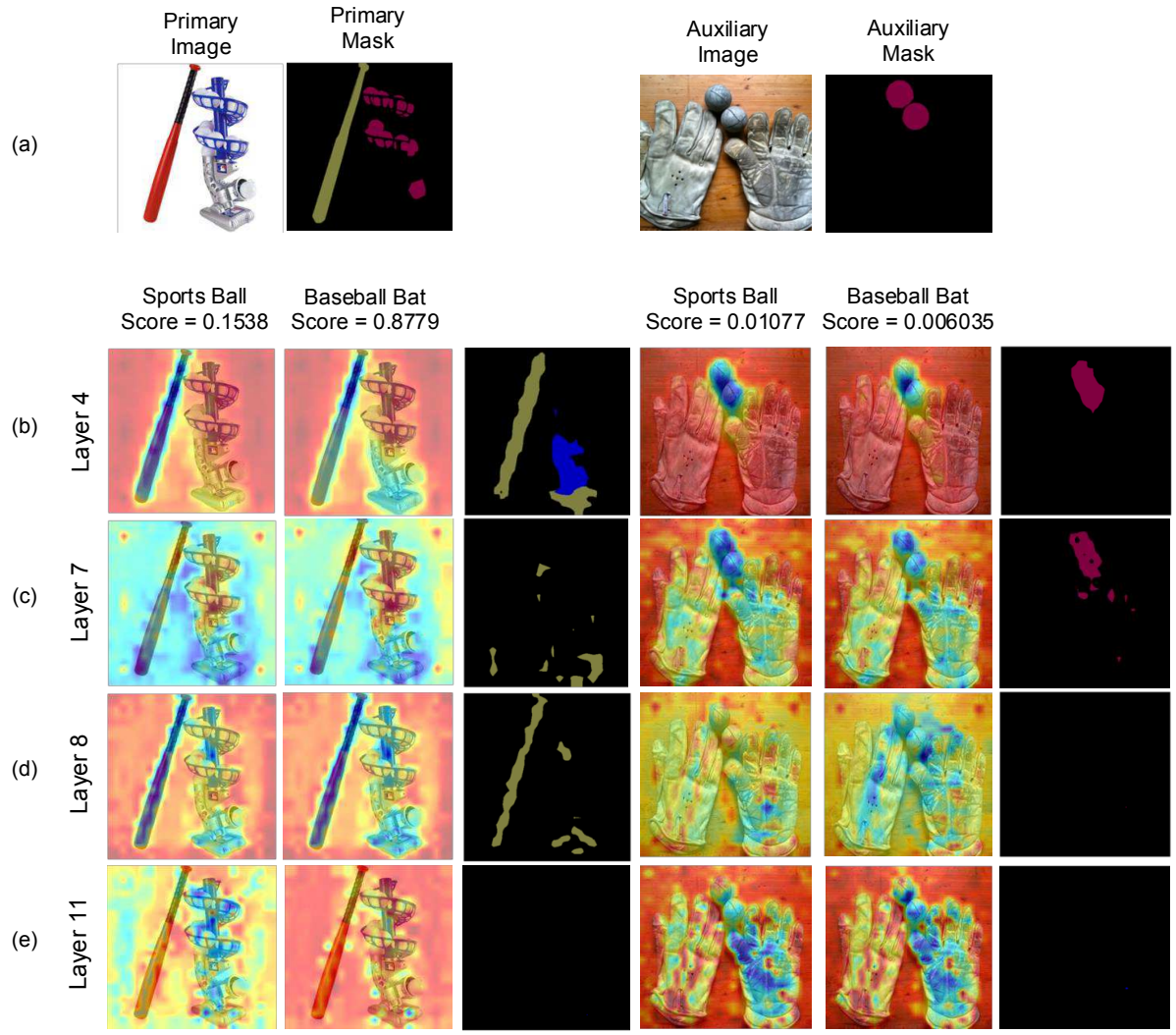


Figure 5.16: **Moderately Co-occurring** Case (16.43%). Diagram showing the attention maps for the primary (left) and auxiliary (right) images, when Explicit Cross Image Attention is applied just before the Layer 8 row (using the Intersection Overlap task), where the images are taken from COCO. There are only two classes present, sports ball and baseball bat, so we show the attention maps for these classes only. The columns are in the same configuration as in Figure 5.15, with the classification score ($\in [0, 1]$) for the specified class included as the header for the column. The classes involved for this case are moderately co-occurring. **The attention or activation heat map is scaled from red (low) to blue (high).**

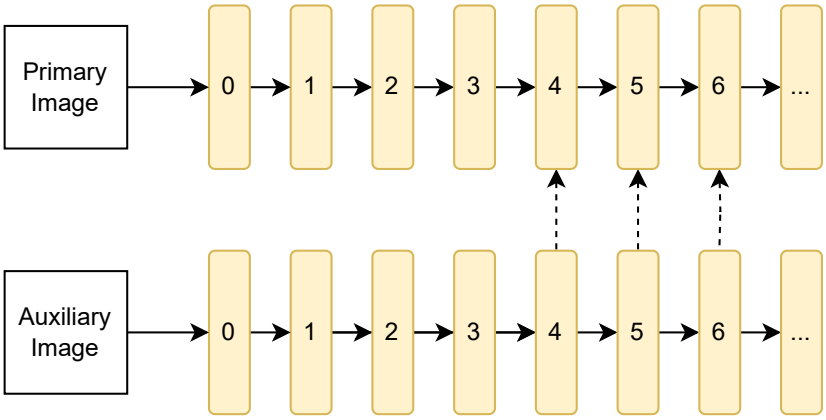


Figure 5.17: Diagram showing the application of cross attention at **multiple, consecutive** transformer layers, specifically, layers in the second half of the transformer, from the **auxiliary to the primary** branches.

transferring the information (i.e. replacing the auxiliary Query matrix with the primary one) led to the model not utilising it effectively.

All the previous cross attention schemes transferred information without considering what information was actually being shared. Our original goal is to solve the class co-occurrence problem. Therefore, the cross image attention should directly share information relating to co-occurring class pairs, but not necessarily information relating to other classes. For example, when considering COCO, the query vectors for the baseball bat and baseball glove pair should be shared, but not those for the bear or elephant classes (which do not co-occur with another class).

Additionally, since it is the primary image that is being segmented, and the auxiliary image is simply meant to aid that process, it is more sensible to change the direction in which the cross attention is applied: from the auxiliary branch to the primary branch. The purpose of this is that the auxiliary branch's recognition of an object is used to improve the recognition of the same class in the primary branch — this is feasible considering that the attention maps in Section 5.2.5 showed an imprint of a class from one image in another image (i.e. class information is transferable). This directional change is depicted graphically in Figure 5.17, for the same multiple, consecutive layers case investigated in Section 5.2.4.

However, Section 5.2.5 made an argument that the explicit cross attention scheme, where the Query matrix in one branch is *replaced* by the Query matrix from the second branch, is deficient. Considering that we are now applying the cross attention with the goal of improving the quality of the attention for specific classes, it makes sense to choose an augmentation scheme for the query matrices from the two branches to reflect that. For example, the model confidence of predicting the existence of a class in the primary and auxiliary images is shown in Table 5.1 for a primary image with both the baseball bat and sports ball classes and an auxiliary image with only the sports ball class. The model is not initially confident about the existence of the sports ball class in the primary image, but after the summation of confidence from the auxiliary

Branch/Class	Baseball Bat	Sports Ball
Primary	1	0.5
Auxiliary	0	0.9
+	1	1.4

Table 5.1: A table showing the probability with which a model thinks a class is present in the primary and auxiliary images. The final row denotes the resulting probability change for the primary image after the two branches' probabilities are summed together. The primary image has both the baseball bat and the sports ball classes, but the auxiliary image only has the sports ball class.

branch, the model has had its confidence regarding the existence of the sports ball in the primary image improved. Consequently, we begin by summing the query vectors from the auxiliary branch to the ones from the primary branch (other augmentation methods are examined in Section 5.2.7).

This new cross image attention scheme (called CIA-AP — Cross Image Attention from the Auxiliary to Primary branch) is what we find most beneficial and, therefore, what we use in our final Cross Image Attention scheme in Section 4.5.1. Consequently, the mechanics and mathematics for CIA-AP is presented in Section 4.5.1.1, and we will re-use the cross attention formula in Equation (4.12) and the masking function, f, in Equation (4.14) here. The only difference to the final formulation is that we begin with the summation as the augmentation method. As such, the function g in Equation (4.13) is redefined as:

$$g(A, B) = A + B, \tag{5.3}$$

for two sets of query vectors, A and B, where the summation is performed element-wise.

However, a different mining strategy is required to sample an auxiliary image to complement the primary one, in order for the cross attention between these two images to be meaningful. Consequently, whenever there is a primary image containing both classes of a co-occurring pair, an auxiliary image that contains precisely one of the pair of classes is chosen. If the primary image does not have a co-occurring pair of classes, then the auxiliary image is randomly mined (random mining is the best performing, naive strategy, as discussed in Section 5.2.2). We manually define which pairs of classes are co-occurring by setting a threshold: any pairs of classes with co-occurrence equal to or greater than this threshold are considered a "co-occurring class pair", otherwise they are not.

We also apply the same co-occurrence threshold to decide which classes to apply cross attention to, since it makes sense that if we mine images based on certain class pairs, we should only use the information from these class pairs for transferring information and potentially improving the attention quality. Note that if the primary image has no "co-occurring class pairs" and random mining is used to obtain the auxiliary image, then cross attention is only applied to the query vectors for the patch tokens.

Finally, the training task for the auxiliary branch is required to be the normal classification task, under this revised cross image attention scheme and mining strategy. Firstly, the auxiliary branch no longer receives additional information from the cross attention (since it operates in the reverse direction now), so its training task needs to be based solely on information contained in the auxiliary image itself. Secondly, the normal classification task will drive the auxiliary class tokens to represent what is in the auxiliary image, so that when they are used for cross attention, they are useful representations. The primary branch remains training on the normal classification tasks because we want to recognise all the classes in the image (we are doing segmentation on this image) — the cross attention will merely apply extra attention (if necessary) to the identified co-occurring classes.

In summary, the revised CIA-AP implements the following:

1. Apply cross image attention from the auxiliary branch to the primary branch.
2. Sum the query matrices together instead of replacing the "target" one (the primary one in this case). This augmentation method is revisited in Section 5.2.7.
3. Only apply cross attention to the query vectors corresponding to certain classes. These classes are co-occurring pairs, as decided by a pre-determined co-occurrence threshold.
4. Cross attention is applied to all the patch token query vectors whenever it is applied to at least one query vector corresponding to a class, a decision which is revisited in Section 5.2.8. However, if there are no classes to which we are applying the cross attention, then we can optionally turn the entire mechanism off and not apply cross attention to the patches either (this setting is called "turnoff" mode).
5. Use the updated mining strategy to choose an auxiliary image with exactly one of a co-occurring pair of classes, whenever the primary image has both of the co-occurring pair of classes. Otherwise, use random mining. The threshold to determine which class pairs are co-occurring is the same as for the cross attention co-occurrence threshold.
6. Use normal classification as the training task for both branches.

Experimental Results

The model configurations tested under this adjusted scheme are as follows:

1. The mining and cross image attention co-occurrence thresholds are set, for COCO, to be 0.46 or 0.2491, which correspond to the top 2 and 10 co-occurring pairs of classes, respectively. After the two worst cases, there is approximately a 16% decrease in co-occurrence to the third case — it is prudent to investigate these two class pairs in isolation. Thereafter, the decrease in co-occurrence is fairly linear, so we also choose to investigate the ten worst cases for comparison.
2. The cross attention is applied selectively to the classes and if no classes are selected (because no co-occurring class pairs are present), then cross attention will

still be performed on the patch query vectors. The optional "turnoff" mode allows the cross attention mechanism to be disabled in these cases, thereby preventing cross attention on the patch query vectors as well.

3. We follow the promising result in Section 5.2.4 and choose to use the 2 best groups of multiple, consecutive layers. These are $(6, 7, 8)$ and $(7, 8, 9)$, since they had the single highest and most stable performance improvement, respectively.
4. Contrastive learning is also temporarily disabled, since it was found to hinder performance in Section 5.2.4.
5. Since contrastive learning is disabled, we also use MCTFormer+ without contrastive learning as our baseline. As noted in Section 4.3.5, this version of the baseline also improves over the one with contrastive learning, although we expect that is, in part, due to the presence of strong co-occurrence in COCO.

Figure 5.18 compares results for our model configurations. At least one of our configurations generally performs on-par or better than the baseline for most of the classes of interest. We can immediately see that we achieve around double the performance for the baseball bat class, with lower, but still notable, improvements for its partner, baseball glove. Despite these impressive improvements, the knife and fork pair generally suffers, and the overall performance slightly decreases compared to the baseline.

For completeness, we include the percentage increase of these models over the MCTFormer+ version with contrastive learning (the baseline we have used previously) in Figure 5.19. In these cases, all the classes of interest, except for a couple of cases, increase performance, and generally do so by a large amount. In this comparison, baseball glove outperforms baseball bat, but both still improve by a large amount. Impressively, there is barely any change in the overall mIoU performance, suggesting we are maintaining good general performance for all the other classes in the dataset.

Comparison to the complementary results to this set, obtained using the Multi-Layer Explicit Cross Image Attention scheme and presented in Figure 5.14, shows superior performance in nearly all aspects for CIA-AP. Specifically, in the best cases, performance for the classes is improved, with larger percentage increases over MCTFormer+, and in the few cases where performance degrades (keyboard for layer group $(6, 7, 8)$ in Figure 5.19a, and fork for layer groups $(6, 7, 8, \text{turnoff})$ and $(7, 8, 9, \text{turnoff})$), it does so to a lesser degree.

Both co-occurrence thresholds tested deliver promising results, and both enabling and disabling "turnoff" mode provided merit. In both of these configurations, there is no single setting that outperforms the others, so we continue investigating these configurations in later sections.

Finally, Section 5.2.1 found that the use of the "same" inference mining strategy outperformed other ones and it became the default approach. However, as discussed in Section 5.3, using such a strategy for the CIA-AP cross attention scheme is detrimental to performance, since it leads to the softmax distribution in the self-attention mechanism becoming more "peaked", which greatly reduces the coverage of the important co-occurring objects. However, Section 5.2.1 also observed that the "same" mining

strategy is tantamount to no cross attention being applied during inference. Therefore, when performing inference with CIA-AP models, we do not apply any cross attention and only pass the primary image through the model.

Consequently, the revised CIA-AP is a success.

5.2.7 CIA-AP Query Augmentation Methods

Section 5.2.6 found CIA-AP to be quite successful and outperforms previous cross image attention methods. One of the main adjustments it made was to sum the relevant query vectors together. However, this introduces potential scaling issues during inference.

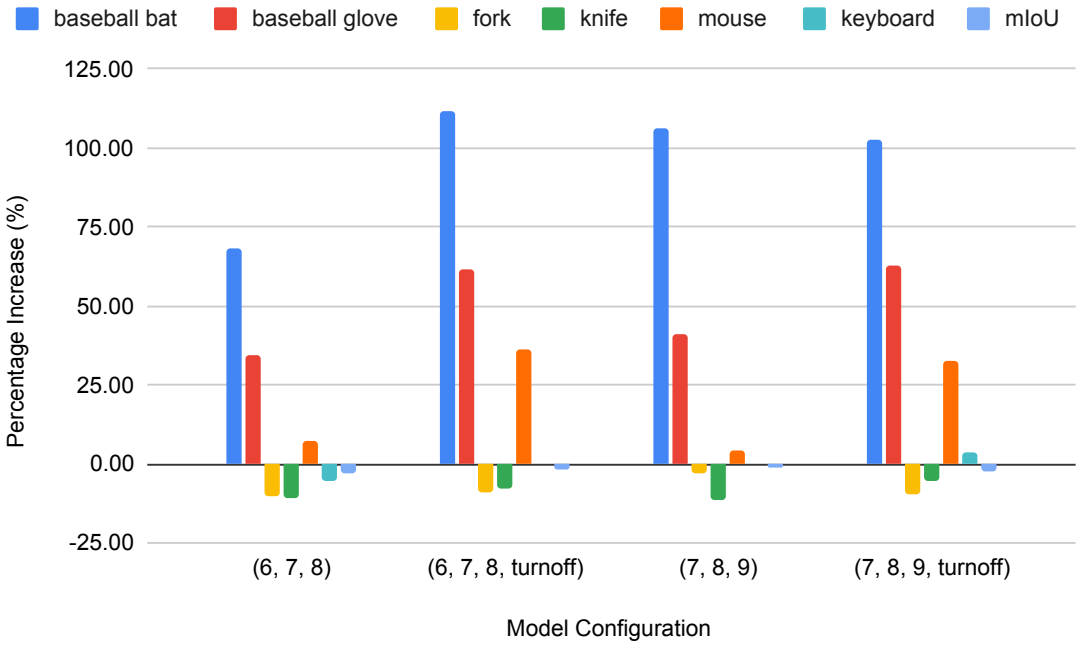
If the normal numerical range for a component of a query vector is $[a, b] \in \mathbb{R}^2$, then by adding two such components together, the range of the sum falls in the range $[2a, 2b]$. During training, this re-scaling is incorporated into the model's learning and it might come to expect that for members of co-occurring class pairs, the query vector will generally be larger. However, during inference, since we no longer perform cross attention, there is no opportunity for these query vectors to be re-scaled. If the model expects a larger scale than it receives, then it might "under-activate" for the co-occurring class tokens.

Therefore, we investigate two other cross attention augmentation methods:

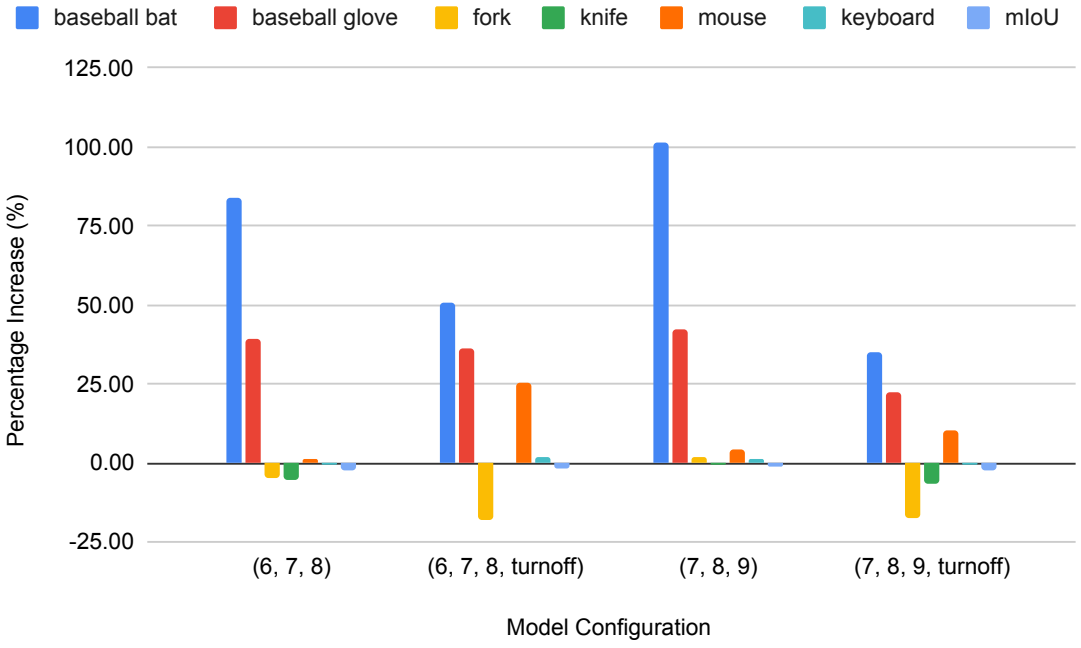
1. Mean: by taking the average of two corresponding components between the query vectors, the numerical scaling is maintained. This is mathematically identical to the final cross attention scheme (in Equation (4.12)) and the final augmentation function, g, chosen for our final model, and represented in Equation (4.13).
2. Swap: by replacing a component in the primary query vectors with the component from the corresponding auxiliary query vector, the numerical scaling is maintained. This is the same augmentation method used for all previous Cross Image Attention schemes. Even though Section 5.2.5 found it to be deficient, we include it for completeness. This is mathematically identical to the cross attention used in the final model, represented in Equation (4.12), with the only change being the augmentation method, g:

   $$g(A, B) = B, \tag{5.4}$$

   where the sets of query vectors A and B originate from the primary and auxiliary branches, respectively.
3. We briefly consider taking the projection of the primary query vector onto the auxiliary query vector, and using that as the updated primary query vector. This implies that we combine the query vector magnitude from the primary query vector with the direction from the auxiliary query vector. The intuition in this projection is that if the primary query vector is ambiguous, then we can refine it by moving it in the direction of the high-quality auxiliary query vector. Performing the projection in the other direction (from the auxiliary to the primary query vector) also has merit. However, we do not pursue this concept further, since the potential decrease in magnitude the primary query vector will incur as a result of

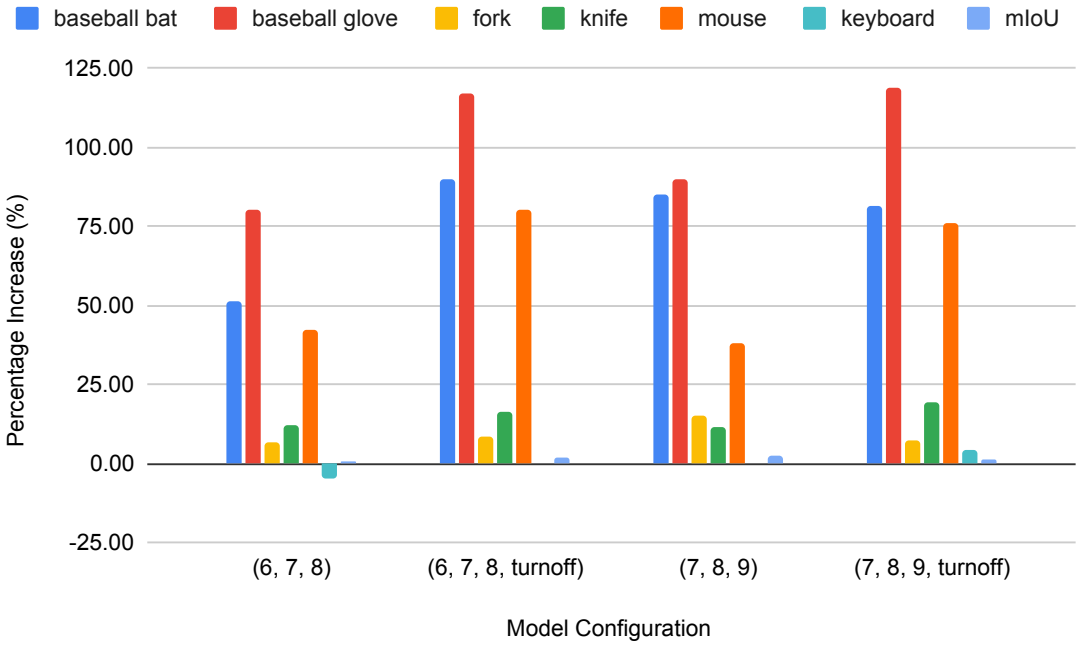


(a) Co-occurrence Thresholds of 0.2491 (top 10 class pairs).

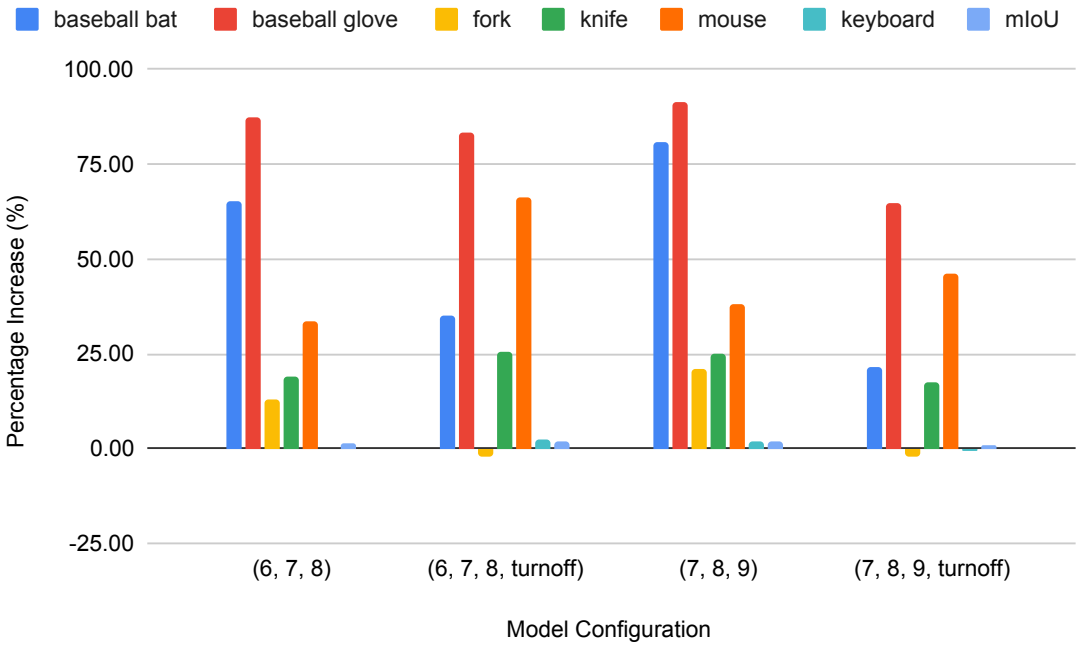


(b) Co-occurrence Thresholds of 0.46 (top 2 class pairs).

Figure 5.18: A grid of graphs comparing the percentage increase in performance of CIA-AP for different groups of layers, for COCO, **over MCTFormer+ *without* CCT**. The layers chosen are in groups of 3, from the second half of the transformer, which were high-performing configurations in Section 5.2.4. "turnoff" indicates that patch cross attention is not used if there are no co-occurring pairs of classes (and so no class cross attention). Each graph shows results for prominent class co-occurrence pairs (baseball bat & baseball glove, mouse & keyboard, knife & fork) and the overall performance (mIoU).



(a) Co-occurrence thresholds of 0.2491 (top 10).



(b) Co-occurrence thresholds of 0.46 (top 2).

Figure 5.19: Graphs showing the percentage increase in performance, **over MCT-Former+ *with* CCT**, for different model configurations in terms of layer groupings, "turnoff" mode for the cross attention being optionally enabled, and co-occurrence threshold. The classes shown are those with the most prominent co-occurrence in COCO (baseball bat & baseball glove, mouse & keyboard, knife & fork) and the overall performance (mIoU).

the projection re-introduces the scaling concern that prompted the investigation of alternate augmentation methods to the "sum" in the first place.

The results from these three augmentation methods (sum, mean, and swap) are presented in Figure 5.20, which presents an aggregation of results from a large number of experiments in order to compare their performance effectively. Plotted are the best performance for each class of interest (which may come from different model configurations) and the average performance over each experiment, per augmentation method. The model configurations investigated are the same as those in the previous section, Section 5.2.6.

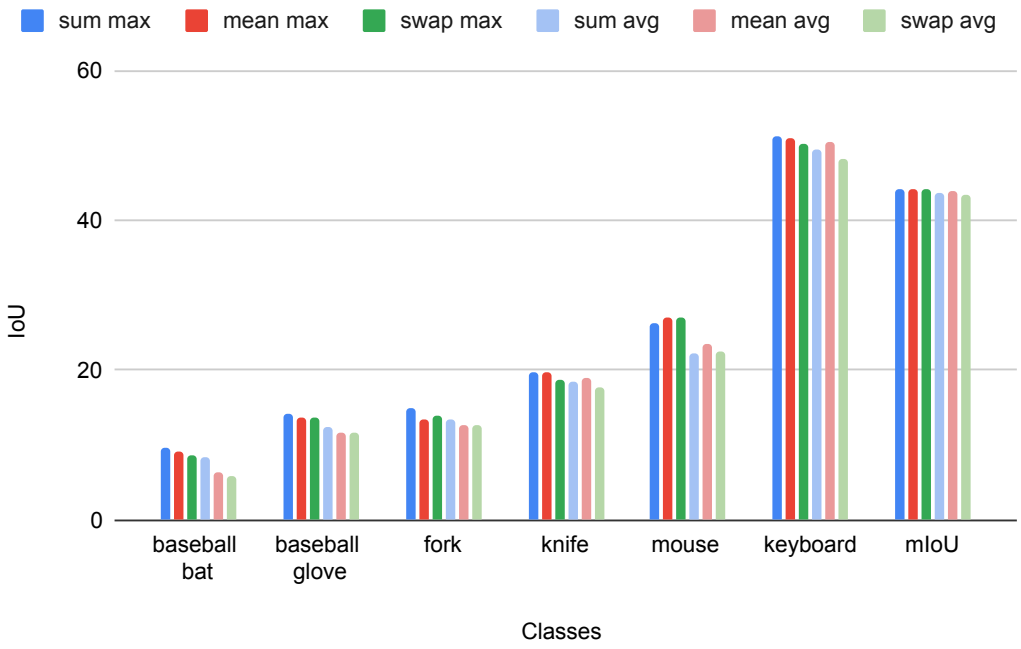
Initially considering the "swap" method, it is outperformed by the other methods on both graphs' metrics (absolute performance comparison and percentage increase over MCTFormer+), for all the classes of interest: in the cases where it outperforms either the sum or the mean, it will only outperform one of them. Combined with the finding from Section 5.2.5 about the deficiency of the "swap" method, we choose to remove it from future investigations.

The "mean" method performs largely on-par with the "sum" method. Even though the percentage increase for the "sum" method appears to be substantially larger than the "mean" method for the baseball bat, baseball glove, and fork classes, the absolute performance increase is much more marginal. For direct comparison between the methods, it is more correct to use the absolute performance, since the percentage increase metric depends on the baseline performance and is tempered by the baseline's predisposition to perform well for some classes but not for others. Additionally, despite that for these same classes (baseball bat, baseball glove, and fork), the absolute performance average is better for "sum" than "mean", the maximum performance is much closer, suggesting that there is a model configuration that performs well even though another configuration performs badly (which is what would drag the average down).

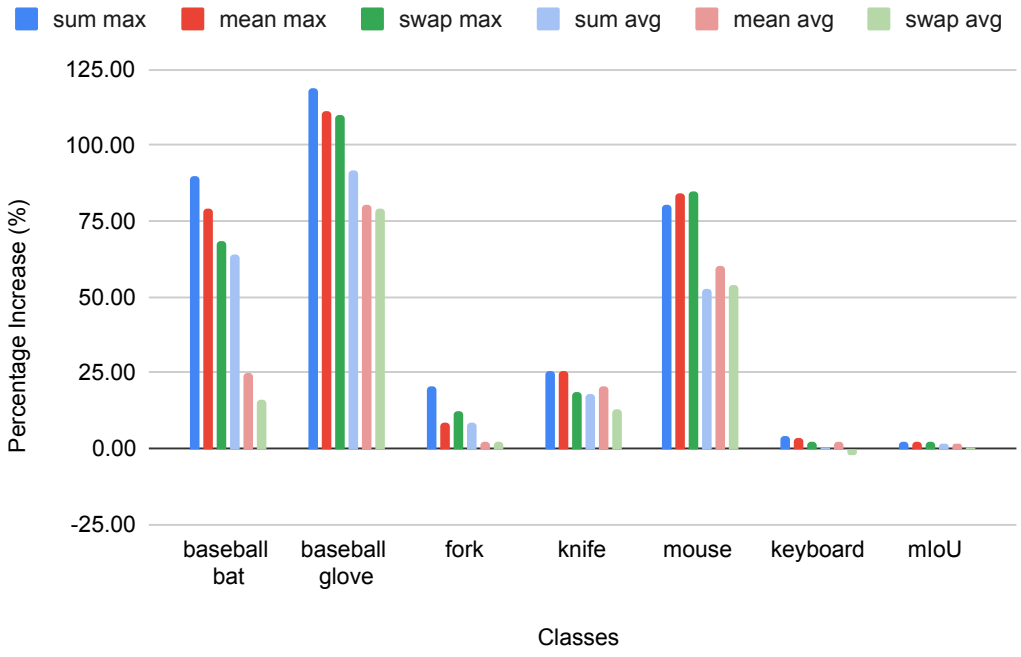
Furthermore, the semantics surrounding the "mean" method are more favourable than the "sum" method: there are no scaling concerns; and the mean also mitigates instances where the model is over- or under-confident for a class's existence in either the primary or auxiliary images.

Therefore, we find that the "mean" method of augmenting the query vectors during cross attention is favourable to the "sum" method.

Figure 5.21 compares the performance of the two co-occurrence thresholds when using the "mean" method (0.2491 in blue and 0.46 in red) — it is just a different view of the results presented in Figure 5.20. Generally speaking, the 0.2491 models outperform the 0.46 ones: for the most co-occurring pair of classes, baseball bat and baseball glove, the former threshold far outperforms the latter, and only the knife and fork pair have 0.46 models slightly outperforming the 0.2491 models. In terms of overall performance, the models trained under the two thresholds are largely similar. Therefore, since our focus is on improving performance for highly co-occurring class pairs, we choose to use 0.2491 as the co-occurrence threshold henceforth. This threshold corresponds to the top 10 pairs of co-occurring classes.



(a) Absolute Performance Comparison.



(b) Performance Percentage Increase over MCTFormer+.

Figure 5.20: Graphs showing the performance comparison for three cross attention augmentation methods: sum, mean, and swap (replacement). Due to the number of experiments, aggregate information for each group is shown. That is, the maximum and mean performance for each class (from COCO) across the tested models is plotted, for each of the augmentation methods. The colours denote the augmentation method. The darker shade is for "maximum" aggregation and the lighter shade is for "average" augmentation.

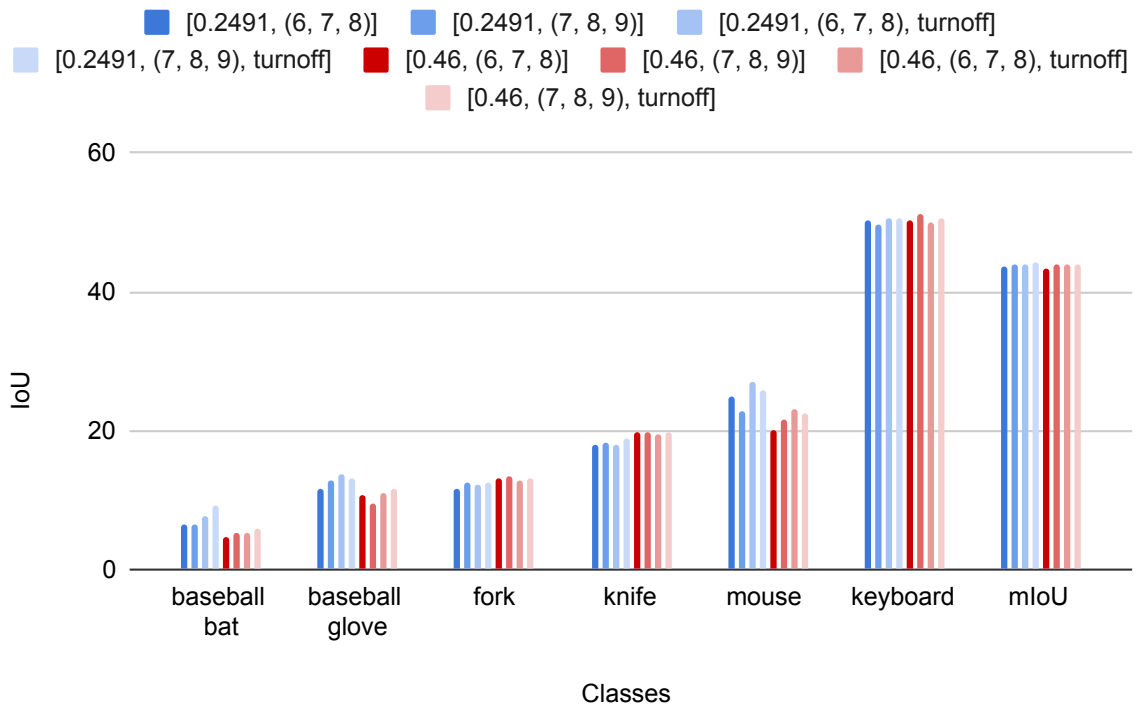


Figure 5.21: Graph comparing the absolute performance of different co-occurrence thresholds for various model configurations, for COCO. The blue results are for 0.2491 (top 10) and the red are are 0.46 (top 2). The legend is formatted at "[*co-occurrence threshold*, *layer group*]".

5.2.8 CIA-AP Layer Choices

This section largely challenges decisions made in earlier sections and argues the merit of certain architectural decisions. Specifically, the following decisions are investigated further:

1. The use of groups of **multiple and consecutive layers** from the **second half of the transformer**, introduced in Section 5.2.4. The decision to use multiple layers is based on finding a middle ground between the underwhelming results of cross attention applied to alternating layers (as in Section 5.2.1) and the promising results of single-layer cross attention (Section 5.2.3). However, the choice of 3 layers was rather arbitrary, as was the restriction that they had to be consecutive. Whilst using layers from the second half of the transformer drew parallels from the behaviour of layers in CNNs, there is no evidence to suggest transformer layers behave the same way, with early layers learning low-level information and later layers learning more abstract information.

2. The use of query vectors corresponding to the patch tokens being used in the cross image attention. Section 5.2.6 introduced "turnoff" mode, whereby no cross attention would be performed if there were no overlapping co-occurring classes between the primary and auxiliary images. However, in the case where there is such an overlapping class, then cross attention is performed between the patch tokens as well. Class tokens encode holistic information about a class and patch tokens encode contextual information about a spatial region of the image. Additionally, the goal of cross attention in CIA-AP is to share high-level and confident information about co-occurring classes and their presence in the images. Therefore, it is reasonable to expect that performing cross attention between the patch tokens is not meaningful, since we cannot identify up-front what classes are present in the patches and we cannot trust that sharing patch-level information will improve attention for co-occurring classes as intended. Considering self attention, this would be tantamount to modelling the relevance of patch content in one image to the patch content of the second image, where we should be refining the relevance of a class to all patches, thereby giving the patches an opportunity to have high attention (resulting from the self attention dot product) for a co-occurring class (if it is necessary).

A suite of experiments is conducted where the following parameters are changed in the configuration between models:

1. The number of layers for which cross attention is applied. Values of 2, 3, 5, 10, and 11 are chosen (the transformer has 12 layers).

2. The specific layers for the cross attention are randomly chosen at the beginning of training and maintained for the whole training cycle. The layers chosen do not have to be consecutive nor do they have to be from the second half of the transformer. The models with 10 and 11 layers exclude the first and last layers, and the last layer respectively. Hence there is only one possible choice of layers for these configurations. The last layer is excluded since its information is directly used by the classification module and performing cross attention here could introduce

unnecessary noise. It is not clear if the first layer acts as an embedding layer (or something of the sort), so we have versions where we include or exclude it (i.e. the 11 and 10 layer models, respectively).

3. For the 2, 3, and 5 layer models, versions are included where the layers for which cross attention is applied are randomly chosen at the beginning of each batch during training. This will help ascertain whether fixing the specific layer choices at the beginning of training introduces some kind of learnt dependency that these layers expect the refinement from cross attention.

4. The inclusion or exclusion of patch token query vectors in cross attention. This is denoted by "incl" or "excl" in the model configuration, respectively. The mathematical formulation of the inclusion or exclusion of patch token query vectors is described in Equation (4.14), as part of the cross attention masking function, f — the inclusion of patch query vectors requires no manipulation since that is the default behaviour, but the exclusion of them is attained by setting the set of patch indices to be empty (i.e. set $P = \varnothing$).

The results are presented in Figure 5.22, for select co-occurring classes of interest and the overall performance — owing to the number of experiments conducts, results from similarly-configured models have been aggregated according to the maximum and average performance (this aggregates models with different random choices of layers, for the same number of layers). However, the complete results for the co-occurring classes of interest and the overall performance are included in Table A.3, which demonstrates the random nature of the choices of layers for the application of cross attention.

With respect to Figure 5.22, when considering the performance of models using 3 layers, the maximum and average performance either outperform or match the performance of other configurations. The only exception is the Baseball Glove class, where 10 and 11 layer models without patch token cross attention outperforms them, but the 3-layer models do still match the previous-best baseline performance. However, 10 and 11 layer configurations are outperformed to a far greater degree for other classes (i.e. Baseball Bat, Mouse, and Keyboard). Additionally, in just about all configurations, the MCTFormer+ baseline is outperformed, and competitive performance to the previous-best model is achieved — there are some cases where this is not the case (i.e. mainly the 10 and 11 layer runs, for the Baseball Bat, Mouse, Keyboard, and mIoU classes). Therefore, using too many layers (refer to the 10 and 11 layer models) is counterproductive, but using similar numbers of fewer layers achieves comparable performance. As a result of this sweep, we find that the use of 3 layers achieves the best general performance for classes of interest.

Additionally, as demonstrated in Table A.3, and by the relatively small differences between the maximum and average performance in Figure 5.22, the exact choice of layers is not important. Various choices result in better performance, but they are still comparable to the other collections of layers and the inclusion of early layers is still competitive. In the same vein, the "batch" performance generally performs worse than the maximum and the average results for given configurations of models, suggesting that constantly changing the layers to which cross attention is applied is actually a

hindrance. Therefore, it is unlikely that the model is learning any dependence on the specific layers for cross attention and, if it is, then it is proving more beneficial than not.

Finally, for a given number of layers, the performance between models including or excluding the patch tokens from cross attention is quite similar. Indeed, for particular combinations of layers, it is arbitrary whether the inclusion or exclusion will lead to better results. Therefore, it stands to reason that the collective combination of the specific layers and the inclusion/exclusion of patch tokens is what achieves high performance, and is not based on only one of these component parts.

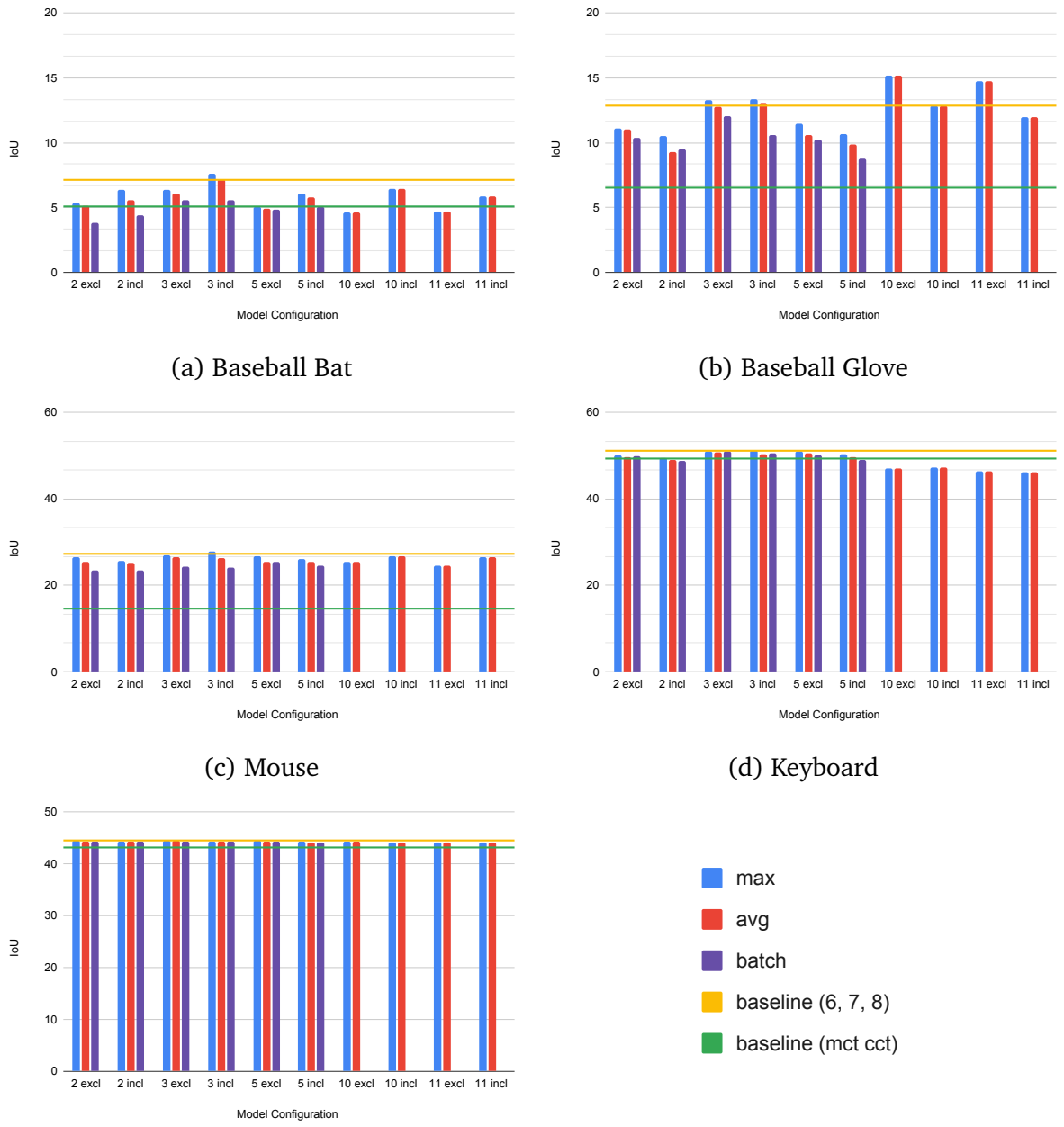
Specifically, the model trained without cross attention applied on the patch tokens, and for layers 3, 5, and 6, is the best-performing model from this suite, in terms of overall performance. It achieves competitive performance on co-occurring classes to the other configurations, but the higher overall performance means it is performing well on other non-co-occurring classes as well. Hence, we use it as one of our final configurations, presented in Section 4.5.

5.2.9 Third Branch

The previous sections in this chapter have all made use of the dual-branch framework. Section 5.2.6, Section 5.2.7, and Section 5.2.8 initially introduced CIA-AP, and refined and justified various design decisions, with success. CIA-AP introduced the core idea of only applying cross image attention between the primary and auxiliary branches when the primary image had a pair of co-occurring classes, where a pair of classes is co-occurring if their co-occurrence rate is above a chosen threshold. The auxiliary image is mined such that it has exactly one of the pair of co-occurring classes present in the auxiliary image.

A natural extension of this dual-branch framework is to introduce a third branch. Currently, the auxiliary image will overlap on only one of the co-occurring classes and will only refine the query vector one of the pair of classes, with no guarantees being made about which of the pair is chosen to overlap with the auxiliary image. Consequently, the remaining class still operates as it did for MCTFormer+, which Chapter 3 found to be inadequate. Additionally, we may inadvertently be introducing bias towards one of the classes since we only aim to improve attention for one of the classes. There is room for additional supervision to the remaining class: by sampling a second auxiliary image, which only overlaps on the remaining class of the pair, this auxiliary image can similarly improve the remaining class.

Similarly to Equation (4.12), let $\mathbf{Q}_p^l$, $\mathbf{Q}_{a_1}^l$, and $\mathbf{Q}_{a_2}^l$ be the learnt query matrices for a particular transformer layer, l, for the primary, and first and second auxiliary branches,



(a) Baseball Bat

(b) Baseball Glove

(c) Mouse

(d) Keyboard

(e) Overall Performance (mIoU)

(f) Legend

Figure 5.22: Graph comparing the aggregated absolute performance (maximum in blue and average in red) for various numbers of randomly-chosen layers, for cross attention that includes or excludes the patch query vectors (which is the model configuration). Each graph corresponds to a class of interest from COCO. The purple ("batch") columns correspond to the layers being randomly changed at the beginning of each batch, where other runs have their layers randomly chosen once. The green baseline is MCTFormer+ (with CCT, as per the original configuration), and the yellow baseline is the previous model configuration which performed the best (it performed cross attention on layers 6, 7, and 8). The 10-layer runs exclude the first and last layers from cross attention, and the 11-layer runs exclude the last layer — there is only one configuration of these layers, hence no "batch" run exists for them.

respectively. Then the query augmentation is the mean of these matrices:

$$
\begin{aligned}
h(\mathbf{Q}_p^l, s_1) &\leftarrow \frac{1}{2}\left(h(\mathbf{Q}_p^l, s_1) + h(\mathbf{Q}_{a_1}^l, s_1)\right) \\
h(\mathbf{Q}_p^l, s_2) &\leftarrow \frac{1}{2}\left(h(\mathbf{Q}_p^l, s_2) + h(\mathbf{Q}_{a_2}^l, s_2)\right) \\
h(\mathbf{Q}_p^l, P) &\leftarrow \frac{1}{3}\left(h(\mathbf{Q}_p^l, P) + h(\mathbf{Q}_{a_1}^l, P) + h(\mathbf{Q}_{a_2}^l, P)\right),
\end{aligned}
\tag{5.5}
$$

where $\mathbf{Q}_{\{p,a\}}^l \in \mathbb{R}^{(C+N^2)\times d}$, C is the number of class tokens, N^2 is the number of patch tokens, and d is the dimensionality of the tokens. h is a function to mask the specified query vectors to operate on — it is identical to f in Equation (4.14), except that it operates directly on the set of indices passed to it without augmenting it: $h(\mathbf{Q}, s) = \mathbf{Q}|s$. s_1 and s_2 are sets containing the indices of the class in the co-occurring pair present in the respective auxiliary image and which overlaps with the primary image. $P = \{\varnothing, \{C, C+1, \cdots N^2 - 1\}\}$ is the set of patch indices, which can optionally be set to the empty set (for example, if the patch token's query vectors are chosen to be excluded from the cross attention).

Equation (5.5) handles the averaging of class query vectors between the primary and auxiliary branches separately, as well as the averaging of the patch query vectors across all three branches separately. Due to the nature of mining the auxiliary images, s_1 and s_2 are mutually exclusive (i.e. a co-occurring class which overlaps between the primary and first auxiliary images will not overlap between the primary and second auxiliary images). Hence, the averaging of class query vectors will happen between the primary branch and precisely one auxiliary branch per query vector, and so the average is taken over two branches. Since g operates on the exact set of indices passed to it, the averaging of class query vectors in the first two parts of Equation (5.5) does not operate on the patch query vectors in any way. Therefore, the patch query vectors are averaged across all three branches.

Figure 5.23 presents the complete results under the triple-branch scheme, with comparisons to two previous high-performing dual-branch models. Due to the number of experiments conducted, we summarise the information with the maximum and average performance across the runs. Table A.4 presents the complete set of results for the classes of interest.

Focusing on the blue columns, which correspond to the pseudo label quality obtained using the final checkpoint after the usual 45 epochs of training, the results are outperformed by both baselines. Even taking the best class score across the models does not outperform the dual-branch baselines, despite the extra cross image attention and supervision.

We suspect the main reason for this underwhelming performance is overfitting. Under the CIA-AP scheme, the auxiliary branches will always perform classification training, even when there is no co-occurring class pair overlap with the primary image. That is, when the primary image has a co-occurring pair of classes, then the two auxiliary images are specially mined, cross attention takes place, and all three branches perform classification training. When the primary image does not have a pair of co-occurring

classes, then random mining for the two auxiliary images is performed instead and no cross attention occurs (since all the models use "turnoff" mode for cross attention), but all three branches still perform classification training on their respective images. Since the branches use shared weights, the triple-branch framework will perform 50% more training than the dual-branch framework.

Hence, we perform the same pseudo label evaluation using an earlier checkpoint, before overfitting has had a chance to occur. We choose checkpoint 25 (instead of the full 45) — this is supported by the training and validation curves in Figure A.1, which shows that at epoch 25, the validation loss had not started increasing again. Incidentally, Figure A.1 also supports the intuition that overfitting has occurred by Epoch 45.

Incorporating the early-stopping results in the red columns in Figure 5.23, it is evident that for all classes of interest except the mouse, performance improves over the 45 epoch version of the pseudo labels. However, the results are still largely underwhelming, with only the peak performance of Baseball Glove and Knife outperforming the dual-branch baselines.

Consequently, we discontinue the third-branch idea, since it obtains equal or worse performance than the dual-branch models.

Finally, this phenomenon of repeated training on the same set of shared weights across the branches also occurs in the dual-branch framework under CIA-AP. It might also run the risk of overfitting on the full 45 epochs of training. However, Section 5.1 finds, using Figure 5.2, that no overfitting occurs with the dual-branch framework. Therefore, the overfitting incident is isolated to the triple-branch case.

5.3 Inference

Inference using the dual-branch framework is an important stage in generating pseudo labels since we now need to utilise the cross image context stored in the transformer's weights.

This section discusses two main ideas relating to inference. Firstly, Section 5.3.1 presents the inference-time auxiliary image mining and cross image attention strategy that is most effective. Secondly, Section 5.3.2 resolves ambiguity regarding the layers to use for extracting the attention maps from the transformer layers.

5.3.1 Cross Image Attention Strategy

The early cross image attention schemes before CIA-AP was introduced in Section 5.2.6 — that is, Conventional Cross Image Attention, Explicit Cross Image Attention (both from Section 5.2.1), Single Layer Explicit Cross Image Attention (Section 5.2.3), and Multi-Layer Cross Image Attention (Section 5.2.4) — used the "same" inference mining strategy, since Section 5.2.1 empirically found it preferable to other inference mining strategies. The "same" mining strategy uses the primary image as the auxiliary image as well. These cross attention schemes all swapped query matrices between the two

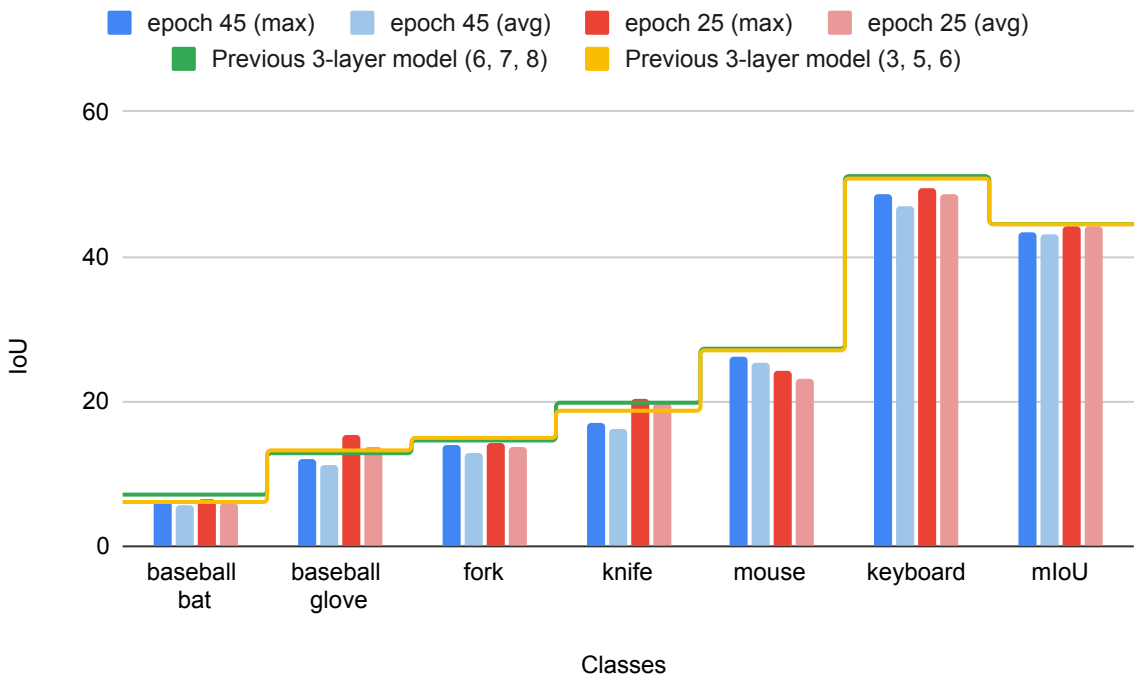


Figure 5.23: Graph showing the performance across classes of interest (from COCO) for the triple-branch version of our framework. Aggregated values (maximum in the darker shade, and average in the lighter shade) are plotted for versions of the model evaluated on the final checkpoint (at Epoch 45) in blue, and on an earlier checkpoint (at Epoch 25) in red. The aggregation is performed over models with varying layer choices, the inclusion and exclusion of patch token query vectors during cross attention, and the inclusion and exclusion of contrastive learning. Two high-performing dual-branch model baselines are included for comparison in green and yellow.

branches in their cross attention. Consequently, during inference, using the same image as primary and auxiliary images is tantamount to not performing cross attention between the branches at all: each branch has the same class tokens (all initialised using the same DeiT-S class token — this is inherited from MCTFormer+) and the same patch tokens (since they have the same image) passing through the same transformer weights (since the branch weights are shared). Therefore, the cross attention swapping Query matrices between the two branches is just replacing numbers with the same numbers, leading to the attention behaving as if no information exchange had taken place.

Section 5.2.6 introduced the concept of Query augmentation with CIA-AP, which let the resultant Query matrix be a combination of both the primary and auxiliary input Query matrices. Using the "same" inference mining strategy under this condition is detrimental.

Consider the Softmax activation function, used in self-attention in Equation (4.1):

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \tag{5.6}$$

where $\vec{z}$ is a row from the scaled dot product from Equation (4.1), $\mathbf{Z} = \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}$.

Therefore, if we refine some components of $\mathbf{Z}$ with CIA-AP, some exponentials in Equation (5.6) are raised to a larger power, making the distribution more peaked for these components. For example, under the "sum" augmentation method, this component would be doubled.

The interpretation of this in terms of attention over the relevant class (that the row from $\mathbf{Z}$ corresponds to — that is, one of the orange, red, or green sets of attention values in Figure 4.6b), is that the "hot" areas get "hotter" and the "warm" and "cool" areas get "colder". This means that the overall attention applied to this class will get smaller, making it more likely another class will have higher attention for this region, thus causing its pixels to be misclassified. Therefore, the coverage of the class (and, by extension, co-occurring classes) will shrink, reducing IoU performance.

In order to avoid this, we adopt the same inference strategy that we had with cross attention schemes prior to CIA-AP: do not perform cross attention. We simply pass the primary image through the network without an auxiliary image (i.e. the auxiliary branch remains idle). Evidently, this strategy leads to favourable results (Section 5.2.6, Section 5.2.7, Section 5.2.8).

5.3.2 Layers for Extracting Attention Maps

Our dual-branch framework is built on MCTFormer+. Seeing as we are performing segmentation on the primary image (i.e. the end goal), we only use the attention maps from the primary branch to obtain estimated object regions. Therefore, we utilise the same method that MCTFormer+ does, and which Section 4.3.3 explains.

However, there is ambiguity in the choice for the value K in Equation (4.7), where K is the number of transformer layers (starting from the last layer) from which attention

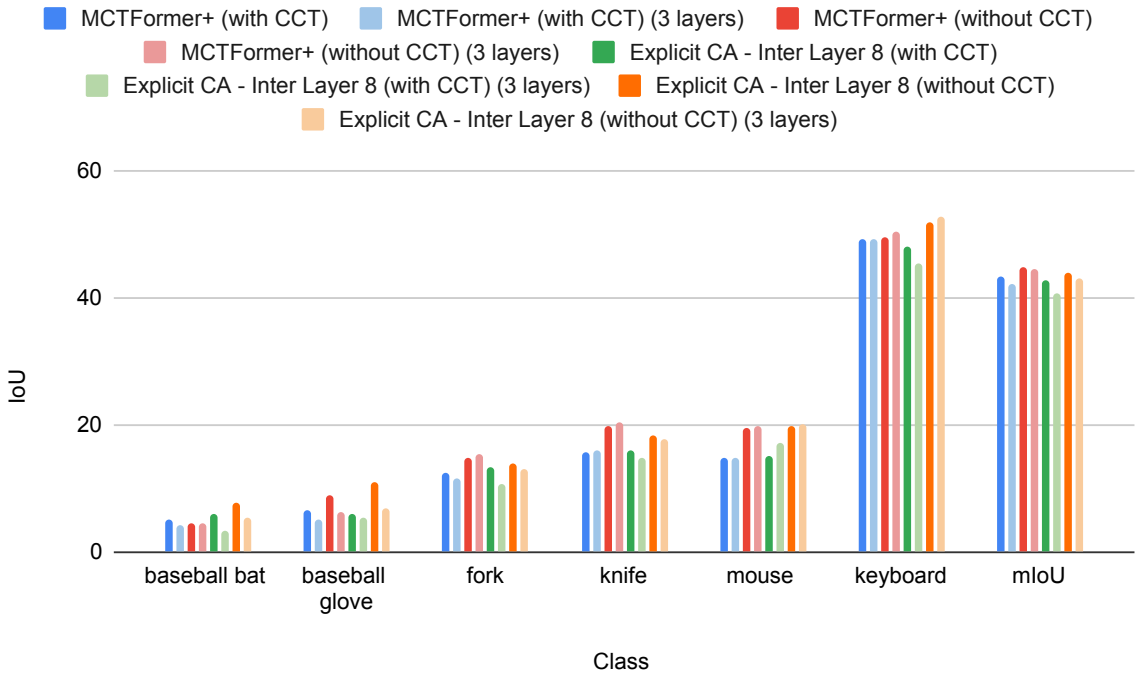


Figure 5.24: Graph comparing performance of various models when extracting attention maps from all the layers versus only the last 3, on COCO. The colours represent groups of runs — presented are, both with and without contrastive learning (CCT), the baseline MCTformer+ and the model we used with Single Layer Explicit Cross Image Attention applied to Layer 8, using the Intersection overlap task. The darker shade of a colour corresponds to inference on all the transformer layers, and the lighter shade to inference on the last 3 transformer layers.

maps are extracted and averaged together. Xu *et al.* [2024] report that the last 3 layers should be used, however, their linked repository implements it with all 12 transformer layers.

Consequently, we test it ourselves, with results presented in Figure 5.24, for two models, both with and without contrastive learning, namely MCTFormer+ and the Explicit Cross Image Attention model where cross attention is applied to Layer 8 and the intersection overlap task is used. For the vast majority of the runs, the 12-layer version outperforms the 3-layer version (i.e. the darker shade column is higher than the lighter shade column), with the exception of MCTFormer+ (with CCT) for *fork*, *knife*, *mouse*, and *keyboard*, and the Explicit Cross Attention model (without CCT) for *keyboard*.

Therefore, we use all the transformer layers when extracting and averaging the attention maps for inference.

5.4 Summary

This chapter presented our investigation into the application of Cross Image Attention for WSSS, specifically aimed at addressing the issue of class co-occurrence (Chapter 3). Our final dual-branch framework applies cross attention by augmenting the primary branch's Query matrix with that of the auxiliary branch, derived from an auxiliary image selected to share only one of the co-occurring classes with the primary image. This ensures that the auxiliary image provides a pure and reliable query vector for the ambiguous class pair.

The ablation studies conducted in this chapter extensively examine various design choices related to the application of cross attention. This includes our proposed CIA-AP mechanism, which selectively targets the co-occurring classes by enhancing their query vectors to improve attention for difficult co-occurring pairs, such as Baseball Bat and Baseball Glove, Mouse and Keyboard, and Knife and Fork. The results demonstrate that using the mean augmentation method is the most effective choice, providing a simple yet powerful way to refine the query vectors and improve segmentation performance. Additionally, we find that the specific choice of layers for applying cross attention is not particularly critical, as effective performance can be achieved across a range of layer groupings.

Furthermore, we compare our cross attention mechanism against conventional alternatives, showing that our approach more effectively handles co-occurrence by transferring relevant information between the branches. The shared-weight architecture of our dual-branch model proves essential, ensuring consistent generation of query vectors and enhancing the compatibility of cross attention.

Overall, while our method does not yet surpass the current state-of-the-art, it effectively mitigates the detrimental effects of class co-occurrence, providing valuable insights into how cross image attention can be better leveraged for weakly supervised semantic segmentation.

Chapter 6

CCT Prime: Class Token Alignment with Contrastive Learning

Contrastive learning is a form of metric learning aimed at learning embeddings where similar input instances are mapped to similar embeddings, while dissimilar instances are mapped to embeddings that are further apart. The core idea is that if the distance between embeddings accurately reflects the similarity between the corresponding input instances, then the embeddings can be used to infer relationships and make deductions about the relatedness of various objects based solely on their representations. In the context of weakly supervised semantic segmentation, contrastive learning can help enhance class differentiation by ensuring that embeddings corresponding to different classes are well-separated.

This chapter first provides a general overview of contrastive learning in Section 6.1 before discussing high performing work using contrastive learning in Section 6.2.

Chapter 4 demonstrated the effectiveness of cross image attention for addressing class co-occurrence in weakly supervised semantic segmentation, and was supported by the ablation studies conducted in Chapter 5. Building on that foundation, this chapter investigates the application of contrastive learning to further enhance performance. Earlier discussions in Section 4.3 highlighted the contrastive learning module from MCTFormer+, CCT, [Xu *et al.* 2024], which successfully improved performance on PASCAL VOC 2012 but failed to achieve similar success on the more challenging MS COCO 2014 dataset, where increased class co-occurrence likely confuses the CCT mechanism.

To address this, Section 6.3.1 initially presents the inherited CCT module from MCTFormer+ and Section 6.3.2 conducts a detailed behavioural analysis to establish its strengths and weaknesses. Section 6.3.3 also provides initial performance evaluations using earlier cross attention techniques. Following this, we propose an additional contrastive learning module, CCT Prime, designed to complement CCT by aligning representations between branches (Section 6.3.4). While CCT focuses on learning relationships among class tokens within a single branch, CCT Prime aims to harmonise representations across branches to enhance performance.

Finally, Section 6.3.5 integrates CCT Prime with the CIA-AP mechanism introduced in Section 5.2.6, presenting a unified framework that leverages both cross attention and contrastive learning to effectively mitigate the co-occurrence problem.

6.1 Background

Contrastive learning is a popular self-supervised learning technique designed to learn robust representations by bringing similar samples closer together and pushing dissimilar ones further apart. Commonly used approaches include contrastive loss [Hadsell *et al.* 2006], where positive pairs (similar samples) are encouraged to have high similarity scores while negative pairs (dissimilar samples) are pushed apart. Another frequently used approach is the triplet loss [Schroff *et al.* 2015], which minimises the distance between an anchor and a positive sample while maximising the distance between the anchor and a negative sample. These methods have been instrumental in various domains, including visual representation learning, metric learning, and natural language processing.

While numerous contrastive learning frameworks exist, our work primarily focuses on a method inherited from MCTFormer+ [Xu *et al.* 2024], which employs a contrastive learning mechanism specifically designed to enhance class token separation within a transformer-based architecture. Importantly, the underlying mechanism relies on a cross-entropy loss, which is a simpler yet effective alternative to traditional contrastive losses and triplet losses.

6.1.1 Cross Entropy Loss

Cross Entropy Loss is a widely used objective function for training classification models, measuring the discrepancy between predicted probability distributions and true label distributions, and using this to encourage the model to produce high probabilities for the correct classes. For a single data point with a true label y and a predicted probability distribution $\hat{y}$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CrossEntropy}} = -\sum_{i=1}^{C} y_i \log \hat{y}_i, \tag{6.1}$$

where C is the number of classes, $y_i \in \{0, 1\}$ represents the ground truth label (1 for the correct class and 0 otherwise), and $\hat{y}_i$ is the predicted probability of the i-th class.

While Cross Entropy Loss is commonly associated with classification tasks, it also plays a significant role in contrastive learning under certain frameworks. When applied in a contrastive learning setting, the loss function is modified to distinguish positive pairs from negative pairs by treating the problem as a binary classification task:

$$\mathcal{L}_{\text{CrossEntropy}}(i, j) = -(y_{ij} \log \hat{y}_{ij} + (1 - y_{ij}) \log(1 - \hat{y}_{ij})), \tag{6.2}$$

where $y_{ij} \in \{0, 1\}$ is 1 if embeddings i and j should be similar and 0 if they should not. $\hat{y}_{ij}$ is the predicted similarity score between the embeddings.

In this work, CCT rephrases the contrastive learning aspect as a classification task. In Equation (6.4), we take the computed similarity matrix, $\mathbf{S}$, for a layer in the transformer and encourage it to converge to the identity matrix. Cross Entropy Loss is used to encourage the model to "classify" each element of the similarity matrix according to its corresponding element in the identity matrix. The construction of $\mathbf{S}$ is done such that it represents the degree of similarity between all the class tokens, and the classification problem then follows.

6.2 Related Work

In this section, we review previous work related to metric learning, focusing on approaches that leverage learnt distance metrics and contrastive learning for weakly supervised semantic segmentation. Section 6.2.1 discusses methods that employ a learned distance metric to enhance feature representation. Section 6.2.2 covers recent state-of-the-art techniques that incorporate contrastive learning, a powerful paradigm for representation learning that aims to distinguish between similar and dissimilar samples, particularly in scenarios where supervision is limited or noisy.

6.2.1 Metric Learning (Distance Measures)

Metric Learning is discussed in Section 6.1, where a Distance Measure is a function used to determine the similarity between two embedding vectors: similar embeddings are "close" together and dissimilar embeddings are "far" apart.

Generally, fixed distance measures, such as the Cosine Similarity or L_2 Norm, are used. Cosine Similarity measures the cosine of the angle between the vectors. This is based on the idea that similar vectors tend to be more colinear, whilst dissimilar vectors tend to point in opposite directions. The main advantage of Cosine Similarity is that it is independent of the scale of the vector components. The L_2 Norm takes the Euclidean Distance between two vectors. This has the drawback of being skewed by different scales amongst the vector components. To fix this, the embeddings are generally normalised first, resulting in the L_2-normalised Euclidean Distance as the distance measure, which is mathematically equivalent to the Cosine Similarity: both are monotonic functions and yield the same ordering.

Additionally, learnt distance measures have also been used. Most notably, Chen *et al.* [2017b] introduced a Quadruplet Loss, where an anchor, positive, and two negative images are used to calculate the loss. However, the distance function used is replaced by a trainable neural network, which acts as a learnt non-linear distance function. The Quadruplet Loss aims to reduce intra-class variation, whilst increasing inter-class variation, thereby improving generalisability from the training to the test set. However, reimplementations struggle to get a similar performance improvement. Musgrave *et al.* [2020] find that the improved results from metric learning papers (in the prior 4 years) are overestimated and marginal at best, often stemming from unfair comparisons and leakage of the test set into the training regime.

Consequently, it is preferable to use a fixed distance measure, as opposed to a learnt one like Quadruplet Loss.

6.2.2 Contrastive Learning Approaches

One of the most recent and high-performing models, SeCo [Yang *et al.* 2024], stands out for its general applicability, as it can be applied to both class-background and class-class co-occurrence. The model segments each image into multiple patches (a common practice when using Vision Transformers) and assigns a category tag to each patch. This tag represents an estimate of the most prominent class within the patch, derived from CAMs produced by an auxiliary classification head incorporated into the model. Contrastive learning is then applied between the various patch embeddings and between the patch embeddings and global class embeddings, with one global embedding per class in the dataset. However, the authors acknowledge that deriving category tags from initial CAMs is a noisy process and attempt to address this issue using a confidence-thresholding system. It is this aspect that is unattractive to us: since contrastive learning relies on the imperfect category tags as labels, errors are compounded, resulting in a non-stationary learning problem.

An additional competitive method, ToCo [Ru *et al.* 2023], enhances contrastive learning in Vision Transformers (ViTs) through two modules: Patch-Token Contrast (PTC) and Class-Token Contrast (CTC). To estimate patch relations, a Class Activation Map (CAM) is generated from an intermediate ViT layer using an auxiliary classification head. A dual-threshold system identifies reliable foreground, background, and uncertain regions. PTC, the primary contribution, maximises or minimises the absolute cosine similarity between positive and negative patch pairs, ensuring diversity by mitigating issues where negative similarities could misleadingly indicate strong correlations. CTC addresses uncertainty in CAMs by aligning global class tokens (representing the entire image) with local class tokens derived from randomly cropped uncertain regions. To counter potential errors from missing foreground objects in crops, additional background regions are sampled, and their tokens are contrasted against the global token. This approach ensures robust contrastive learning by clearly distinguishing positive and negative pairs through careful token comparison.

MCTFormer+ [Xu *et al.* 2024] is quite competitive with SeCo. They introduce a contrastive learning module, called Contrastive Class Tokens (CCT), which they use to help disentangle their multiple class tokens, where each class token corresponds to a class in the dataset, with reasonable success. Seeing as MCTFormer+ is our base model, CCT is discussed in-depth in Section 6.3.1.

6.3 Contrastive Learning in our Dual-Branch Framework

This work is primarily built on MCTFormer+ [Xu *et al.* 2024], which is one of the state-of-the-art works in the area of weakly supervised semantic segmentation. Section 4.3 discussed the base models in-depth and presented their core mathematical concepts. Section 4.3.2 further discussed the two major improvements MCTFormer+ made over

its predecessor, MCTFormer [Xu *et al.* 2022], namely Global Weighted Ranking Pooling (GWRP) and Contrastive Class Tokens (CCT). The latter is the contrastive learning addition they made, but Section 4.3.2 only introduced the concept.

Consequently, Section 6.3.1 reiterates the conceptual foundation for MCTFormer+'s contrastive learning and additionally presents its mathematics. Section 6.3.2 then investigates the actual behavioural impact CCT has on the class tokens, before Section 6.3.3 presents effect CCT has on the performance of both the baseline model and initial dual-branch models. Finally, Section 6.3.4 develops a principled addition to CCT, called CCT Prime, to account for the cross image context present in the dual-branch framework, and Section 6.3.5 ultimately fine-tunes CCT and CCT Prime to the final version of the cross image attention, CIA-AP.

6.3.1 Contrastive Class Tokens (CCT)

MCTFormer+ was initially presented in Section 4.3.2, where GRWP, the first of its two improvements over its predecessor, was presented in-depth. Contrastive Class Tokens (CCT) was the second major contribution, which this section presents in-depth, before Section 6.3.2 determines its behavioural impact.

CCT is a contrastive learning module that operates on the class tokens in each of the transformer's layers. The underlying premise of CCT is that the dataset's classes are distinct and have no hierarchical structure, therefore the class tokens should be completely disentangled, since these tokens encode each classes' characteristics. Xu *et al.* [2024] had initially noticed that different class-specific localisation maps had considerable overlap and attributed this to two factors: the classification loss only has a substantial effect for the final layers of the transformer, on which the classification module is most dependent; and the nature of the classification loss, being a multi-label one-versus-all loss, does not enforce distance between the class tokens.

Consequently, contrastive learning is used to encourage the class tokens to be "distant" from each other. This takes the form of computing the pairwise similarity between class tokens by multiplying the class tokens at a particular layer with themselves:

$$\mathbf{S}_l = \mathbf{T}_l \cdot \mathbf{T}_l^T, \tag{6.3}$$

where $\mathbf{S}_l \in \mathbb{R}^{C \times C}$ is the pairwise similarity matrix, $\mathbf{T}_l \in \mathbb{R}^{C \times d}$ is the class token matrix, and l is the specific layer in the transformer from which the tokens are extracted. C is the number of classes and d is the dimensions of the class tokens. $\cdot$ denotes matrix multiplication.

Each element in the similarity matrix, $\mathbf{S}_l$, represents the degree of alignment between the two input class tokens. The main diagonal models the similarity of a class token to itself and, therefore, should be perfectly aligned. All the other matrix elements model the similarity between different class tokens, and, therefore, should be orthogonally aligned. Hence, contrastive learning is used to encourage this matrix to be similar to an identity matrix, by means of a Cross Entropy Loss (introduced in Section 6.1.1), where

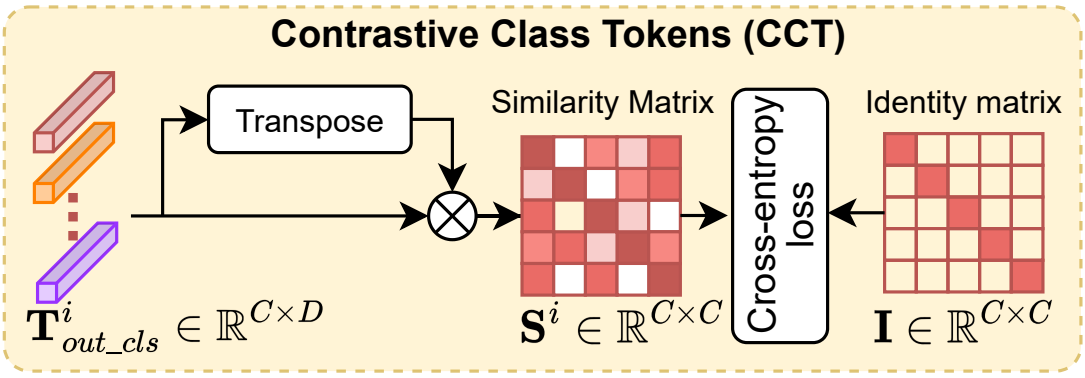


Figure 6.1: Contrastive Class Token (CCT) Module (from Xu *et al.* [2024]). For a given transformer layer, i in this case, the C class tokens of dimension D from this layer, represented by $\mathbf{T}^i_{\text{out_cls}} \in \mathbb{R}^{C\times D}$ are multiplied by themselves to compute the pairwise similarity between them, collectively represented by $\mathbf{S} \in \mathbb{R}^{C\times C}$. The main diagonal is the correlation of a class token to itself and so should be perfect. All other elements measure the correlation between different class tokens (determined by the row and column indices), and so should be completely different. Hence, $\mathbf{S}$ should ideally take the form of an identity matrix, $\mathbf{I}$, which is encouraged through a Cross Entropy Loss.

the contrastive loss is averaged over all the transformer layers:

$$\mathcal{L}_{\text{CCT}} = \frac{1}{L}\sum_{i=0}^{L-1} \text{CrossEntropyLoss}(\mathbf{S}_i, \mathbf{I}), \tag{6.4}$$

where i is the index of the transformer layer and L is the number of transformer layers in total. This CCT concept is depicted graphically in Figure 6.1.

6.3.2 CCT Behaviour

Xu *et al.* [2024] introduced CCT and showed its effectiveness at increasing the weakly supervised segmentation performance. They performed several quantitative studies, showing that applying it to all layers in the transformer is the most effective strategy and, via an ablation study on VOC, showed that it was a leading contributor to their increased performance.

Mathematically, Equation (6.3) takes the matrix product of the class tokens with its transpose to get the similarity matrix, essentially taking the dot product between all pairs of class tokens (including with themselves), where the individual matrix elements correspond to the tokens represented by the row and column indices. By encouraging this similarity matrix to be an identity matrix, CCT attempts to enforce the condition of orthonormality on the class tokens — the main diagonal (product of a class token with itself) is driven to be 1, and all other elements (products between different class tokens) are driven to be 0. Xu *et al.* [2024] apply masking to CCT such that it only operates on tokens for classes present in the image. Therefore, for a particular image, only the present class tokens are encouraged to be orthonormal.

However, the authors do not include an analysis of what CCT actually does to the class tokens in practice. This is important for our work since we are sharing information about class tokens between the two branches via a cross attention scheme — by understanding how CCT impacts the class token behaviour, we can decide if CCT is the correct form of contrastive learning and, if not, how to adapt it to our case.

To this end, we extract the class tokens from each layer of a trained MCTFormer+ model when we perform inference on particular images. To compare the class tokens and ascertain their similarity, we compute the Cosine Similarity between them:

$$S_C(\mathbf{T}_i^l, \mathbf{T}_j^l) = \frac{\mathbf{T}_i^l \cdot \mathbf{T}_j^l}{||\mathbf{T}_i^l||_2||\mathbf{T}_j^l||_2}, \tag{6.5}$$

where $S_C \in [-1, 1]$ is the cosine similarity between the i^{th} and j^{th} class tokens in layer l, and $\mathbf{T}^l \in \mathbb{R}^{C \times d}$ is the class token matrix for a particular layer. $\cdot$ denotes the dot product between vectors.

The Cosine Similarity measures the correlation between the input vectors' directions and returns values in the range $[-1, 1]$, where -1 means the vectors point in opposite directions, 0 means they are orthogonal, and 1 means they point in the same direction.

Figure 6.2 depicts some example images and the cosine similarities between their class tokens taken from the final transformer layer. The matrices in the final column show the cosine similarity between different class tokens. Since the cosine similarity is symmetrical, the matrix is symmetrical, so we only include the lower triangular matrix. As expected, the main diagonal is 1 since each class token points in the same direction as itself.

An important detail to consider is that CCT applies masking to the class tokens involved in contrastive learning, ensuring that only tokens corresponding to classes present in the image are actively used. Consequently, CCT does not operate on the majority of class tokens simply because their classes are absent, which means these class tokens remain largely aligned by their initialisation (which is to the same class token). Each class token is still operated on by the transformer's attention and they evolve as they pass through the transformer layers, thereby gaining slight variations from each other. Therefore, the cosine similarity between pairs of class tokens for absent classes is close to 1, but may not be exactly 1 due to this natural variation. Similarly, the class tokens for class present in the image are also aligned in such a manner — close to 1, but not exactly 1. However, the cosine similarity between a class token present in the image and one not present in the image are nearly-oppositely — these values approach -1, but are not exactly -1.

This pattern suggests a specific directional structure: each class token has its own distinct direction, but tokens corresponding to present classes generally align in one overall direction, while tokens corresponding to absent classes align in another, nearly opposite direction. This phenomenon is illustrated in Figure 6.3. The directions that are orthogonal to these two main directions tend to represent class tokens for which the model is uncertain are present in the image. Interestingly, some class tokens that are truly absent still point in the "present" direction (i.e. are aligned with other "present"

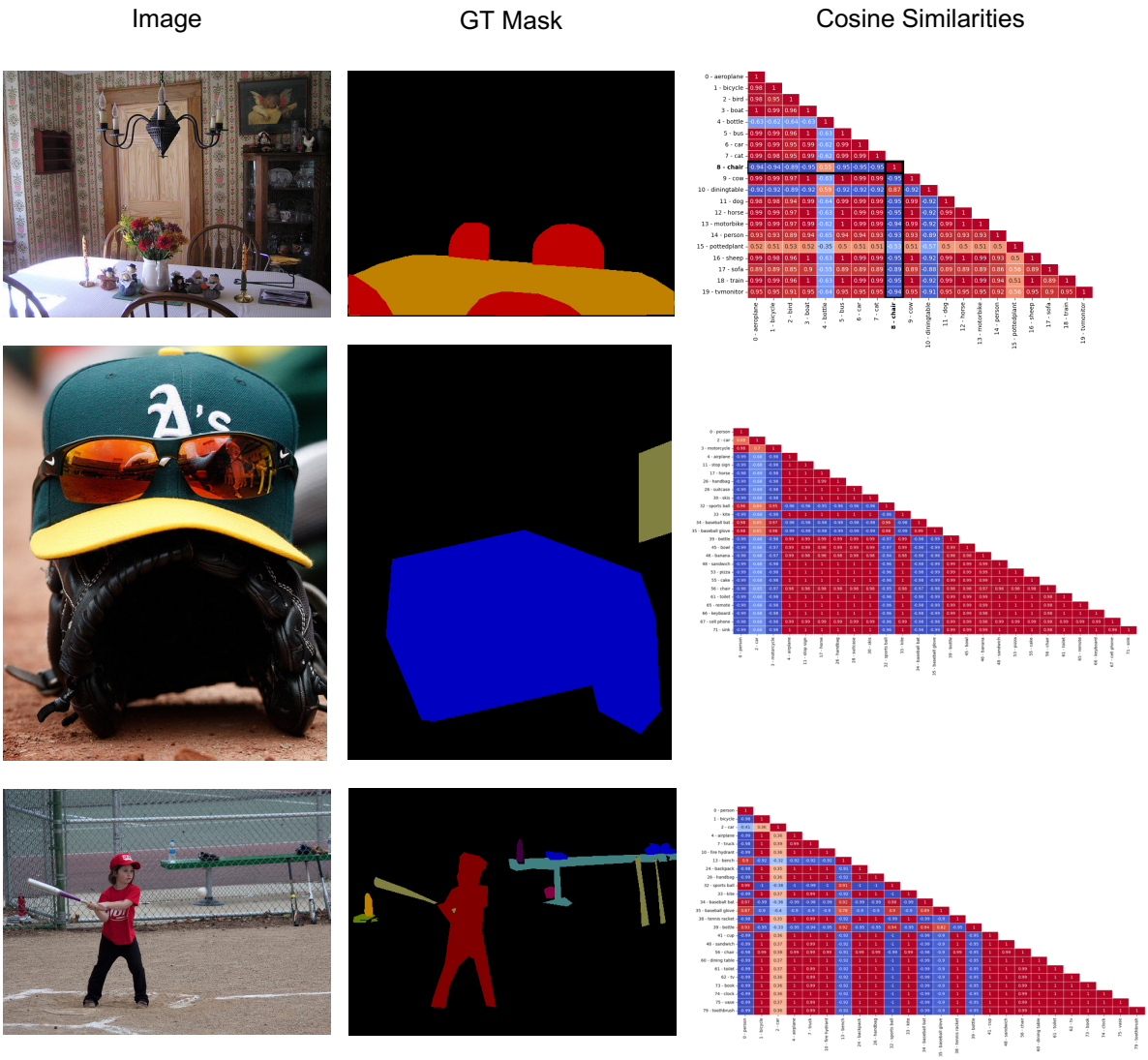


Figure 6.2: Figure showing the cosine similarity between class tokens from the final layer of the transformer in MCTFormer+ for various images. Each row corresponds to an image, with the first one taken from VOC and the last two taken from COCO. The first column is the original image, the second is the ground truth pixel mask, and the final column presents the lower triangular matrix of cosine similarity between the class token represented by each row and column (the complete matrix is symmetrical). The cosine similarity ranges from -1 (blue) to 1 (red).

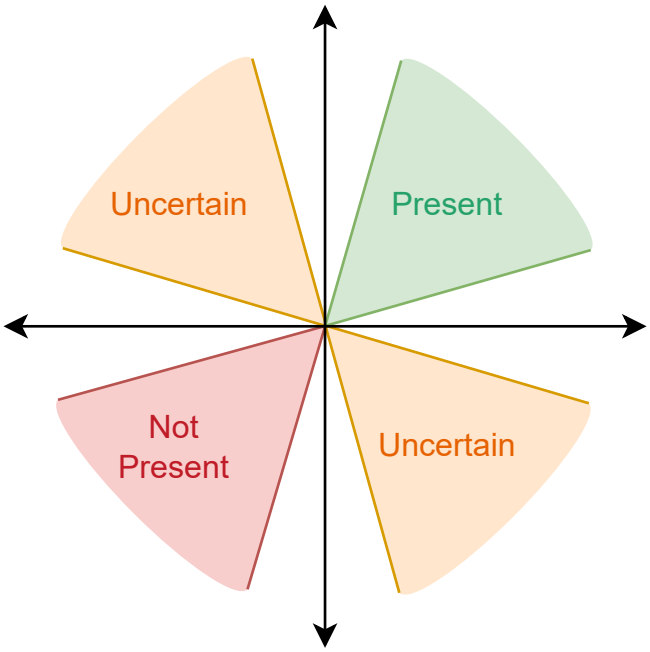


Figure 6.3: Figure representing the behavioural impact of CCT. Using Cosine Similarity to determine the directional similarity between class tokens, each class token points in a unique direction. Those that are present in the image generally point in the same direction and those that are not also generally point in the same direction that is different to the "present" direction. Class tokens that the model is uncertain are present, generally point in orthogonal directions to the "present" and "absent" directions.

class tokens), indicating instances where the model has misclassified those classes as being present.

This contradicts the mathematical intention behind CCT, which was to enforce orthonormality over the class tokens present in an image. In practice, the "conic" directional structure in Figure 6.3 is encouraged by the classifier, which has a stronger influence on the class token's structure than CCT. Nevertheless, CCT still plays a role in encouraging the class tokens to point in distinct directions (hence the values close to, but not exactly equal to, 1 and -1). When the classifier is uncertain about the existence of a class in the image, it has reduced influence over the class token structure and CCT has a stronger influence. Hence, the intended effect of CCT is observed, with those class tokens becoming orthogonal to the others.

6.3.3 CCT Initial Performance

Section 6.3.2 established the behavioural impact CCT has on the class tokens within MCTFormer+ and found that it imposed directional structure on them. However, it is unclear how this will impact class tokens from two different images, especially when information is exchanged between them. Therefore, we need to establish how CCT performs in the dual-branch models out-of-the-box.

Consequently, we take several high-performing (but early-stage) models with Cross Image Attention, and compare them when run without CCT. Specifically, we use models trained with Single Layer Cross Image Attention during Section 5.2.3. These models' performances are presented in Figure 6.4 and Figure 6.5 for VOC and COCO, respectively, along with baseline comparisons with and without CCT.

Figure 6.4 shows a slight increase in the overall baseline performance when including CCT for VOC, a result which corroborates Xu *et al.* [2024]. However, the degree of improvement is less than reported (0.24% compared to the reported 2.9%). Simply looking at the graph, it is unclear whether CCT helps or hinders the dual-branch models on VOC, since the Intersection Layer 8 model has worse results with CCT, but the Union Layer 11 model has better results when including CCT, for all classes of interest. The numerical results for the VOC classes of interest are presented in Table A.5.

Figure 6.5a shows that excluding CCT outperforms its inclusion for MCTFormer+, which is contrary to the finding about the effectiveness of CCT from Xu *et al.* [2024]. However, the reported results only consider VOC in their ablation study, not COCO, so there is no way to establish whether or not their work on CCT extends to COCO. We conjecture that the reason is the increased co-occurrence present in COCO interfering with the "disentanglement" mechanism of CCT. An additional reason could be related to the determination of the actual class token behaviour induced by CCT in Section 6.3.2, which shows that class tokens do not necessarily adopt orthogonality, but rather point in general, but still distinct, directions based on whether the class is present or not. This could lead to agreement issues between our two branches, since there is no enforcement that they have to agree on which direction is "present".

Additionally, Figure 6.5b shows that for the majority of model configurations, excluding CCT leads to performance improvements, with the exception of Union Layer 7 and Union Layer 11 for both the fork class and the overall, mIoU performance. This suggests the initial cross attention methods (the ones prior to CIA-AP, which is introduced in Section 5.2.6) are not able to overcome the increased co-occurrence in COCO, something which is supported in Section 5.2.5, where we find these initial cross attention methods to be inadequate. The numerical results for the COCO classes of interest are presented in Table A.6.

6.3.4 CCT Prime

Section 6.3.3 shows that in its current form, CCT is inadequate to help disambiguate co-occurring classes. Indeed, it seems to perform worse the higher the co-occurrence is (hence its performance on COCO is so much worse than on VOC).

Examining the cosine similarities in the third row of Figure 6.6, which are taken from MCTFormer+, it is evident that substantial misclassification occurs for co-occurring classes. In the first example shown, only the Person and Baseball Bat classes are present, and so we only expect those classes to point in the "present" direction. However, the Sports Ball and Baseball Glove classes are also identified as being present in the image, which is a clear mistake. There is a higher degree of dependence between these combinations of classes — even though only the Baseball Bat and Baseball Glove classes are

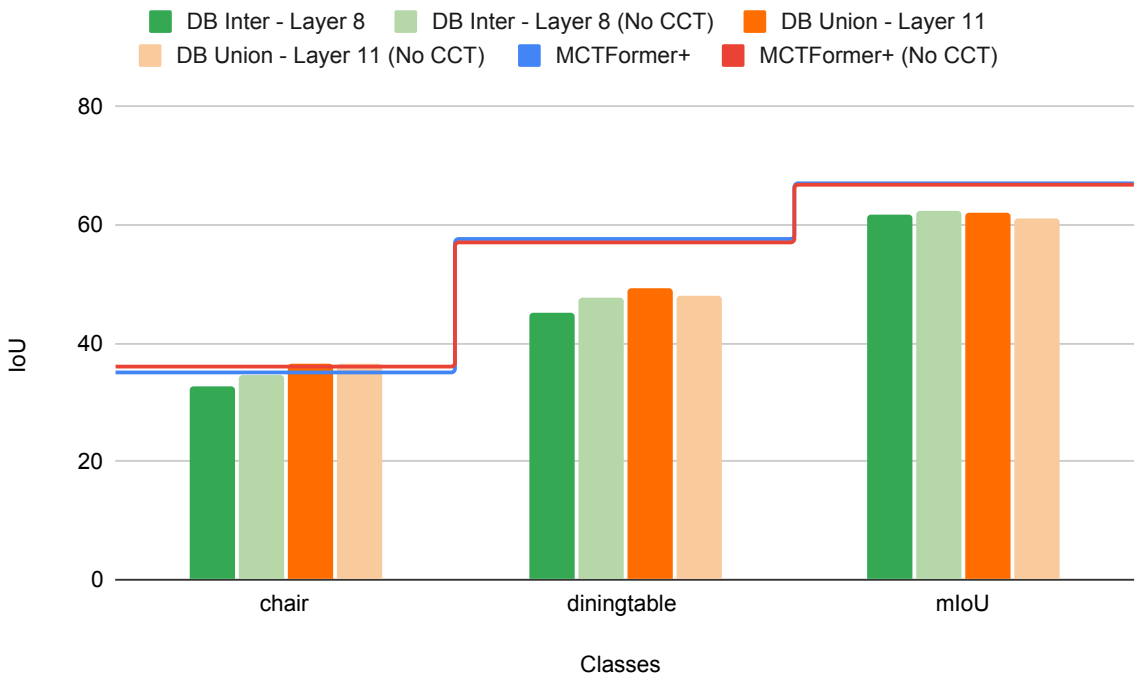
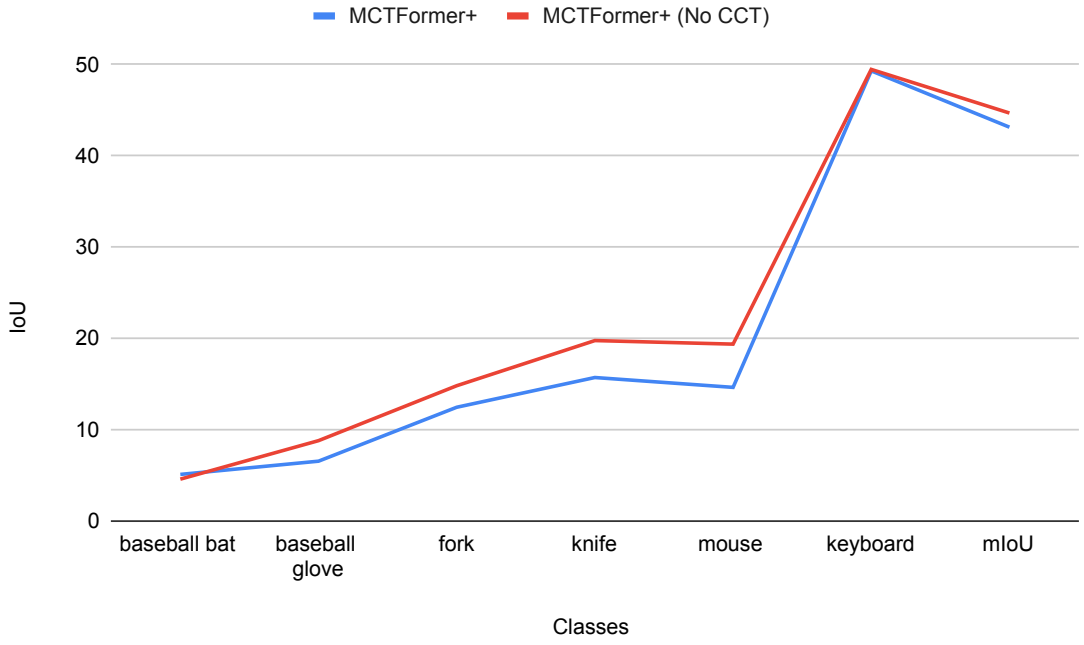
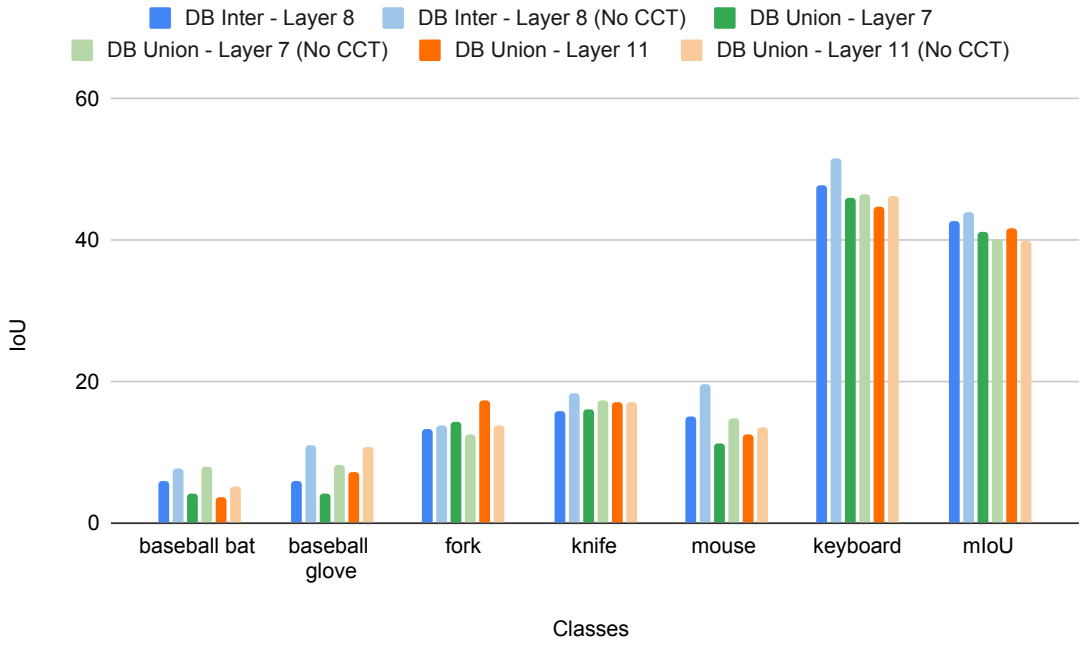


Figure 6.4: Graph comparing the performance of various models including and excluding CCT, for VOC. Included are baseline comparisons to MCTFormer+ — the blue line includes CCT and the red line does not. Two high-performing Single Layer Explicit Cross Image Attention models are included — the first uses the Intersection overlap task and applies cross attention on Layer 8, and the second uses the Union overlap task and applies cross attention on Layer 11. The dual-branch models are shaded: the darker shade includes CCT and the lighter shade does not.



(a) Baseline Comparison



(b) Early Dual-Branch Model Comparison

Figure 6.5: Graph comparing the performance of various models including and excluding CCT, for COCO. (a) shows the baseline comparison between MCTFormer+ — the blue line includes CCT and the red line does not. (b) shows comparisons between three high-performing Single Layer Explicit Cross Image Attention models — the first uses the Intersection overlap task and applies cross attention on Layer 8, and the second and third use the Union overlap task and apply cross attention on Layer 7 and 11, respectively. The dual-branch models are shaded: the darker shade includes CCT and the lighter shade does not.

identified as having extreme co-occurrence, there is still high co-occurrence between the Baseball Bat and Sports Ball classes (16.43%), and there is one-way dependence between all these classes and Person (i.e. there are a lot of images of a person without one of these classes, but few images of one of these classes without a person). This increased dependence between the classes is likely what causes the model to mistakenly think they are present, and so it "finds" a region of the image which it allocates to these absent classes.

Similarly, in the second example in Figure 6.6, only the Baseball Bat and Baseball Glove are present, but the third row's cosine similarities also identify the Person and Sports Ball classes as being present, although with a far greater degree of uncertainty. Therefore, we have a scenario where the existence of one class (e.g. Baseball bat) implies the existence of a second class (e.g. Person), but not the other way round. This dependence needs to be overcome.

CCT is intended to help distance the class tokens within an image and, in our dual-branch framework, it will perform this task in both branches. However, there is no guarantee that the class tokens will align between the branches (i.e. the "present" direction in the primary branch might be different from the "present" direction in the auxiliary branch), hence the cross attention will struggle to effectively exchange information. This is conceptually presented in Figure 6.7, where the tokens for the same, present classes are aligned within each branch, but are not aligned between the branches. Despite this, the individual class tokens will still average to a positive value, meaning the correct, positive classification will still be reached by the Classification Module for each branch in our dual-branch framework. This potential alignment disagreement, combined with the false activations in the third row of Figure 6.6, necessitate an additional learning module to help align the distance metric between the branches and to help disambiguate co-occurring classes.

To help develop intuition for this additional module, which we call CCT Prime, let c_i, for $c \in \{b, p\}$ (Baseball Bat and Person, respectively) and $i \in \{p, a\}$ (Primary and Auxiliary, respectively), be a class token for class c taken from branch i. Also let b_p and p_p be present for the primary image, and only b_a be present for the auxiliary image. Then CCT encourages the following alignment between the class tokens for the primary image:

$$b_p \cdot b_p = 1 \tag{6.6}$$
$$p_p \cdot p_p = 1 \tag{6.7}$$
$$b_p \cdot p_p = 0, \tag{6.8}$$

and for the auxiliary image (remember that absent classes have their tokens masked out of CCT, since they cannot be meaningfully operated on):

$$b_a \cdot b_a = 1 \tag{6.9}$$
$$p_a \cdot p_a \quad (\text{masked}) \tag{6.10}$$
$$b_a \cdot p_a \quad (\text{masked}). \tag{6.11}$$

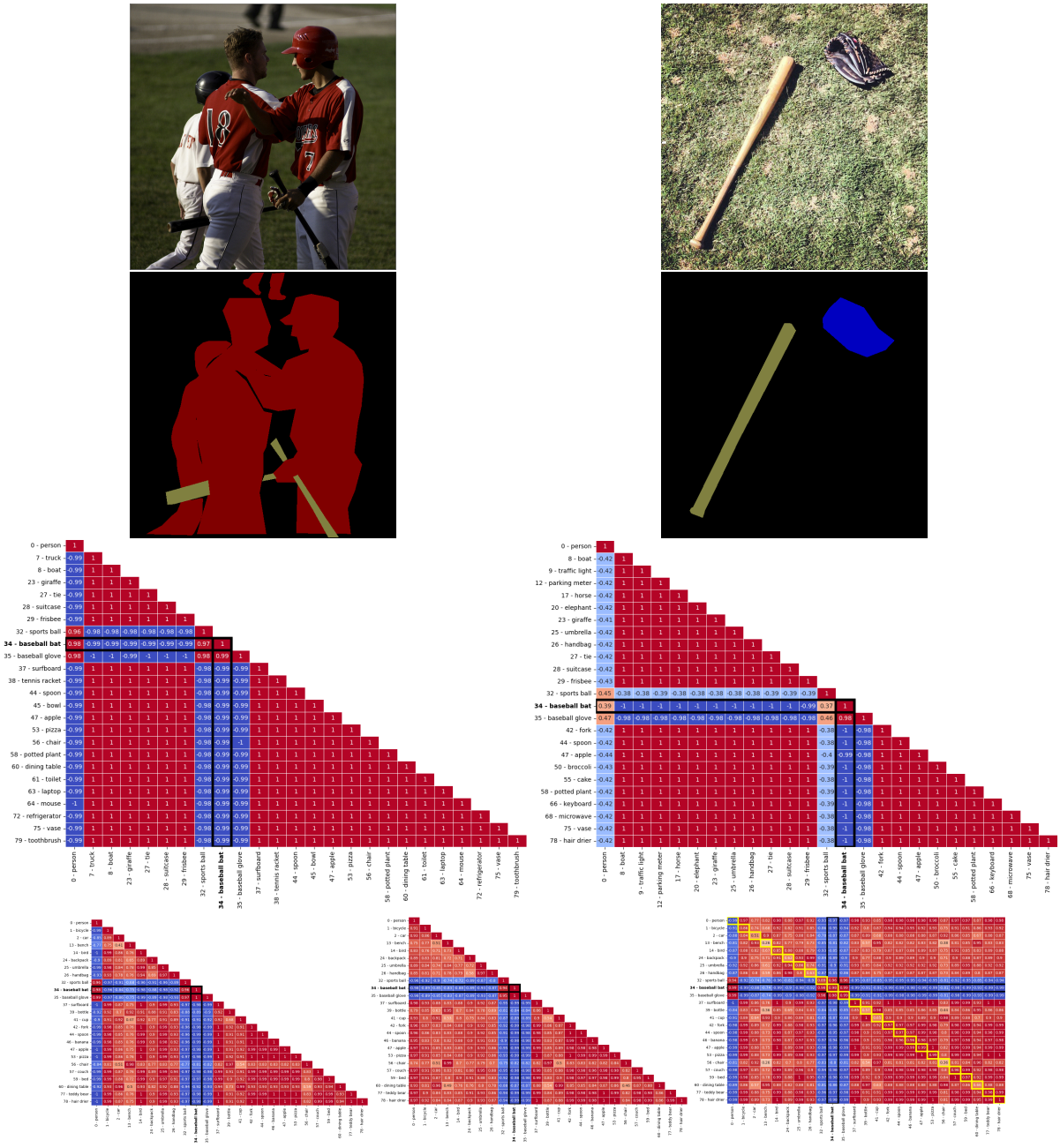
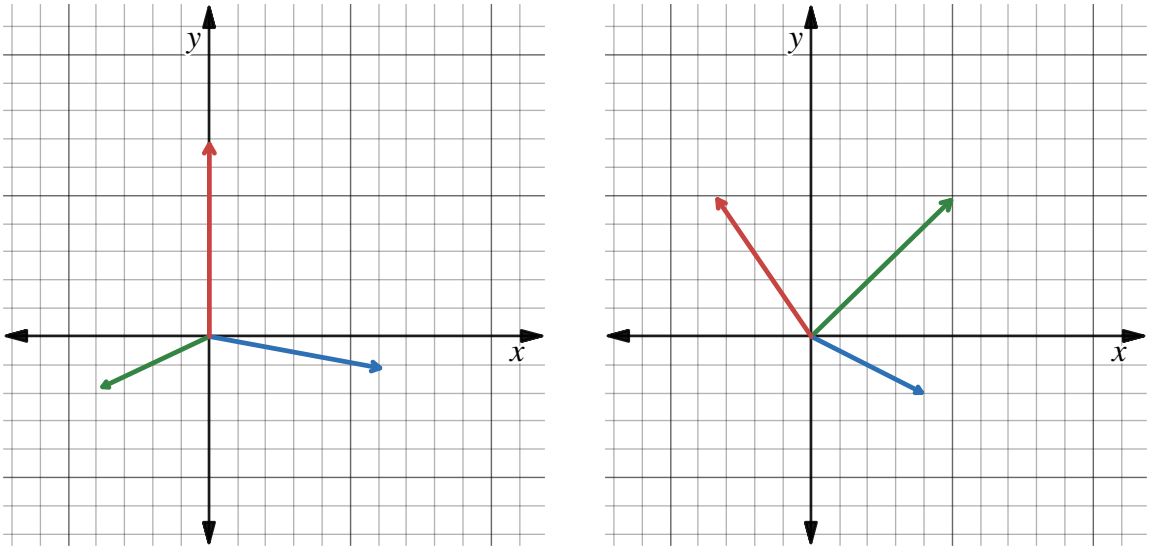


Figure 6.6: Figure showing the cosine similarity between class tokens from the final layer of the transformer for various images. Each column corresponds to an image from COCO. The first row is the original image, the second is the ground truth pixel mask, and the final two rows present the lower triangular matrix of cosine similarity between the class token represented by each row and column (the complete matrix is symmetrical). Specifically, the third row presents the cosine similarities for MCTFormer+ and the last row presents them for the dual-branch model trained with no cross image attention, only CCT and CCT Prime. **The first two plots in the final row are the cosine similarities between the class tokens for each image separately, and the final plot is the cosine similarities between each image's class tokens (columns are the first image's class tokens and rows are the second image's).** The cosine similarity ranges from -1 (blue) to 1 (red).



(a) Primary Branch (b) Auxiliary Branch

Figure 6.7: Comparison between potential class token directions being aligned differently between the primary and auxiliary branches. Three arbitrary classes are shown. Each class token still averages to a positive score, which is directly used in the Classification Module of our dual-branch framework, allowing the correct classification to be reached. The grid lines are only included to reference the change in direction between the two branches.

CCT does not align the class tokens between the primary and auxiliary branches. Hence, we need a module to perform:

$$p_p \cdot p_a = -1 \tag{6.12}$$
$$p_p \cdot b_a = 0 \tag{6.13}$$
$$b_p \cdot p_a \quad (\text{masked}) \tag{6.14}$$
$$b_p \cdot b_a = 1, \tag{6.15}$$

where the semantics mimic CCT:

1. Equation (6.12) equals -1 because p_p and p_a point in different directions (only p_p is present),
2. Equation (6.13) equals 0 because p_p and b_a are both present classes, but are different classes,
3. Equation (6.14) is masked because p_a is not present and the dot product does not involve the same class and therefore there is no useful information to be gleaned, and
4. Equation (6.15) equals 1 because b_p and b_a are both present.

This can be encoded in matrix form:

$$\begin{pmatrix} p_p \cdot -p_a & p_p \cdot b_a \\ \cancel{b_p \cdot -p_a} & b_p \cdot b_a \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \emptyset & 1 \end{pmatrix}, \tag{6.16}$$

where the "cancellation" indicates the masking. Note that in order for the matrix to equal the identity matrix, any absent class tokens (i.e. p_a) have to be negated (instead of negating the corresponding element in the identity matrix).

This becomes:

$$\begin{pmatrix} p_p \\ b_p \end{pmatrix} \begin{pmatrix} -p_a & b_a \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \emptyset & 1 \end{pmatrix}, \tag{6.17}$$

where the masking is applied after the vectors' outer product is taken.

This reduced example with only two classes generalises to all classes as:

$$f\left(\mathbf{C}_p^T \cdot \mathbf{C}_a\right) = \mathbf{I}, \tag{6.18}$$

where $\mathbf{C}_{\{p,a\}} \in \mathbb{R}^{C \times d}$ is the class token matrix for the primary and auxiliary branches, respectively, $\cdot$ denotes matrix multiplication, and function f applies the negation to absent class tokens and masks the dot product between two class tokens, one from each branch, where one of them is absent and the second is not the token for the same class as the first. There are C class tokens, each of dimension d.

Equation (6.18) describes the mathematics for the additional learning module, CCT Prime. Its loss is similar to that of CCT's in Equation (6.4):

$$\mathcal{L}_{\text{CCTPrime}} = \frac{1}{L} \sum_{i=0}^{L-1} \text{CrossEntropyLoss}(f\left(\mathbf{C}_p^T \cdot \mathbf{C}_a\right), \mathbf{I}), \tag{6.19}$$

where the symbols have the same meaning as in the CCT cross entropy loss function in Equation (6.4).

6.3.4.1 Initial Experimental Results

In order to establish the effectiveness of CCT Prime, we train two dual-branch models without cross attention, where both have CCT Prime enabled, but only one has CCT enabled. Training these models without cross attention is reducing each branch to MCTFormer+, since no direct communication between them occurs (communication is implicit by way of CCT Prime).

The results are presented in Table 6.1 for classes of interest. Comparing MCTFormer+ without CCT (2nd row) to the dual-branch model with CCT Prime (last row) (the only difference is essentially turning on CCT Prime), performance drops substantially for all classes, suggesting CCT Prime is hurting performance. However, it is designed to work in tandem with CCT. When comparing the two dual-branch models (i.e. the only difference is that CCT is turned on for one of them), some classes improve performance (Baseball Bat, Fork, and mIoU), but the others don't. When comparing MCTFormer+ with CCT (1st row) to the dual-branch model with both CCT and CCT Prime (3rd row) (the only difference is essentially turning on CCT Prime), performance decreases for all except the knife and fork classes. Overall, this would suggest that CCT Prime is either not performing as intended, or that there is some fundamental flaw in its design.

To help decide which of these causes is more accurate, we examine the cosine similarities between class tokens and how they change with the addition of CCT Prime. These

Model	CCT	CCT Prime	Classes (IoU)						
			baseball bat	baseball glove	fork	knife	mouse	keyboard	mIoU
MCTFormer+	✓	✗	5.08	6.53	12.44	15.70	14.62	49.31	43.14
MCTFormer+	✗	✗	4.56	8.78	14.80	19.75	19.36	49.47	44.69
Dual-Branch (No Cross Attention)	✓	✓	4.23	5.65	16.47	15.72	14.00	47.75	42.62
Dual-Branch (No Cross Attention)	✗	✓	2.96	5.84	13.16	15.71	15.03	48.15	41.71

Table 6.1: The experimental data for models trained with CCT Prime and no Cross Image Attention, on **COCO**. The dual-branch model does not have cross attention applied — it is essentially parallel MCTFormer+ models, with no communication happening between them, in an attempt to isolate the effect of CCT Prime. There is also a dual-branch model version with CCT for comparison. A baseline comparison to MCTFormer+ with and without CCT is also included. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest.

comparison plots are included in the last row of Figure 6.6: the first two are the cosine similarities between the class tokens within each of the first and second images, respectively, and the last plot is the cosine similarities between the class tokens from each image. The first image's cosine similarities are largely the same as before we applied CCT Prime, except that the absent class tokens point in a larger array of directions (that are still opposite to the "present" direction). The second image's cosine similarities follow the same trend, except that the person class is no longer erroneously activated. The final plot highlights that the class tokens between the images are aligned such that each branch is agreed on the "present" and "absent" directions. Therefore, CCT Prime is performing as intended, and addresses some of the misclassification issues. The cause of it degrading performance must be more intricate and requires further investigation in Section 6.3.5.

Consequently, CCT Prime addresses the one-way dependence issues, but not the co-occurrence issues, which we address with the final Cross Image Attention (CIA-AP) in Section 4.5.1.

6.3.5 Contrastive Learning with CIA-AP

Since in the introduction of CCT Prime in Section 6.3.4, substantial parallel development occurred for Cross Image Attention. The initial CCT performance (Section 6.3.3) was established using Single Layer Explicit Cross Attention (Section 5.2.3), which has since been improved upon, with the vast majority of that iterative development occurring with CCT disabled (since the baseline MCTFormer+ performed better without it). Consequently, CIA-AP (Section 5.2.6) using the mean to augment the primary and auxiliary query matrices (Section 5.2.7) has been found to be the best cross image attention scheme.

To re-establish the impact of CCT and CCT Prime, we simply re-enable them for some of the best-performing models so far. That is, we use the following configurations (where each setting was explained in depth in Section 5.2.6 when CIA-AP was formulated):

1. Co-occurrence thresholds of 0.46 (top 2 co-occurring pairs of classes) and 0.2491 (top 10) — this determines which classes are used for cross attention (and, by extension, when special mining is used to choose the auxiliary image).

2. "Turnoff" mode optionally enabled — if no co-occurring class pair exists in the primary image, then no cross attention will be performed between the class tokens, and this setting allows the patch token cross attention to be disabled as well.

3. The layer groups (6, 7, 8) and (7, 8, 9) on which we apply cross attention.

Figure 6.8 presents the results from these runs, and includes comparisons to MCTFormer+ (with and without CCT) and prior high-performing CIA-AP models trained with mean augmentation and no CCT. In these runs where we re-enable CCT and CCT Prime, both are given a weighting of 1 when contributing to the loss function (which is described in overall loss function in Equation (4.17)). Since there are many models, only the aggregate performance is shown (by taking the maximum and average per class over the models) — the complete set of results for the classes of interest is provided in Table A.7. Evidently, the results become notably worse when we enable CCT and CCT Prime compared to when both were disabled. The only exception is that the peak performance for the Fork class outperformed all the baselines, but the average on this class did not, suggesting this was an inconsistent result.

This can be explained by looking at the curves for components of the loss function in Figure 6.9, where each component is equally weighted. The loss components are the classification and CCT losses for each branch, and CCT Prime, which only has a single instance, since it applies to both branches simultaneously. The models' training curves are averaged together over each loss component (and branch). When the components start plateauing at Epoch 10, the classification loss is 0.09 for each branch, the CCT losses are 3.36 and 3.37 for each branch respectively, and the CCT Prime loss is 3.72. This means classification contributes $\frac{0.18}{10.63} = 1.69\%$ to the total loss and contrastive learning contributes $\frac{10.45}{10.63} = 98.31\%$ — this is a massive difference in magnitude that most likely induces the model to focus more on solving the contrastive learning objective as opposed to simply using it as an aid.

Therefore, we perform a hyperparameter sweep over the weight with which CCT and CCT Prime should contribute to the loss function, which is presented in Table 6.2. We use the best performing model (CIA-AP with mean augmentation and "turnoff" mode enabled, co-occurrence threshold of 0.2491, and cross attention applied to layers 6, 7, and 8) and vary the coefficients of CCT and CCT Prime over all combinations of the values $(0, 0.01, 0.1, 0.5)$.

As a general trend, weightings of 0.5 for CCT and CCT Prime perform the worst (paired with any other weight), and do so by a considerable amount. The other combinations of weightings have merit based on the particular combination. CCT weighted at 0.1 seems to perform well for mIoU overall, but is outperformed for the classes of interest by other weight combinations — this means that "other" non-co-occurring classes perform better for these models. For CCT and CCT Prime weighted at combinations of 0 and 0.01, the best performance is achieved for the Baseball Bat, Baseball Glove, Knife,

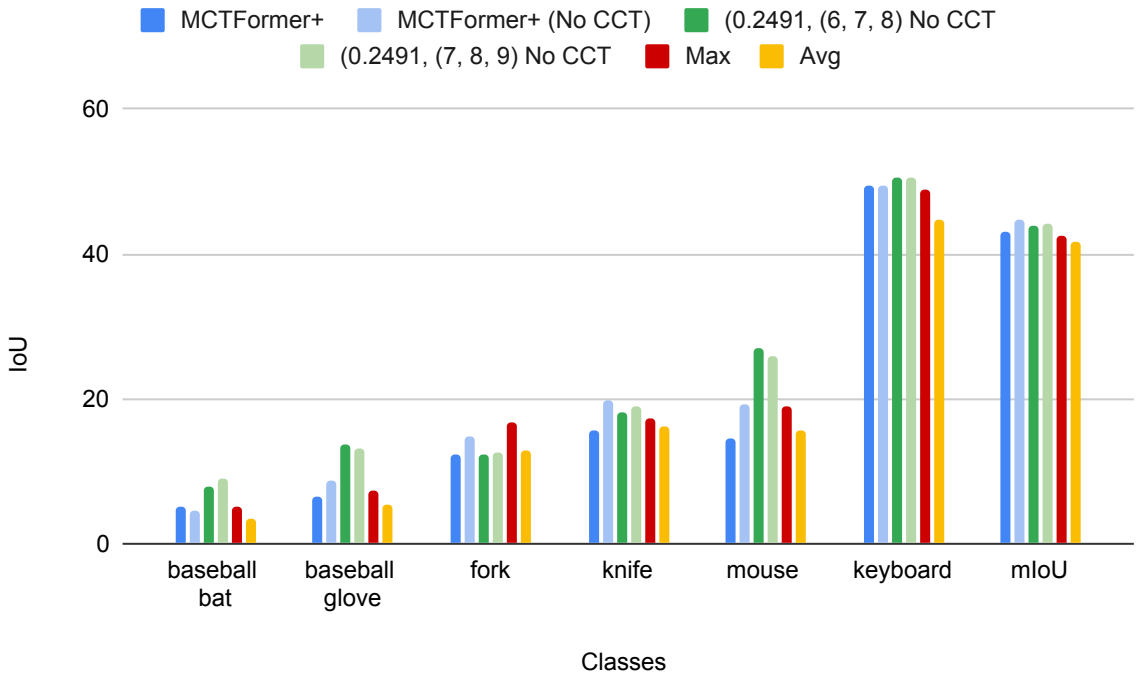


Figure 6.8: Comparison plot showing the performance of high performing CIA-AP models with CCT and CCT Prime re-enabled, and each with a weight of 1 (i.e. contribution to the loss function), on COCO. Included are baseline comparison to MCTFormer+ with and without CCT, in dark and light blue, respectively. The two green models are included for comparison, and are two of the highest performing models trained with CIA-AP with mean augmentations and without CCT — both use a co-occurrence threshold of 0.2491, but the darker one uses layers 6, 7, and 8, and the lighter one uses layers 7, 8, and 9. In total, 8 experiments were conducted with CCT and CCT Prime re-enabled, but only the maximum (red) and average (yellow) per class (of interest) over these runs is presented.

Relative Weighting		Classes (IoU)						
CCT	**CCT Prime**	**baseball bat**	**baseball glove**	**fork**	**knife**	**mouse**	**keyboard**	**mIoU**
MCTFormer+		5.08	6.53	12.44	15.70	14.62	49.31	43.14
MCTFormer+ (no CCT)		4.56	8.78	14.80	19.75	19.36	49.47	44.69
0	0	7.80	13.79	12.34	18.02	26.95	50.63	44.01
0	0.01	7.41	14.38	12.45	18.47	26.32	50.43	44.26
0	0.1	6.53	10.23	15.79	17.29	22.53	50.39	43.91
0	0.5	5.02	9.76	13.58	16.72	21.53	49.34	42.70
0.01	0	7.68	14.31	13.94	19.10	27.36	51.06	44.26
0.01	0.01	7.13	12.85	14.61	19.82	27.28	51.11	44.49
0.01	0.1	4.85	9.11	16.16	18.09	23.02	51.61	44.22
0.01	0.5	5.23	10.28	12.66	16.79	21.24	47.92	43.04
0.1	0	5.80	10.28	13.63	18.36	23.53	53.37	44.44
0.1	0.01	6.80	9.19	13.82	17.67	24.35	52.88	44.41
0.1	0.1	4.30	12.95	13.80	19.80	23.94	50.84	44.34
0.1	0.5	4.80	6.82	11.10	15.94	19.89	49.09	43.08
0.5	0	3.88	7.48	13.22	17.39	21.30	51.71	43.84
0.5	0.01	3.72	7.94	14.30	16.90	23.93	50.53	43.64
0.5	0.1	3.11	9.00	13.10	18.63	22.47	51.92	43.85
0.5	0.5	4.04	4.86	6.96	14.98	20.26	50.54	42.33

Table 6.2: A table showing the sweep over CCT and CCT Prime weightings for their contribution to the overall loss function (Equation (4.17)). A single already-high-performing model configuration was used, with only the CCT and CCT Prime coefficients changing: apply CIA-AP using mean augmentation to layers 6, 7, and 8, using "turnoff" mode, and a co-occurrence threshold of 0.2491. The first two rows are baseline comparisons to MCTFormer+ (with and without CCT). The first two columns are the coefficient values, which are colour-coded for each of comparison and reference. The other columns present the IoU scores for co-occurring class pairs of interest and the overall mIoU score for COCO: the values are shaded from the smallest value in each column (red) to the largest value (green).

and Mouse classes, which is the majority of the classes of interest, with the remaining classes still performing competitively to the configurations with their best scores. Consequently, we opt to use 0.01 as the weight for both CCT and CCT Prime, since this model obtains amongst the highest scores for classes of interest, and also maintains the highest overall mIoU score (and is within 0.2% of the best performing baseline — MCTFormer+ without CCT). This implies that we obtain substantial improvement in our target area — the co-occurring class pairs — and also maintain reasonable performance in all the other classes in the dataset (since the co-occurring classes increase notably, there is a slight decrease in the performance of the other classes, such that the mIoU is approximately maintained — this is discussed further in Section 4.5 when the final model's results are presented).

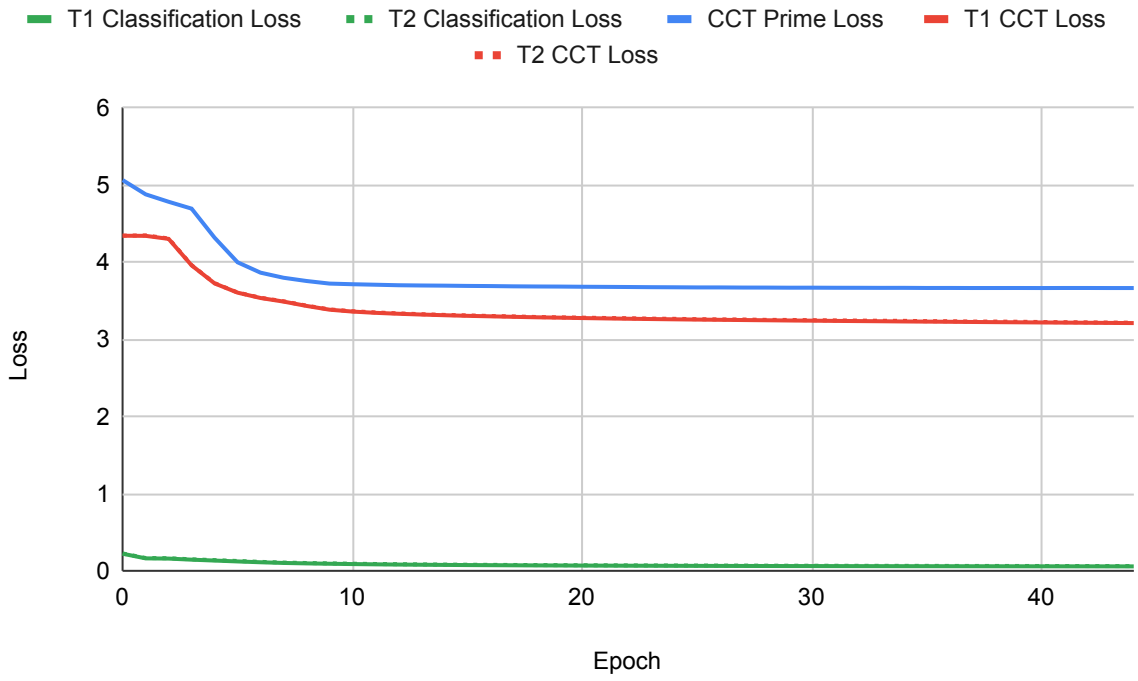


Figure 6.9: Training curves for components of the Loss Function over the course of 45 epochs, aggregated over multiple training runs. The classification and CCT losses per branch are displayed ("T1" for the first branch and "T2" for the second branch), as well as the CCT Prime loss between the branches. Each component is equally weighted in the final loss function. The models used are the ones trained using CIA-AP (with mean augmentation), various co-occurrence thresholds, and CCT and CCT Prime enabled and equally weighted to the other loss components (i.e. a weighting of 1) — their average per component is displayed, along with the variance (which is small). The top curve is for CCT Prime, the middle one is for CCT, and the bottom one is for Classification. The Classification Loss and CCT Loss curves actually contain both the first and second branches' losses superimposed on each other since they are so similar in value. When the components start plateauing at Epoch 10, the classification loss is 0.09 for each branch, the CCT losses are 3.36 and 3.37 for each branch respectively, and the CCT Prime loss is 3.72.

6.4 Summary

This chapter presented a thorough investigation into the application of contrastive learning within our framework for weakly supervised semantic segmentation. We began by introducing the foundational CCT method from MCTFormer+ in Section 6.3.1, which aimed to enforce orthogonality among class token embeddings using Cross Entropy Loss. However, our investigation into the practical behaviour of these class tokens in Section 6.3.2 revealed that instead of forming an orthogonal basis, the tokens pointed in general directions, but still distinct directions, based on whether the corresponding class was present in the image, thereby forming a conic structure, as depicted in Figure 6.3. This phenomenon occurs because the classifier primarily drives the tokens toward these directions, except when the classifier is uncertain, allowing CCT to influence the embeddings to a more orthogonal setting. This insight helped explain why CCT degraded performance using initial cross attention schemes in our dual-branch framework, and on COCO (Section 6.3.3), despite showing promising results for the single-branch MCTFormer+ on VOC.

Our subsequent efforts focused on addressing this issue through the development of an additional contrastive learning module, CCT Prime, introduced in Section 6.3.4. CCT Prime aimed to align the representations between branches, acknowledging that while CCT promotes alignment within each branch in our dual-branch framework, it does not necessarily ensure alignment across branches. Misalignment between the primary and auxiliary branches could undermine the cross attention mechanism, particularly when the class token directions differ substantially.

Finally, Section 6.3.5 integrated both CCT and CCT Prime with the CIA-AP cross attention mechanism from Section 5.2.6. We found that reducing the weighted contribution of CCT and CCT Prime was essential to avoid overpowering the classification loss, which previously led to degraded performance. This integration achieved a better balance between the various learning objectives, resulting in improved performance for the co-occurring classes of interest while maintaining competitive overall results.

Chapter 7

Conclusion and Future Work

In this chapter, we first outline potential directions for future research in Section 7.1, emphasising areas where our work encountered specific challenges or raised concerns that merit further investigation. These avenues include enhancing our cross attention and contrastive learning mechanisms, improving segmentation model performance, and expanding our framework to new domains. Following this, we consolidate our findings and their implications in Section 7.2, providing a cohesive summary of our contributions and their significance.

7.1 Future Work

While our proposed framework achieves substantial improvements for co-occurring classes in Weakly Supervised Semantic Segmentation, several areas warrant further investigation to improve performance and broaden the applicability of our approach.

7.1.1 Investigating Trade-Off for Co-Occurring Class Performance Increase

An important avenue for future work is investigating the underlying cause of the observed trade-off between improvements for co-occurring classes and slight degradations for non-co-occurring classes, which is evident in the final results presented in Section 4.5. While our approach effectively boosts performance for co-occurring class pairs, we occasionally see a minor decline in accuracy for other classes that do not experience strong co-occurrence effects. One potential explanation is that the enhanced cross attention mechanism and contrastive learning strategies may shift the model's focus toward resolving co-occurrence ambiguities, inadvertently reducing emphasis on other object categories. Another possibility is that the auxiliary image selection process introduces subtle biases that affect the feature representations of non-co-occurring classes. Future research could explore strategies to mitigate this trade-off, such as refining the auxiliary image mining process, introducing adaptive weighting mechanisms in the cross attention module, or adjusting the balance between contrastive learning objectives and standard classification constraints. Understanding and addressing this

issue will be critical for ensuring broad performance gains, especially for co-occurring classes, without unintended side effects.

7.1.2 Improving Cross-Attention Mechanisms

Our CIA-AP mechanism (Section 5.2.6) provides a principled method for enhancing attention to co-occurring classes. However, its application from the beginning of training may not be optimal. Currently, CIA-AP is applied to any auxiliary image satisfying the condition of containing exactly one overlapping class with the primary image, as determined by image-level labels. This condition does not account for model confidence regarding the presence of the co-occurring class, which could result in a low-confidence image being sampled, which does not aid in improving the attention for co-occurring classes.

Future work should consider improving the selection of auxiliary images by incorporating model confidence. One potential approach involves using an epsilon-greedy mechanism inspired by reinforcement learning, where high-confidence images are predominantly selected, but low-confidence samples are occasionally included to encourage exploration. Additionally, since the model requires time to learn object recognition, implementing a training warmup phase where only the primary branch is used could be beneficial before incorporating the auxiliary branch via CIA-AP.

Additionally, our ablation studies in Section 5.2.6 and Section 5.2.7 only really considered three methods of augmenting the primary and auxiliary sets of query vectors: taking the mean (which we finally use), the sum, and swapping the sets of vectors. Other augmentation methods exist to combine the query vectors that might show promise in their own right, such as projecting one query vector onto another (which we discarded, but did not actually investigate), taking the cosine similarity between them, or even taking the dot product between them.

Furthermore, although our initial experiments with a third branch did not yield improvements over the dual-branch case (Section 5.2.9), this limitation may have been due to the particular cross-attention or contrastive learning schemes we employed. Revisiting this idea with improved techniques could potentially enhance performance. Another area worth investigating is incorporating more images for the learning of global class tokens. For example, a memory-based mechanism, similar to RCA [Zhou *et al.* 2022], could be employed to refine global representations that persist across different images.

7.1.3 Enhancing Contrastive Learning

Our introduction of CCT Prime in Section 6.3.4 improved performance by promoting alignment between the primary and auxiliary branches. However, reducing the weight of CCT and CCT Prime was necessary to avoid overpowering the classification loss (Section 6.3.5). While our modifications have proven effective, further refinement of these contrastive learning modules could yield additional improvements. Balancing

their contributions with the classification loss remains a challenge, but enhanced design or training protocols could address this issue.

Moreover, incorporating memory mechanisms as mentioned previously for CIA-AP may also benefit contrastive learning. Promoting better alignment through representations that persist across images could further enhance segmentation performance for co-occurring classes.

7.1.4 Improving Segmentation Model Performance

One of the most critical areas for future work is enhancing the performance of the segmentation model itself. As evidenced by our results in Table 4.4, certain classes like "toaster" and "hair drier" experienced substantial degradation from the pseudo label quality to the segmentation mask quality. This collapse in performance persists even when the segmentation model is trained using ground-truth masks (discussed in Section 4.5, with data presented in Table A.2), suggesting that the issue is inherent to the segmentation model rather than our pseudo label generation process.

A key aspect of this investigation should focus on exploring more advanced segmentation architectures, such as DeepLab-V3 [Chen *et al.* 2017a] and DeepLab-V3+ [Chen *et al.* 2018]. While we employed DeepLab-V1 primarily to maintain consistency with prior work [Ahn and Kwak 2018; Zhang *et al.* 2020 2021; Xu *et al.* 2021 2024], these more modern architectures are known for their strong performance in supervised segmentation tasks, and their integration could enhance weakly supervised learning when paired with high-quality pseudo labels. Understanding why the segmentation model underperforms for certain classes, even when provided with ground-truth masks, will be crucial for identifying architectural or training-related factors contributing to this collapse.

Additionally, improving the segmentation model itself remains essential for future work. As our pseudo labels continue to approach the quality of ground-truth masks, diminishing returns from segmentation model training may arise. There is a risk of reaching a point where the pseudo labels surpass the quality of the predicted segmentation masks, potentially resulting in regression if the segmentation model is used. Parallel development focusing on both improving the segmentation model and refining pseudo label generation will be necessary to avoid this outcome.

7.1.5 Evaluation with COCO Masks

In Section 3.2.2.1, we developed our own versions of the COCO segmentation masks since a standardised version was not available. It is essential to investigate whether these custom masks alter the relative performance of prior methods. Re-evaluating previous models using our segmentation masks will help determine if their reported performances were influenced by differing mask generation techniques. Such an investigation would provide a fairer comparison across methods and help establish a more robust evaluation benchmark.

7.1.6 Broader Applications and Domain Adaptation

The class co-occurrence problem is not unique to natural image datasets like PASCAL VOC 2012 and MS COCO 2014. Our framework could be particularly valuable for weakly supervised segmentation in medical imaging, where certain structures, such as organs, consistently co-occur within the same scans. Extending our work to this domain would require developing appropriate datasets and adapting our framework to account for the unique challenges posed by medical imagery. Additionally, further investigating the generalisation of our framework to other domains where co-occurrence is prevalent could significantly broaden its applicability.

7.2 Conclusion

This dissertation addresses the challenge of Weakly Supervised Semantic Segmentation (WSSS) with a particular focus on the issue of class co-occurrence, a problem that has been largely overlooked in previous research. Unlike co-occurrence between classes and background features, our work investigates co-occurrence between classes within the dataset itself, which introduces distinct and often more nuanced challenges that warrant further attention.

In Chapter 3, we establish the existence of the class co-occurrence problem and provide empirical evidence of its detrimental effects on segmentation performance, particularly for the MS COCO 2014 dataset, where co-occurrence is far more prevalent compared to PASCAL VOC 2012. We demonstrate that performance degradation is not evenly distributed across co-occurring classes; one class often suffers more than the other. This imbalance arises because the model tends to focus more on distinguishing the classes rather than learning their fine-grained characteristics, such as precise object boundaries, which are essential for accurate segmentation.

Building on this insight, Chapter 4 presents our final Cross Image Attention (CIA-AP) method alongside the intuition behind CCT Prime. Our approach develops a dual-branch transformer framework inspired by MCTFormer+, where cross attention is applied between two images, one primary and one auxiliary image, where the auxiliary image is chosen to share exactly one co-occurring class with the primary image. By augmenting the primary image's Query matrix with that of the auxiliary image, CIA-AP enhances attention to the co-occurring classes, leading to improved segmentation performance, particularly on challenging co-occurring class pairs in MS COCO 2014. Meanwhile, Chapter 5 provides an ablation study of various factors influencing cross attention, systematically evaluating different design choices to refine our final approach.

Chapter 6 then delves into the use of contrastive learning, specifically the CCT module of MCTFormer+, which aims to enforce orthogonality among class token embeddings. However, our investigation reveals that instead of forming a strictly orthogonal basis, class tokens align themselves according to broad directional clusters associated with class presence. This behaviour, driven primarily by classification loss rather than orthogonality constraints, limits the effectiveness of CCT when applied to COCO, particularly under the cross attention framework.

Additionally, we introduce CCT Prime, a contrastive learning module designed to address potential alignment inconsistencies between the class tokens of the primary and auxiliary branches. While CCT promotes separation of class tokens within a branch, it does not inherently ensure that corresponding class tokens across branches remain aligned. This misalignment can weaken the effectiveness of cross attention, as the auxiliary branch may not provide the intended complementary information. CCT Prime mitigates this issue by explicitly encouraging alignment between class tokens across branches, reinforcing the cross attention mechanism's ability to refine feature representations. The integration of CCT Prime with CIA-AP achieves a more balanced optimisation of learning objectives, leading to improved segmentation performance for co-occurring classes while maintaining competitive overall accuracy.

Overall, our work demonstrates that addressing the class co-occurrence problem is essential for improving WSSS performance. While our approach does not yet surpass the state-of-the-art in terms of overall mIoU, it achieves substantial performance gains for the targeted co-occurring classes while maintaining competitive overall accuracy.

Specifically, on PASCAL VOC 2012, our final model achieves notable improvements for co-occurring classes, such as *chair*, where the pseudo label quality improves by a factor of 1.15×, and *dining table*, which improves by 1.04×. Importantly, these improvements come without a substantial drop in overall performance, as our model maintains a competitive pseudo label performance of 66.29% mIoU compared to 67.04% in MCTFormer+.

On MS COCO 2014, where the class co-occurrence issue is more pronounced, our improvements for co-occurring classes are even more substantial. For *baseball glove*, our pseudo labels improve by a factor of 2.0×, with further enhancements in the segmentation masks, achieving a 3.3× increase on both the training and validation sets. Similarly, *baseball bat* sees a 1.4× improvement in pseudo labels, and a 2.4× and 2.8× increase in the training and validation segmentation masks. Mouse follows the same trend, with 1.9× better pseudo labels and a 2.4× and 2.5× improvement in training and validation segmentation masks, respectively. While the overall mIoU for pseudo labels sees only a modest increase of 1.03×, and increases of 1.07× and 1.05× for the training and validation segmentation masks, these results highlight that our method effectively recovers performance for co-occurring classes, which would otherwise be severely degraded due to their frequent co-occurrence. The remaining co-occurring classes generally mentioned through this work (*keyboard*, *knife*, and *fork*) all improve over MCTFormer+, albeit by a small amount, for the pseudo labels, and the training and validation segmentation masks, with the sole exception of the *keyboard* validation segmentation masks, which decrease by a miniscule 0.7%.[1]

These findings confirm that our approach successfully mitigates the detrimental effects of class co-occurrence in weakly supervised semantic segmentation, achieving substan-

[1]Note that the improvement factors mentioned are relative to MCTFormer+ using CCT, and the class scores for our approach is the highest score from the two models we present during our final method.

tial improvements for targeted classes while preserving competitive performance across the dataset.

References

[Ahn and Kwak 2018] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4981–4990, 2018.

[Araslanov and Roth 2020] Nikita Araslanov and Stefan Roth. Single-stage semantic segmentation from image labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4253–4262, 2020.

[Bearman *et al.* 2016] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 549–565. Springer, 2016.

[Caesar *et al.* 2018] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE, 2018.

[Cao and Zhang 2023] Zhiyuan Cao and Jiacai Zhang. Gradient-coupled cross-patch attention map for weakly supervised semantic segmentation. *Neurocomputing*, 535:83–96, 2023.

[Caron *et al.* 2021] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[Chang *et al.* 2020] Yu-Ting Chang, Qiaosong Wang, Wei-Chih Hung, Robinson Piramuthu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation via sub-category exploration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8991–9000, 2020.

[Chen *et al.* 2017a] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *ArXiv*, abs/1706.05587, 2017.

[Chen *et al.* 2017b] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2017.

[Chen *et al.* 2018] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[Chen *et al.* 2021] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, 2021.

[Chen *et al.* 2023a] Junliang Chen, Weizeng Lu, Yuexiang Li, Linlin Shen, and Jinming Duan. Adversarial learning of object-aware activation map for weakly-supervised semantic segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[Chen *et al.* 2023b] Liyi Chen, Chenyang Lei, Ruihuang Li, Shuai Li, Zhaoxiang Zhang, and Lei Zhang. Fpr: False positive rectification for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1108–1118, 2023.

[Chen *et al.* 2023c] Xin Chen, Ben Kang, Dong Wang, Dongdong Li, and Huchuan Lu. Efficient visual tracking via hierarchical cross-attention transformer. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*, pages 461–477. Springer, 2023.

[Cheng *et al.* 2023] Zesen Cheng, Pengchong Qiao, Kehan Li, Siheng Li, Pengxu Wei, Xiangyang Ji, Li Yuan, Chang Liu, and Jie Chen. Out-of-candidate rectification for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23673–23684, 2023.

[Dai *et al.* 2015] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1635–1643, 2015.

[Deng *et al.* 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Devlin *et al.* 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Dosovitskiy *et al.* 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Everingham *et al.* 2012] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Ob-*

ject Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012.

[Gao *et al.* 2021] Wei Gao, Fang Wan, Xingjia Pan, Zhiliang Peng, Qi Tian, Zhenjun Han, Bolei Zhou, and Qixiang Ye. Ts-cam: Token semantic coupled attention map for weakly supervised object localization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2886–2895, 2021.

[Hadsell *et al.* 2006] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[Hariharan *et al.* 2011] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 international conference on computer vision*, pages 991–998. IEEE, 2011.

[He *et al.* 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[He *et al.* 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[He *et al.* 2023] Jingxuan He, Lechao Cheng, Chaowei Fang, Dingwen Zhang, Zhangye Wang, and Wei Chen. Mitigating undisciplined over-smoothing in transformer for weakly supervised semantic segmentation. *arXiv preprint arXiv:2305.03112*, 2023.

[Jiang *et al.* 2022] Peng-Tao Jiang, Yuqi Yang, Qibin Hou, and Yunchao Wei. L2g: A simple local-to-global knowledge transfer framework for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16886–16896, 2022.

[Khoreva *et al.* 2017] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 876–885, 2017.

[Kolesnikov and Lampert 2016] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 695–711. Springer, 2016.

[Krähenbühl and Koltun 2011] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24, 2011.

[Lee *et al.* 2021a] Jungbeom Lee, Eunji Kim, and Sungroh Yoon. Anti-adversarially manipulated attributions for weakly and semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4071–4080, 2021.

[Lee *et al.* 2021b] Seungho Lee, Minhyun Lee, Jongwuk Lee, and Hyunjung Shim. Railroad is not a train: Saliency as pseudo-pixel supervision for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5495–5505, 2021.

[Lee *et al.* 2022] Jungbeom Lee, Seong Joon Oh, Sangdoo Yun, Junsuk Choe, Eunji Kim, and Sungroh Yoon. Weakly supervised semantic segmentation using out-of-distribution data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16897–16906, 2022.

[Li *et al.* 2023] Ruiwen Li, Zheda Mai, Zhibo Zhang, Jongseong Jang, and Scott Sanner. Transcam: Transformer attention-based cam refinement for weakly supervised semantic segmentation. *Journal of Visual Communication and Image Representation*, 92:103800, 2023.

[Liang-Chieh *et al.* 2015] Chen Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *International conference on learning representations*, 2015.

[Lin *et al.* 2014] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[Lin *et al.* 2016] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3159–3167, 2016.

[Lin *et al.* 2022] Hezheng Lin, Xing Cheng, Xiangyu Wu, and Dong Shen. Cat: Cross attention in vision transformer. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022.

[Lin *et al.* 2023] Yuqi Lin, Minghao Chen, Wenxiao Wang, Boxi Wu, Ke Li, Binbin Lin, Haifeng Liu, and Xiaofei He. Clip is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15305–15314, 2023.

[Long *et al.* 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[Mai *et al.* 2020] Jinjie Mai, Meng Yang, and Wenfeng Luo. Erasing integrated learning: A simple yet effective approach for weakly supervised object localization. In

Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 8766–8775, 2020.

[Musgrave *et al.* 2020] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 681–699. Springer, 2020.

[Radford *et al.* 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Redmon *et al.* 2016] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[Ronneberger *et al.* 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[Rossetti *et al.* 2022] Simone Rossetti, Damiano Zappia, Marta Sanzari, Marco Schaerf, and Fiora Pirri. Max pooling with vision transformers reconciles class and shape in weakly supervised semantic segmentation. In *European conference on computer vision*, pages 446–463. Springer, 2022.

[Ru *et al.* 2022] Lixiang Ru, Yibing Zhan, Baosheng Yu, and Bo Du. Learning affinity from attention: End-to-end weakly-supervised semantic segmentation with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16846–16855, 2022.

[Ru *et al.* 2023] Lixiang Ru, Heliang Zheng, Yibing Zhan, and Bo Du. Token contrast for weakly-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3093–3102, 2023.

[Sanh *et al.* 2022] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

[Schroff *et al.* 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[Selvaraju *et al.* 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[Simonyan and Zisserman 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Springenberg *et al.* 2014] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[Su *et al.* 2021] Yukun Su, Ruizhou Sun, Guosheng Lin, and Qingyao Wu. Context decoupling augmentation for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7004–7014, 2021.

[Sundararajan *et al.* 2017] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[Touvron *et al.* 2021] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

[Vaswani *et al.* 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.* 2020] Guangrun Wang, Guangcong Wang, Xujie Zhang, Jianhuang Lai, Zhengtao Yu, and Liang Lin. Weakly supervised person re-id: Differentiable graphical learning and a new benchmark. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2142–2156, 2020.

[Wei *et al.* 2017] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1568–1576, 2017.

[Wu *et al.* 2019] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern recognition*, 90:119–133, 2019.

[Xie *et al.* 2022] Jinheng Xie, Xianxu Hou, Kai Ye, and Linlin Shen. Clims: Cross language image matching for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4483–4492, 2022.

[Xu *et al.* 2021] Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, Ferdous Sohel, and Dan Xu. Leveraging auxiliary tasks with affinity learning for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6984–6993, 2021.

[Xu *et al.* 2022] Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, and Dan Xu. Multi-class token transformer for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4310–4319, 2022.

[Xu *et al.* 2024] Lian Xu, Mohammed Bennamoun, Farid Boussaid, Hamid Laga, Wanli Ouyang, and Dan Xu. Mctformer+: Multi-class token transformer for weakly supervised semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 2024.

[Yang *et al.* 2024] Zhiwei Yang, Kexue Fu, Minghong Duan, Linhao Qu, Shuo Wang, and Zhijian Song. Separate and conquer: Decoupling co-occurrence via decomposition and representation for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3606–3615, 2024.

[Yoon *et al.* 2022] Sung-Hoon Yoon, Hyeokjun Kweon, Jegyeong Cho, Shinjeong Kim, and Kuk-Jin Yoon. Adversarial erasing framework via triplet with gated pyramid pooling layer for weakly supervised semantic segmentation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIX*, pages 326–344. Springer, 2022.

[Zeiler and Fergus 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[Zhang *et al.* 2015] Wei Zhang, Sheng Zeng, Dequan Wang, and Xiangyang Xue. Weakly supervised semantic segmentation for social images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2718–2726, 2015.

[Zhang *et al.* 2018] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1325–1334, 2018.

[Zhang *et al.* 2020] Bingfeng Zhang, Jimin Xiao, Yunchao Wei, Mingjie Sun, and Kaizhu Huang. Reliability does matter: An end-to-end weakly supervised se-

mantic segmentation approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12765–12772, 2020.

[Zhang *et al.* 2021] Fei Zhang, Chaochen Gu, Chenyue Zhang, and Yuchao Dai. Complementary patch for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7242–7251, 2021.

[Zhou *et al.* 2016] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[Zhou *et al.* 2022] Tianfei Zhou, Meijie Zhang, Fang Zhao, and Jianwu Li. Regional semantic contrast and aggregation for weakly supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4299–4309, 2022.

Appendix A

Supporting Experimental Data

This appendix presents supporting experimental results for the investigations conducted and discussed throughout the main body of this dissertation, particularly those related to the Cross Image Attention and Contrastive Learning chapters — Chapter 5 and Chapter 6, respectively. It comprises multiple tables and graphs, each accompanied by extensive captions clearly describing the data being presented and including references to the relevant sections or figures they are intended to support.

These results are not meant to be read in isolation but are intended to complement the concepts discussed earlier in the dissertation. Readers are encouraged to refer to these tables and graphs while reviewing the corresponding discussions to better understand and validate the findings of this work.

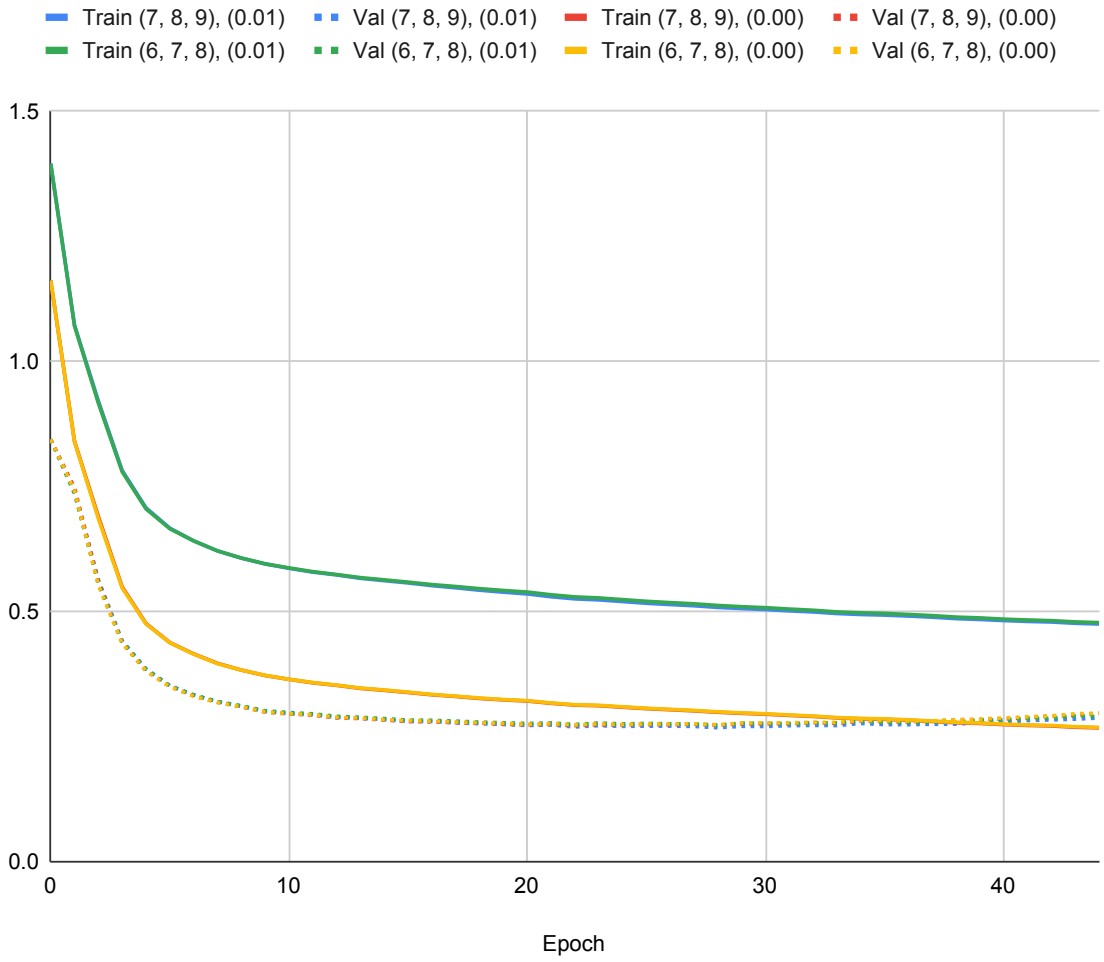


Figure A.1: Graph showing the training (solid lines) and validation loss curves (dotted) during training for the triple-branch version of our framework. The model configuration varies over the layer groups for which cross attention is applied and the loss weighting of CCT and CCT Prime — the legend is formatted as "*split* (*layer group*, (loss weight))". Whilst the training curves decrease over all the epochs, the validation curves become flat between epochs 20 and 30, and increase thereafter. This data supports the idea of overfitting and justifies the choice of using Epoch 25 as the "early checkpoint" in Section 5.2.9.

Classes	High Co-occurence	Split (IoU)		Classes	High Co-occurence	Split (IoU)	
		train	val			train	val
person		84.16	82.53	horse		90.40	82.49
chair	✓	43.44	33.80	mIoU		80.28	74.17
boat		80.31	67.77	bus		88.46	88.90
diningtable	✓	76.69	61.86	train		86.41	80.92
car		84.50	82.74	cow		85.75	82.59
sheep		92.43	89.32	cat		93.46	90.00
motorbike		83.93	81.57	bottle		81.42	78.01
bicycle		51.28	37.35	dog		92.62	85.96
bird		87.26	88.60	pottedplant		70.82	52.28
aeroplane		88.92	86.37	sofa		59.59	48.07
background		94.31	93.24	tvmonitor		69.79	63.23

Table A.1: Segmentation scores for **VOC** when the segmentation model (DeepLab-V1) is trained using the ground truth segmentation masks. Pairs of highly co-occurring classes have been identified. This data is used to generate the comparison ratios in Table 4.3 in Section 4.5, and is presented with the classes in the same order.

Classes	High Co-occurrence	Split (IoU)	
		train	val
baseball glove	✓	40.78	33.46
mouse	✓	48.74	41.90
tie		25.39	22.15
tennis racket		59.46	59.16
skis		7.57	8.33
baseball bat	✓	17.20	13.10
toaster		5.54	5.99
frisbee		59.07	54.09
skateboard		40.81	31.71
knife	✓	16.07	13.42
handbag		10.60	6.60
spoon		0.74	0.45
kite		48.55	46.96
chair		30.12	27.14
surfboard		60.07	55.53
cup		40.83	37.51
fork	✓	10.88	11.69
clock		61.60	56.29
sink		42.15	39.18
toilet		68.91	67.93
microwave		54.92	46.54
snowboard		41.44	36.11
vase		58.31	49.77
car		52.67	50.62
truck		58.53	50.69
potted plant		23.63	18.66
boat		52.27	47.51
train		74.41	71.35
wine glass		42.44	44.50
bed		59.70	55.41
bottle		33.71	32.92
carrot		41.08	34.56
remote		55.54	45.46
scissors		34.93	35.09
toothbrush		23.33	11.73
sandwich		46.10	39.30
bench		45.13	42.24
keyboard	✓	63.61	58.48
mIoU		52.36	49.13
backpack		14.34	13.70
horse		75.23	73.90

Classes	High Co-occurrence	Split (IoU)	
		train	val
banana		68.76	65.86
bowl		37.40	30.22
zebra		84.74	85.13
person		78.07	77.47
airplane		73.95	73.05
couch		47.70	44.21
cell phone		57.34	57.92
broccoli		55.68	53.69
background		88.66	87.88
hot dog		56.77	46.51
orange		66.94	64.01
book		33.55	28.56
traffic light		50.63	50.71
apple		51.52	52.43
parking meter		73.06	65.36
cake		54.31	49.55
dog		74.83	71.49
bus		77.83	74.00
teddy bear		68.05	65.92
cow		78.59	74.79
tv		60.02	55.60
donut		59.12	60.08
bird		70.51	64.37
bicycle		51.92	50.10
motorcycle		72.53	71.14
umbrella		66.60	68.20
elephant		84.82	83.94
suitcase		57.28	52.40
giraffe		80.06	79.74
bear		83.63	80.61
fire hydrant		78.01	75.52
stop sign		79.88	80.11
sheep		75.95	74.52
refrigerator		61.11	57.89
pizza		71.53	70.15
cat		76.52	74.86
sports ball		34.39	28.63
oven		48.20	43.07
laptop		63.10	59.92
dining table		41.41	38.92
hair drier		0.00	0.00

Table A.2: Segmentation scores for **COCO** when the segmentation model (DeepLab-V1) is trained using the ground truth segmentation masks. Pairs of highly co-occurring classes have been identified. This data is used to generate the comparison ratios in Table 4.4 in Section 4.5, and is presented with the classes in the same order.

Number of Layers	Including/Excluding Patch Token CA	Specific Layers	IoU						
			baseball bat	baseball glove	fork	knife	mouse	keyboard	mIoU
MCTFormer+ (with CCT)			5.08	6.53	12.44	15.70	14.62	49.31	43.14
MCTFormer+ (without CCT)			4.56	8.78	14.80	19.75	19.36	49.47	44.69
Previous 3-layer model (layers 6, 7, 8)			7.13	12.85	14.61	19.82	27.28	51.11	44.49
2	✗	batch	3.84	10.34	13.70	17.89	23.54	49.83	44.22
2	✗	2, 7	5.34	11.06	13.36	18.14	26.60	49.55	44.32
2	✗	7, 10	5.35	10.96	13.11	17.41	25.12	49.67	44.30
2	✗	2, 4	4.74	10.96	13.33	17.83	24.64	50.00	44.39
2	✓	batch	4.41	9.49	14.05	17.63	23.51	48.80	44.26
2	✓	3, 4	5.20	8.69	14.50	18.79	25.49	48.97	44.25
2	✓	7, 9	6.38	10.48	13.53	17.71	25.63	48.71	44.12
2	✓	3, 9	5.02	8.75	13.75	18.27	24.63	49.36	44.27
3	✗	batch	5.58	12.06	14.31	18.42	24.27	51.06	44.27
3	✗	3, 5, 6	6.11	13.24	14.96	18.71	27.05	50.78	44.46
3	✗	3, 7, 9	6.38	12.07	14.03	18.56	25.77	50.38	44.32
3	✗	4, 5, 7	5.68	13.00	14.72	18.63	26.75	51.05	44.41
3	✓	batch	5.58	10.55	14.35	18.98	24.16	50.58	44.28
3	✓	3, 9, 10	7.52	12.86	14.59	18.81	25.55	50.18	44.25
3	✓	4, 6, 7	6.40	13.36	14.95	19.43	25.71	51.13	44.24
3	✓	2, 3, 8	7.62	12.89	13.73	19.28	27.82	49.51	44.17
5	✗	batch	4.81	10.24	12.81	17.43	25.51	50.05	44.32
5	✗	1, 6, 7, 8, 10	5.01	10.81	13.21	17.82	26.47	50.60	44.35
5	✗	3, 4, 6, 8, 10	5.07	11.44	13.30	17.74	26.65	49.95	44.39
5	✗	1, 3, 4, 9, 10	4.64	9.42	13.52	17.98	23.16	50.99	44.27
5	✓	batch	5.02	8.75	15.13	18.78	24.61	49.07	44.17
5	✓	1, 4, 6, 7, 9	5.96	9.55	15.36	19.37	25.99	50.36	44.14
5	✓	3, 4, 5, 9, 10	5.24	9.31	14.42	18.64	24.45	49.16	44.30
5	✓	1, 2, 3, 5, 7	6.04	10.65	15.49	18.90	25.63	49.22	44.01
10	✗	1, 2, ⋯, 10	4.64	15.15	14.67	18.92	25.36	47.02	44.20
10	✓	1, 2, ⋯, 10	6.47	12.84	15.09	19.59	26.68	47.21	44.10
11	✗	0, 1, ⋯, 10	4.68	14.74	14.49	19.02	24.63	46.44	44.07
11	✓	0, 1, ⋯, 10	5.85	11.97	15.49	20.16	26.60	46.09	44.08

Table A.3: The experimental data for cross attention applied to various numbers of layers, where the specific layers are randomly chosen at the beginning of training (the "batch" value in Specific Layers indicates a new set of random layers was chosen for each batch). The runs also vary by their inclusion or exclusion of the query vectors corresponding to patch tokens during cross attention (CA). The first 3 rows are baseline comparisons. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest. This COCO data supports Figure 5.22 in Section 5.2.8.

Specific Layers	Including/Excluding Patch Token CA	Contrastive Learning	Early Stopping	IoU						
				baseball bat	baseball glove	fork	knife	mouse	keyboard	mIoU
MCTFormer+		✓	✗	5.08	6.53	12.44	15.70	14.62	49.31	43.14
MCTFormer+		✗	✗	4.56	8.78	14.80	19.75	19.36	49.47	44.69
Previous 3-layer model (6, 7, 8)		✓	✗	7.13	12.85	14.61	19.82	27.28	51.11	44.49
Previous 3-layer model (3, 5, 6)		✓	✗	6.11	13.24	14.96	18.71	27.05	50.78	44.46
7, 8, 9	✗	✗	✗	5.93	11.05	12.31	15.44	25.58	45.88	42.91
7, 8, 9	✗	✓	✗	5.37	11.98	13.92	16.95	25.10	47.12	43.43
7, 8, 9	✓	✗	✗	6.27	11.53	12.78	15.26	26.09	47.21	42.83
7, 8, 9	✓	✓	✗	5.43	10.85	14.07	16.48	25.33	48.48	43.20
8, 9, 10	✗	✗	✗	5.65	11.81	12.14	15.77	26.27	45.55	43.08
8, 9, 10	✗	✓	✗	4.83	10.78	13.32	17.14	25.31	46.73	43.48
8, 9, 10	✓	✗	✗	6.11	11.35	11.76	15.08	25.29	46.75	43.01
8, 9, 10	✓	✓	✗	5.12	10.32	13.53	17.08	24.15	48.12	43.42
7, 8, 9	✗	✗	✓	6.17	15.31	13.96	19.73	24.31	47.91	44.16
7, 8, 9	✗	✓	✓	6.40	14.15	14.22	19.75	22.66	49.38	44.17
7, 8, 9	✓	✗	✓	6.46	14.87	14.36	19.51	24.18	49.18	43.95
7, 8, 9	✓	✓	✓	5.87	13.17	13.55	19.77	21.85	49.50	43.90
8, 9, 10	✗	✗	✓	5.78	14.91	13.18	19.97	23.91	47.24	44.07
8, 9, 10	✗	✓	✓	5.26	12.36	13.86	20.43	22.55	49.37	44.21
8, 9, 10	✓	✗	✓	5.90	14.20	13.25	19.64	24.28	47.79	43.96
8, 9, 10	✓	✓	✓	5.13	11.46	13.76	20.02	21.83	48.99	44.00

Table A.4: The experimental data for cross attention applied to the triple-branch version of our framework. It is applied to two main layer groupings (6, 7, 8, and 7, 8, 9), which is consistent with Section 5.2.6 (and its following sections), a decision which is inherited as the best performers from Section 5.2.4. The runs also vary by their inclusion or exclusion of the query vectors corresponding to patch tokens during cross attention (CA), and their inclusion or exclusion of contrastive learning. The models all use "turnoff" mode for cross attention — in the case where there is no co-occurring class pair in the primary image, no cross attention at all will take place. The first 4 rows are baseline comparisons. The final 8 rows (where "early stopping" is performed) are the results from taking the checkpoint after Epoch 25, instead of the one after Epoch 45 — these are included to address overfitting concerns. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest. This COCO data supports Figure 5.23 in Section 5.2.9.

Model	Including CCT	Classes (IoU)		
		chair	diningtable	mIoU
MCTFormer+	✓	35.03	57.62	67.04
	✗	36.02	57.04	66.80
DB Inter - Layer 8	✓	32.57	45.03	61.62
	✗	34.69	47.59	62.45
DB Union - Layer 11	✓	36.50	49.41	61.99
	✗	36.40	47.96	61.22

Table A.5: The experimental data for models trained with and without CCT on **VOC**. It is applied to two high-performing single-layer cross image attention models from Section 5.2.3. A baseline comparison to MCTFormer+ with and without CCT is also included. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest. This data supports Figure 6.4 in Section 6.3.3.

Model	Include CCT	Classes (IoU)						
		baseball bat	baseball glove	fork	knife	mouse	keyboard	mIoU
MCTFormer+	✓	5.08	6.53	12.44	15.70	14.62	49.31	43.14
	✗	4.56	8.78	14.80	19.75	19.36	49.47	44.69
DB Inter - Layer 8	✓	5.95	5.82	13.24	15.92	14.95	47.87	42.74
	✗	7.66	10.90	13.89	18.27	19.73	51.68	43.88
DB Union - Layer 7	✓	4.06	4.05	14.30	16.15	11.20	46.01	41.25
	✗	7.88	8.12	12.47	17.39	14.80	46.41	40.06
DB Union - Layer 11	✓	3.73	7.09	17.42	17.17	12.56	44.83	41.73
	✗	5.08	10.80	13.70	17.19	13.51	46.19	39.97

Table A.6: The experimental data for models trained with and without CCT on **COCO**. It is applied to three high-performing single-layer cross image attention models from Section 5.2.3. A baseline comparison to MCTFormer+ with and without CCT is also included. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest. This data supports Figure 6.5 in Section 6.3.3.

Co-occurrence Threshold	Turnoff Mode	Specific Layers	CCT and CCT Prime	Classes (IoU)						
				baseball bat	baseball glove	fork	knife	mouse	keyboard	mIoU
MCTFormer+			✓	5.08	6.53	12.44	15.70	14.62	49.31	43.14
MCTFormer+			✗	4.56	8.78	14.80	19.75	19.36	49.47	44.69
0.2491	✓	6, 7, 8	✗	7.80	13.79	12.34	18.02	26.95	50.63	44.01
0.2491	✓	7, 8, 9	✗	9.10	13.07	12.52	19.02	25.80	50.62	44.15
0.2491	✗	6, 7, 8	✓	2.97	4.79	11.12	15.41	14.59	45.34	41.37
0.2491	✗	7, 8, 9	✓	4.30	5.02	14.39	16.25	16.58	47.55	41.99
0.2491	✓	6, 7, 8	✓	3.94	5.56	16.88	17.22	18.91	48.05	42.07
0.2491	✓	7, 8, 9	✓	4.98	3.98	9.51	13.80	9.43	34.59	42.08
0.46	✗	6, 7, 8	✓	3.00	6.30	13.13	15.87	15.95	44.27	41.40
0.46	✗	7, 8, 9	✓	3.02	3.23	13.96	16.76	11.96	41.28	41.64
0.46	✓	6, 7, 8	✓	3.15	6.38	11.75	17.40	18.70	47.52	42.41
0.46	✓	7, 8, 9	✓	3.09	7.41	12.67	16.74	18.32	48.98	41.53

Table A.7: The experimental data for CCT and CCT Prime being enabled in CIA-AP models (using the Mean augmentation method). These model configurations vary by the co-occurrence threshold, the enabling of "turnoff" mode, and the specific layers to which the cross attention is applied. The first two rows are baseline comparisons to MCTFormer+ with and without CCT, respectively. The third and fourth rows are baseline comparisons to equivalent models trained using CIA-AP with a co-occurrence threshold of 0.2491, "turnoff" mode enabled, differing groups of layers to which the cross attention is applied, and **CCT and CCT Prime disabled**. The IoU values are shaded for ease of interpretation: from red for the smallest value in the column to green for the largest. This COCO data supports Figure 6.8 in Section 6.3.5.