Self Supervised Salient Object Detection using Pseudo Labels

School of Computer Science & Applied Mathematics
University of the Witwatersrand

Kidhar Bachan
2014927

Supervised by Dr Hairong Wang

August 7, 2023

A research report submitted to the Faculty of Science, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Masters of Science

Abstract

Deep Convolutional Neural Networks have dominated salient object detection methods in recent history. A determining factor for salient object detection network performance is the quality and quantity of pixel-wise annotated labels. This annotation is performed manually, making it expensive (time-consuming, tedious), while limiting the training data to the available annotated datasets. Alternatively, unsupervised models are able to learn from unlabelled datasets or datasets in the wild. In this work, an existing algorithm [Li *et al.* 2020] is used to refine the generated pseudo labels before training. This research focuses on the changes made to the pseudo label refinement algorithm and its effect on performance for unsupervised saliency object detection tasks. We show that using this novel approach leads to statistically negligible performance improvements and discuss the reasons why this is the case.

Declaration

I, Kidhar Bachan, hereby declare the contents of this research report to be my own work. This research report is submitted in partial fulfilment of the requirements for the degree of Masters 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.

Kidhar Bachan	August 7, 2023 Date

Acknowledgements

I thank Doctor Hairong Wang for her supervision, patience and guidance as well as for having the faith in me to undertake this arduous task. I thank Doctor Richard Klein for providing a solid foundation to me in the field of Computer Vision which contributed to me choosing a research topic in this field. I thank the remaining faculty members and my fellow students who I have interacted with as they have all contributed to this work. Lastly, I thank my wife for giving me the time and space to complete this study at the sacrifice of many hours of family time. This work could not have been completed without your understanding and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Salient" is defined in the English language as the "most notable" or "most important". Salient object detection is therefore concerned with identifying the most interesting visual information in a scene. In computer vision, an image is analysed for a salient object where pixel intensity represents the probability of that pixel's saliency. Applications of salient object detection (SOD) include background removal, motion prediction, image captioning, autonomous vehicle control systems, and anomaly detection. In early works, intensity, colour, and orientation were hard-coded and used as image features [Itti *et al.* 1998], while current deep models learn the optimal features. Deep models are categorised into either unsupervised or supervised techniques. Supervised techniques are named as such as they require large, accurate, human-annotated training labels which are consequently time-consuming to generate. Given an input image, the process of manually annotating labels consists of first identifying what the salient object is and then selecting all pixels which contribute to that salient object. The output of such annotation process is given as a saliency mask or saliency map. Some examples of input images and their saliency maps are shown in Figure 1.1.

Figure 1.1: Some examples from the MSRA-B dataset [Wang *et al.* 2017] which show the detected salient object in a bounding box [top row] and the corresponding saliency mask [bottom row] for each image.

An alternative approach for generating saliency masks is to infer pseudo labels from a given image. In Lee [2013], pseudo labels are defined as target classes for unlabelled data as if they were ground truth labels. Pseudo labels refer to pixel-wise saliency maps that have not been annotated by humans but have solely been produced by some computational method. A popular approach to generating pseudo labels is to use hard-coded image features to make saliency predictions. This approach, however, leads to unsatisfactory results that incentivise the trained model to learn simple features instead of capturing deeper saliency patterns.

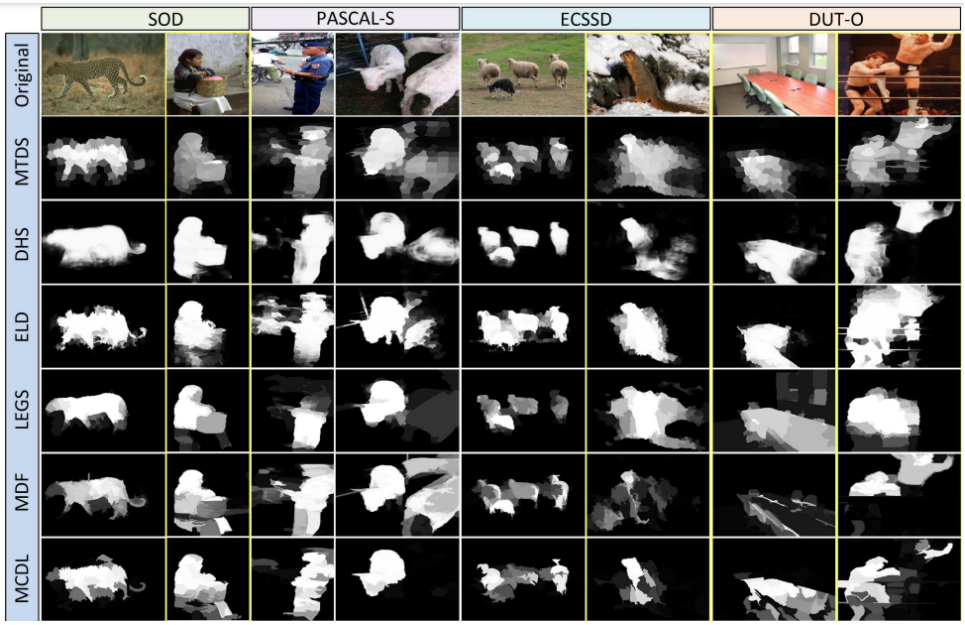

Figure 1.2: Examples from four datasets and six hand-crafted saliency prediction models which anecdotally shows the variation in label predictions (noise) among the selected methods [Dingwen *et al.* 2017].

Figure 1.2 depicts the blurriness of the pseudo labels when hard-coded image features are used to predict saliency. If a learning method were to use these labels directly, the noise in the labels would cause the network to generalise too much and thereby result in unsatisfactory predictions. One way to improve the labels is to apply a refinement process to them. Dingwen *et al.* [2017] proposed Supervision by Fusion in which the output of multiple handcrafted methods are fused together to form a single refined saliency prediction. Using Deep Unsupervised Saliency Detection, proposed by Zhang *et al.* [2018b], the noise from the handcrafted methods is explicitly modelled and jointly optimized, together with the saliency prediction network. During prediction, the noise module is removed, which results in noise-free but smooth predictions. In both approaches, noisy handcrafted saliency predictions are used directly as supervision during training, resulting in sub-optimal predictions. The above methods take different approaches to solving the noisy pseudo label problem, Dingwen *et al.* [2017] refines the pseudo labels in order to minimises the noise while Zhang *et al.* [2018b] models the noise in the pseudo label.

In our proposed method, we modify an existing label refinement model, developed for object classification, to refine salient object pseudo labels.

This study focuses on the effect that refining pseudo labels will have on the performance of a salient object detection network. The research aim is to quantify the change in performance, if any, that this type of refinement makes while the objective is to develop a novel approach to unsupervised salient object network training by combining the methods of Deep Robust Unsupervised Saliency Detection [Nguyen *et al.* 2019a] and DivideMix : Learning with Noisy Labels as Semi-supervised Learning [Li *et al.* 2020].

To provide a background to the SOD problem, a discussion of the evolution from the earliest hand-crafted methods to the latest deep learning methods is presented in Chapter 2. In Chapter 2, we also discuss relevant works on learning with noisy labels. Focus will be given to the algorithms that directly impact this work, namely Deep Robust Unsupervised Saliency Detection with Self Supervision [Nguyen *et al.* 2019a] and DivideMix [Li *et al.* 2020]. We describe the research aim, datasets, implementation details and research hypothesis in Chapter 3. The details of the experimental setup and obtained results are discussed in Chapter 4. We conclude this report by discussing the limitations of this study as well as future routes to pursue in Chapter 5.

Chapter 2

Background and Related Work

2.1 Introduction

Here, we discuss commonly used salient object detection terminology. In Section 2.1.1 we discuss Image Features by way of example and how they were used in some early handcrafted methods to detect salient objects. Handcrafted methods utilise hard-coded feature selection, and can therefore result in suboptimal saliency prediction results. Conversely, Convolutional Neural Networks (CNN) learn the best features to select per dataset. In Section 2.4, two popular Convolutional Neural Networks (CNN) architectures are discussed, namely Residual Network and Dilated Residual Network [Yu *et al.* 2017]. CNN-based models currently dominate the salient object detection landscape, and we discuss how different methods employ them in Section 2.6. A foundational building block in salient object detection is that of image features. Image features were originally hardcoded and can be edges, corners or differences in contrast, amongst others.

2.1.1 Image Features

An image feature can be thought of as something that describes a part of an image. Features such as corners or edges help to detect the boundaries of objects, but these cannot be used in isolation to detect the salient objects.

In handcrafted methods, features such as intensity, orientation, colour, and contrast were used and often combined to predict salient objects with greater accuracy than basing the prediction on a single feature. A common feature in early methods was that of contrast and was widely used as it simulates the human visual receptive field. In Liu *et al.* [2007], linearly combining the contrast features of the Gaussian Image Pyramids is used to generate a single multiscale contrast feature, as seen in Figure 2.1. This multiscale approach is common in current literature as it identifies local (low-level) and global (high-level) features.

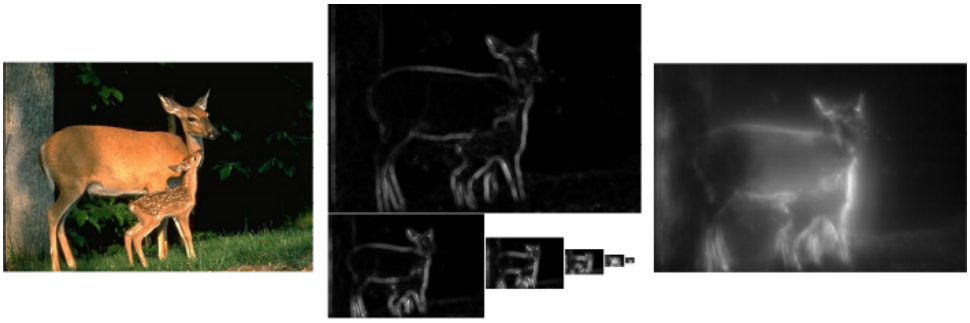

Figure 2.1: Input image [left] and the Gaussian pyramid [middle] showing regions of high contrast and the resulting multiscale fused contrast feature [right] [Liu *et al.* 2007]

2.1.2 Conditional Random Field (CRF)

In a Conditional Random Field (CRF) we have **X** as an input vector (eg. of image features) and we want to predict an output vector **Y** (labels), given **X**. The goal during training of the CRF is to find the best labeling configuration **Y** that maximises the conditional probability $P(Y \mid X)$. In image segmentation, a CRF uses the labels of pixels around the current pixel to make a prediction of the current pixels' label.

2.2 Convolutional Neural Networks (CNN)

Any network that consists of a convolutional layer is called a Convolutional Neural Network (CNN). The network aims to learn the filter or kernel weights that result in detecting the features which can be used in image segmentation, salient object detection or object classification, amongst others. In the input layer, a two-dimensional (Width x Height) image with three colour channels (red, blue, green) is fed into the network. Thereafter, a series of convolution + Rectified Linear Unit (ReLU), max pooling, normalisation and fully connected layers contribute to the final architecture of a given CNN. A description of these major components is discussed in the next subsection.

2.2.1 Convolution layer + ReLU

In this layer, an input tensor and a kernel are convolved which produces an activation or feature map, as in Figure 2.2. In the context of this work, the input tensor is the two-dimensional input image used to train the network, while the kernel values are parameters that will be learned in order to extract salient objects from the input images.

The height and width of the kernel is a hyperparameter, which means that it has to be decided beforehand. Another hyperparameter, depth, is determined by the number of kernels that we want to learn and is also referred to as the number of channels. After convolution, non-linearity is introduced by way of an activation function, the most popular of which is called the rectified linear unit (ReLU): $f(x) = max(0, x)$.

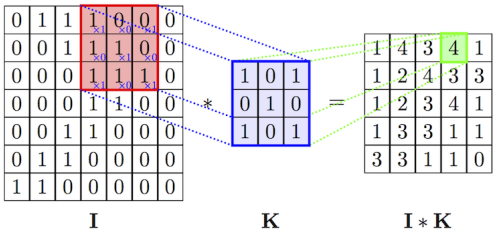

Figure 2.2: A convolution operation takes place between an input image [I] and a Kernel [K] to produce an activation map of output [I * K] [S. Mohamed 2017]

2.2.2 Pooling layer

While there exists various pooling methods (global, minimum, maximum and average), the two types often employed in CNNs are average pooling and max-pooling, see Figure 2.3 for an example.

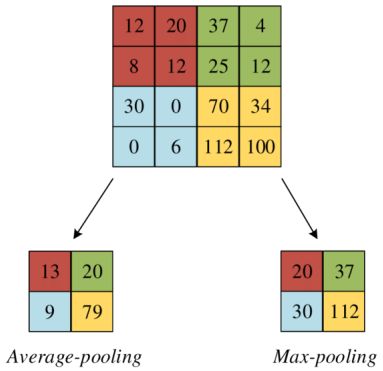

Figure 2.3: A diagrammatic representation showing the difference between average-pooling and max-pooling when using a 2×2 filter with a stride of 2 [S. Mohamed 2017]

Pooling works in the following way

- Take a small window (2-by-2 or 3-by-3 window) and slide it over the input data.
- Find the maximum (or average in the case of average pooling) value (most prominent feature) within each window.
- Place the maximum (or average in the case of average pooling) value in the corresponding location in the output.
- Move the window to the next location based on the stride, and repeat the process until the entire input is covered.

Both max-pooling and average-pooling reduce dimensionality, which results in fewer parameters to learn and therefore less computation cost. Max-pooling is better suited to detecting features and is the most common type of pooling used in CNNs.

2.2.3 Dilation

While the operations of convolution and pooling reduce the spatial resolution of the receptive field, dilation can be used to maintain or increase it. In a standard convolution operation, each element of the filter is multiplied with the corresponding element of the input, to compute the output. The dilation parameter sets the spacing between the filter elements, thus increasing the receptive field and allowing the filter to capture information from a larger area at no increase in computational cost as seen in Figure 2.4. The benefits of dilation include:

- Dilation allows filters to have a larger view of the input data, which can be beneficial for recognizing larger patterns or features.
- Dilated convolutions capture fine-grained details while still maintaining a broader view, which means that both local and global information are retained.
- Dilation can be used to reduce the downsampling effect of pooling layers, thus preserving more spatial information throughout the network.

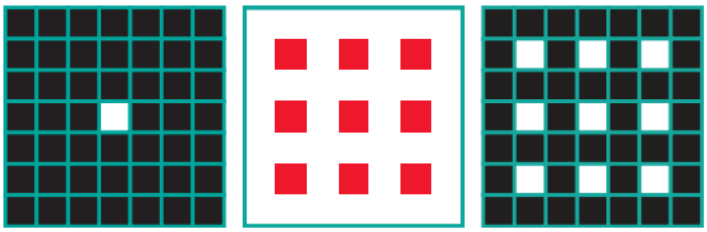

Figure 2.4: Dilation applied with a factor of 2 to the filter (middle). After the input (left) undergoes convolution with the dilated filter, the receptive field dimensionality remains the same, as it was before the operation (right) [Yu *et al.* 2017].

2.2.4 Dropout

Regularization can be achieved by means of dropout which randomly turns off a percentage of the neurons to nullify their influence [Srivastava *et al.* 2014], see Figure 2.5 for an example.

CNNs generally learn the most information from the initial training rounds, which can result in over-fitting and makes learning later on less effective. Dropout aims to mitigate the influence that early learning has on the network by applying a dropout mask to a layer's output. The mask randomly switches off part of the weights of the network to zero. The fraction of neurons to be turned off is a hyperparameter that normally ranges from 0.2-0.5 and determines the percentage of weights that are set to zero per layer. The dropout mask is also randomly generated for each sample and iteration thus reducing the dependence on specific neurons. During inference, the dropout mask is not applied, however the weights of the neurons that were dropped during training are

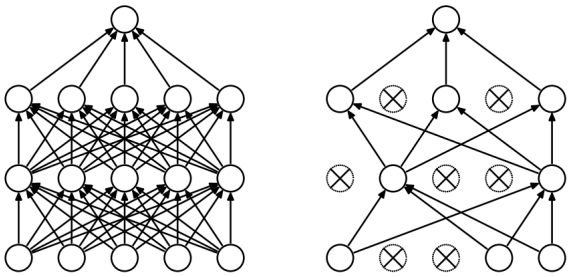

Figure 2.5: A network without dropout on the left and one with dropout on the right![Srivastava *et al.* 2014].

scaled down to ensure consistency of the expected values. The advantages of dropout are

- Regularisation - the network is less likely to memorise data and more likely to generalise features.
- Each iteration uses different neurons which is synonymous with training multiple networks.
- When dropout is applied, the likelihood of neuron dependence is reduced which means that specific neurons do not need to compensate for others.

2.2.5 Batch Normalisation

During the training of a CNN, the distribution of data at each layer in the CNN can change as the learnt parameters change. Batch normalisation mitigates this effect by normalising the input data to each layer to have a zero mean and unit variance. It has been shown to speed up learning as well as provide regularization [Ioffe and Szegedy 2015]. The following is applied to the input of each neuron in a layer during batch normalisation.

- Normalization: Calculate the mean and standard deviation of the input over the mini-batch.
- Centering and Scaling: Subtract the mean, then divide by the standard deviation.
- Scaling and Shifting: Learn the optimal parameters for scaling and shifting for each feature.

2.2.6 Residual Connections

Residual or skip connections help to transfer low-level feature information from early network (higher resolution) layers to later (lower resolution) layers [He *et al.* 2015], without adding extra parameters or increasing computational complexity. After convolution and max-pooling, finer details are lost, and side connections can be used to re-introduce the lost information. Residual connections were developed in order to solve the vanishing gradient problem, which occurs when gradients become very small

as they are back-propagated through the network. The residual connections allows the network to learn the difference between the output and input of a layer instead of learning the expected output. Implementing a residual connection involves the following.

- An input *x*, flows through one or more layers resulting in a feature representation *F(x)*
- *x* is added back to the feature representation *F(x)*
- *x + F(x)* is passed through the layers to produce the final output of the residual connection.

2.3 Optimizers

2.3.1 Stochastic Gradient Descent (SGD)

In traditional gradient descent, the entire dataset is used to compute the gradient of the loss function with respect to the parameters in each iteration. When dealing with large datasets, this can be computationally expensive and time-consuming. SGD addresses this issue by randomly sampling a subset of the data, known as a mini-batch, to compute an estimate of the gradient in each iteration. Since the sampling of the subset of the data is random, the gradient estimates can be noisy and do not always provide an accurate representation of the true gradient. To reduce the noise in the estimates, momentum and adaptive learning rates can be used.

2.3.2 Adaptive Moment Estimation (ADAM)

The ADAM algorithm maintains an adaptive learning rate for each parameter by estimating the first and second moments of the gradients. The first moment (momentum) is the average of the gradients, while the second moment is the average of the squared gradients. The first moment helps to smooth out the updates by accumulating a fraction of the previous gradients. The second moment helps adjust the learning rate (step size) for each parameter based on the magnitude of the gradient.

2.4 Convolutional Neural Network Architecture

Different combinations of the previously discussed building blocks result in unique CNN architectures. In the following section, two relevant network architectures are discussed, namely the Residual Network (ResNet) and the Dilated Residual Network (DRN).

2.4.1 Residual Network

Residual Networks [He *et al.* 2015] were studied to solve the problems associated with training very deep networks, like the vanishing gradient problem. Residual networks

and residual connections which allow the network to learn the difference between the input and the desired output. The residual connection provides a shorter path for the gradient to flow during back propagation, and an example is shown in Figure 2.6. Each convolution block represents a convolution, ReLU and batch normalisation operations. This architecture improved network performance of classification tasks without increasing complexity when compared to a similar sized CNN. Due to the good performance on object detection and classification tasks, this architecture was chosen as the backbone for the Dilated Residual Network, which will be discussed in the next section.

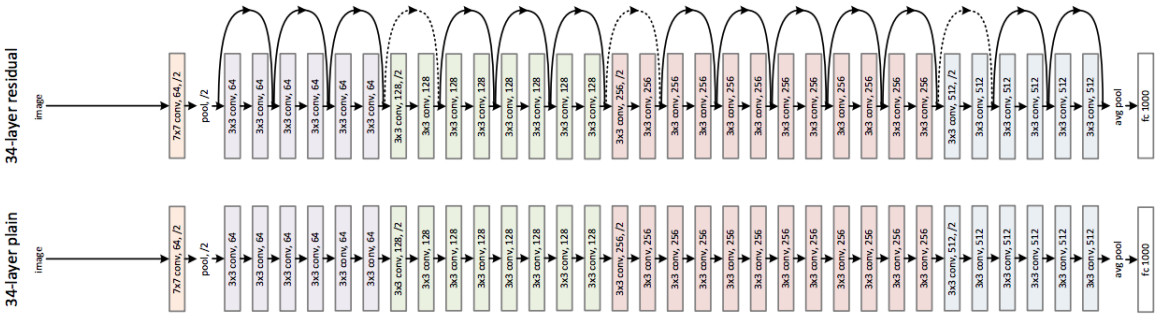

Figure 2.6: A 34-layer plain CNN architecture (bottom) vs a 34-layer ResNet architecture (top), showing the short connections [He *et al.* 2015].

2.4.2 Dilated Residual Network

The Dilated Residual Architecture adds dilation to some layers in the Residual Network architecture. Dilation is used to preserve spatial resolution through to later network layers, without increasing complexity. Three DRNs with different configurations are shown in Figure 2.7.

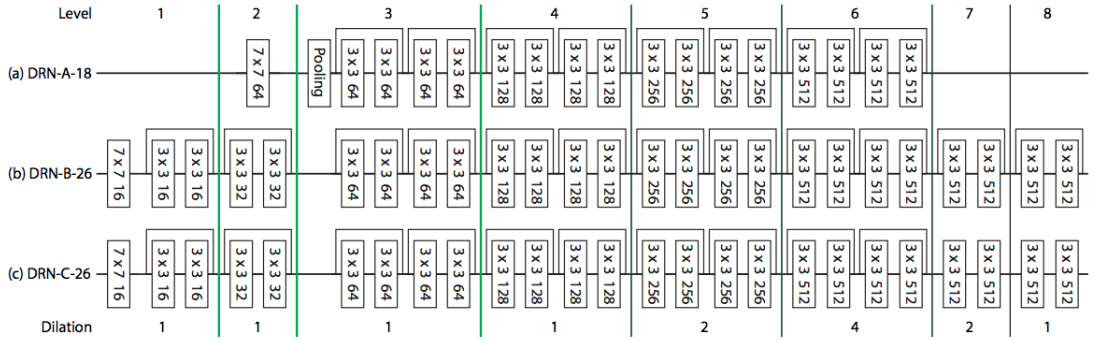

Figure 2.7: Three DRN architectures are shown above, namely DRN-A, DRN-B and DRN-C. DRN-C effectively reduces "gridding" artifacts that are introduced during dilation [Yu *et al.* 2017].

In Figure 2.7, each block consists of a ReLU, batch normalisation and a convolution operation. Filter size is specified by the number (eg. 7×7), followed by the number of

channels (eg. 16). The green lines represent that a stride of 2 is used during convolution, thus resulting in downsampling. The dilation per level is shown in the bottom of the figure.

2.5 Other Terminology

2.5.1 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) represents data as a combination of several Gaussian distributions, each representing a different group or cluster. The GMM tries to find the parameters (mean, variance, and weights) of these Gaussian distributions that best fit the given data. In our model, a GMM uses the predictions loss to separate pseudo labels into noisy or clean groups.

2.5.2 Simple Moving Average

The Simple Moving Average (SMA) is calculated by adding up a specific number of data points in a series and then dividing that sum by the number of data points. For example, a 5-point SMA would be the sum of the last five data points divided by 5. The SMA gives equal weight to each data point in the period.

2.5.3 Image Augmentations

Image augmentation deals with augmenting image data and involves adjusting the color, adding Gaussian blur and Gaussian noise in order to increase the dataset's variability. Increasing the variability of a dataset often leads to better generalisation of the learned model.

2.6 Related Work

The work related to this study is discussed in this section, and it is mainly based on the two modules in the implemented method, namely the salient object detection module (Module A) and the learning with noisy labels module (Module B). In Sections 2.6.1, 2.6.2, and 2.6.3, we discuss salient object detection methods starting from earlier handcrafted methods to recent approaches including supervised and unsupervised CNN-based methods. The methods of learning from noisy labels are then discussed in Section 2.6.4,.

2.6.1 Early Salient Object Detection methods

Early work by Itti *et al.* [1998] was modelled after the primate visual system. In that work, nine spatial scales using Gaussian pyramids are created from input images of 640 x 480 resolutions. The images then undergo a process of linear filtering to determine

the maxima of colour, intensity, and orientation. The maxima of the map (after combining the maxima of colour, intensity and orientation) suggest the area to focus on. The focus areas are fed into a "winner takes all" neural network to determine the focus of attention areas over time, as is the case when a human sees an image.

We can also think of saliency detection as a problem in which an object must be separated from its background. This approach was applied in Liu *et al.* [2007]. Liu *et al.* [2007] focused on generating handcrafted features which are then used to learn a Conditional Random Field [Lafferty *et al.* 2001] and eventually to predict a bounding box around the salient object. Achanta *et al.* [2008] used a contrast detection filter which is generated by measuring the difference in the feature vectors (colour and luminance) of a region, R1, when compared to its neighbouring region, R2, at differing scales. The pixel-wise saliency values across scales are then summed to produce the final saliency map.

Robust Background Detection [Zhu *et al.* 2014] introduces *boundary connectivity*, which is defined as the percentage of a superpixel's perimeter over that superpixel's area. A superpixel is a group of pixels that have some similarity (eg. contrast) with one another. The rationale behind this is that salient objects should not cross the image boundary, and if they do, minimal boundary crossing is present. In Li *et al.* [2013], a background template is generated by using the boundary regions, and the reconstruction error of the dense and sparse regions is calculated. For the foreground, a dense region with a large reconstruction error is present, while a large reconstruction error in a sparse region can be used to detect the background.

In Absorbing Markov Chains [Jiang *et al.* 2013], an image is partitioned according to its superpixels and an absorbing Markov Chain is calculated with absorbing nodes as the boundary superpixels. The similarity of the transient node to a boundary node is represented by the absorbed time. The absorbed time metric can be used to separate the superpixels into foreground or background and can be used to produce the saliency mask.

In Zou and Komodakis [2015], an image is segmented into regions based on the contours in the image. The regions are then merged into new regions called a super region and the super regions are fed into a 7-layer CNN for feature extraction. The prominent features are then used to select regions and super regions which will comprise the salient object.

The performance of the early methods are confined to, and dependent on, the selection of the hard-coded hand-crafted features. These methods yielded satisfactory results, although a more dynamic approach of feature selection (CNN based) was required in the next generation of saliency detection frameworks.

2.6.2 Layered based Approaches

One approach to discovering salient objects is detecting where they are and then finding the object's border. Liu and Han [2016] solved this problem by first applying a global view CNN to detect a coarse salient object. A recurrent CNN then focuses on and

refines the coarse saliency map into finer detail. An alternative approach to locating the salient object is to fuse multi-level features. One of the earliest methods to use this approach is Hou *et al.* [2016]. Hou *et al.* [2016] employs residual connections between convolution layers. Deeper layers can detect salient regions, while the shallower layers focus on lower level features. Combining both of these feature maps paved the way for future methods to benefit from this aggregation approach. Zhang *et al.* [2017] is one such example. Features at multiple scales are extracted by a pre-trained CNN and either extended or shrunk to combine the multi-resolution feature maps in the next stage. Deep recursive supervision is then performed to train an edge detection [Xie and Tu 2015] network, and a boundary refinement module converts the edge detection prediction into a saliency map. Wang *et al.* [2018] employed a feature weighting module that takes residual outputs from the convolution layers and learns a weight for each pixel, which eventually produces a weighted spatial representation. This representation passes through a boundary refinement network to recover some of the lost details and sharpen the salient object's outline. In Wei *et al.* [2019] a CNN is used to extract features at multiple layers while a cross-feature module, merges the multi-layer features after refining them. A weighted Binary Cross Entropy Loss function is used in Wei *et al.* [2019], where a pixel is given a weighting proportional to the difference between it and the surrounding pixels. A similar argument is made for using a weighted Intersection over Union, and both of these weighted parameters are used to generate the loss function. In Pang *et al.* [2020] multi-level and multiscale features are used to generate a more accurate saliency map. Their work also showed that their novel loss (Consistency Enhanced Loss) function, better discriminates between the foreground and background and leads to a further performance improvement in all of the measured performance metrics.

Qin *et al.* [2019] learns the difference between the ground truth and the coarse saliency map. Two modules comprise the network, namely, the prediction module and a module that works on refining the residual. During the prediction module, the features are encoded and decoded to produce the coarse (blurry and noisy) saliency map. The residual refinement module is then supervised while learning the difference between the ground truth and the coarse map.

2.6.3 Unsupervised Approaches

In Dingwen *et al.* [2017]'s seminal paper Supervision by Fusion, multiple weak handcrafted salient models are combined to generate useful salient object annotations. Dingwen *et al.* [2017] found that using a single combined salient object annotator, comprising of multiple weak salient object annotators, yielded poor results. This was due to the inherent noise as a result of the discrepancy between each object detector. To remove the noise from the predicted mask and thereby improve performance, [Dingwen *et al.* 2017] focused on creating a reduced noise training dataset. Three unsupervised weak salient object detectors generate a weak saliency map. The saliency maps from each annotator are broken down into superpixel regions and the mean value for each superpixel region is calculated. The distance between each weak superpixel region and the mean of all the superpixel regions (for each annotator) is then determined,

and if it exceeds a threshold, those regions are fused using the GLAD [Whitehill *et al.* 2009] method. This process results in superpixel-level fusion maps. In the next stage, the mean of the saliency maps for each image is calculated and then fused as for the intra-image stage, resulting in the average saliency map. The distance between the predictions and the superpixel and image saliency maps also allows the algorithm to determine the confidence of each fused map. The larger the distance, the lower the confidence. This enables the cost function to penalise less confident pseudo labels and promote higher confidence pseudo labels. The pseudo labels are then used to learn a CNN.

An alternate approach to minimising noisy (hand-crafted) pseudo labels, is to model it. In Zhang *et al.* [2018b]'s paper, the noise is explicitly modelled and jointly optimised together with the object saliency prediction model. A mapping from colour image to saliency map is learnt together with a Guassian noise model. The aim during training is for the model to fit the noisy saliency maps generated by the hand-crafted methods. During testing, the noise module is detached, resulting in a saliency predictor.

Instead of assuming a Gaussian noise distribution as in [Zhang *et al.* 2018b], in [Zhang *et al.* 2020] the noise distribution is modelled as a latent vector by a neural network, which can approximate any form of structural noise. A noise estimation module, as well as a clean saliency prediction module together, make the saliency prediction network, during the training phase, but only consists of the clean saliency prediction module during testing. A novel back propagation (alternating back propagation) algorithm [Han *et al.* 2016] is employed to learn the estimated parameters for both submodules using gradient ascent, while inferential back-propagation is used to deduce the parameters for the hidden noise vectors. The effect of this method is that noisy training data is not removed from the dataset. Xin *et al.* [2018] recognised that there is information sharing between contour detection and saliency map prediction. An existing, trained contour detection network is split into a contour branch and a saliency branch. The contour branch generates contour maps which are converted to saliency masks via a novel approach. In the saliency branch, the converted masks are used to simulate supervision. In addition, an opposite network is trained (i.e. saliency maps to contour masks) on a different dataset. This alternating training pipeline enables the effective sharing of information between the contour and saliency networks.

In Shin *et al.* [2022] neither handcrafted methods nor refining of pseudo labels occurs. Rather, image encoders are used to generate pseudo labels based on spectral clustering. The best salient mask is then chosen by a novel winner takes all voting strategy, which is then used as pseudo labels during training.

Yan *et al.* [2022] creates synthetic labels by taking salient objects and pasting them over background images. The model continuously learns from both the synthetic and real labels. The discrepancy between the synthetic and real labels is mitigated during a novel uncertainty-aware learning strategy.

In Lin *et al.* [2022] two biases, namely the contrast bias and the spatial bias, are reduced. Contrast bias makes images with similar contrast form contrast clusters and training is directed towards these clusters which are data-rich. Spatial bias occurs due

to the fact that the objects of interest are normally present in the centre of the image. This biases the model to look focus on the centre of the image when detecting salient objects. This work aims to reduce both of these biases and can be used to improve any other salient detection framework.

Deep Robust Unsupervised Saliency Detection Nguyen *et al.* [2019a], when published was the state-of-the-art unsupervised salient object detection network, boasting performance metrics comparable to fully supervised saliency methods. In order to achieve this four improvements over current techniques were employed. Firstly, instead of using the labels generated from hand-crafted methods directly, a DRN uses the hand-crafted predictions as labels and learns the transformation from raw images to pseudo labels. This means that every hand-crafted prediction increases the models knowledge of detecting salient objects. Secondly, due to the high noise amongst hand-crafted predictions, a moving average of predictions for each image is collected every epoch. This has the effect of smoothing out the prediction and thereby reducing noise. Thirdly, the final moving average predictions from the last step are used as labels, and the DRNs are trained until their moving average predictions are stable. Finally, the predictions from the DRNs (which replace each hand-crafted method) are fused together by taking the average of the predictions. Fusing the predictions at the last stage allows the strongest image priors (for each DRN prediction) to be retained in the final pseudo-label prediction. A detailed overview of the entire method is described below.

Deep Robust Unsupervised Saliency Detection Nguyen *et al.* [2019a], falls into the learning from pseudo labels' category of CNNs and can be broken up into four stages. The first stage works by learning a Dilated Residual Network (DRN) for each hand-crafted method. The outputs of a Conditional Random Field (CRF) are accumulated into Moving Average (MVA) predictions. An incremental refining stage retrains the initial DRN with the MVA predictions, as expected outputs. The training process stops when the Moving Average predictions have converged. In the last stage, the saliency predictions from each hand-crafted method are fused resulting in the final saliency predictions. These final saliency predictions are then used as ground truth labels during the training of the final network. To prevent dependence on a single hand-crafted method, multiple hand-crafted methods (*namely* Robust Background Detection [Zhu *et al.* 2014], Dense and Sparse reconstruction [Li *et al.* 2013], Absorbing Markov Chains [Jiang *et al.* 2013] and Hierarchy associated rich features [Zou and Komodakis 2015]) are fused together as in [Dingwen *et al.* 2017]. Each hand-crafted method is substituted with a Deep Neural Network (DNN). The DNN naturally infers inter-image information during the training process as opposed to directly using the hand-crafted methods which do not retain information across predictions. The output of the DNNs at this stage, are called pseudo labels and consists of a probability that pixel, p is part of the saliency map. To reduce noise, the saliency maps are binarized such that $p = 1$, if the probability for that $pixel > 1.5 \times average\_saliency$, or $p = 0$ otherwise. After each epoch, a large variance in the pseudo labels exists. To reduce these variations, MVA predictions are created by accumulating the outputs of a fully connected CRF. As the MVA is composed during training (and not after each epoch), the resulting saliency maps are more robust when compared to taking snapshots after each epoch. The saliency maps at this stage are further improved by using the MVA predictions from stage two to re-

train the DNNs of the hand-crafted models. The pixel-wise loss is back propagated until the MVA(k) predictions are similar to the MVA(k-1) predictions or, in other words, the MVA(k) and MVA(k-1) predictions have converged. Pseudo label diversity is increased by refining each pseudo label in isolation from one another. The maps are then fused and used during network training.

2.6.4 Learning with Noisy Labels

Noise in pseudo labels makes unsupervised learning challenging. The noise results in a larger loss value and therefore a larger correction of the weights and biases during training. Overfitting occurs and can be described as when the model starts memorising the training data instead of generalising to it. A common approach to avoid overfitting is to employ regularisation. Regularisation is any method that counteracts the overfitting of the model, to the data, normally by imposing a constraint and thereby making it generalise better to unseen data. For learning with noisy labels (LNL) there are two loss metrics that are commonly used for regularisation, namely entropy minimisation and consistency regularisation. Consistency regularisation is the process of adding noise or elastically deforming the input (random horizontal flips and crops) and is also known as *data augmentation*. Consistency regularisation enforces that an unlabelled example x and its augmented version, Augment(x) should be classified the same. Entropy minimisation can be enforced by using a loss term that minimises the entropy of a model $p_{model}(y|x;\theta)$, x is the model input while the model parameters are denoted by θ.

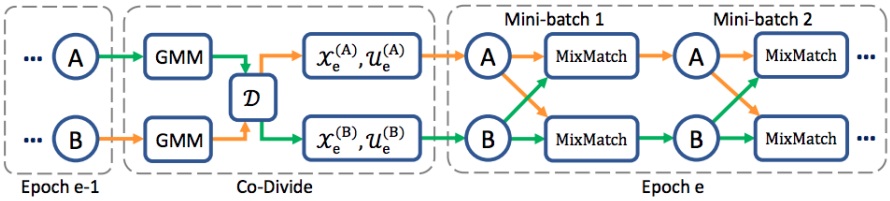

Figure 2.8: Architecture of DivideMix. Two networks A, B are trained simultaneously to be more robust to errors [Li *et al.* 2020]

DivideMix [Li *et al.* 2020] is one such method which considers the noise of pseudo labels and attempts to reduce it. A Gaussian Mixture Model (GMM) ([Permuter *et al.* 2006]) models the loss of each pseudo label and uses this model's prediction as a probability value to determine if a pseudo label is clean or noisy. The probability per pseudo label is then compared to a threshold in order to create two partitions, namely clean and noisy. Two models are trained independently and simultaneously (A, B as in Figure 2.8) to avoid confirmation bias. Each model calculates the cross entropy loss after a training iteration, which, after being modelled by a GMM, partitions the pseudo labels into clean and noisy pseudo labels for the other network [Li *et al.* 2020] until training is completed.

2.7 Conclusion

This chapter provided the relevant background and related work to formally define the research question, objective and hypothesis in the next chapter. The advancement of unsupervised and supervised salient object detection models were described in Section 2.6, as well as the various approaches that have been developed to improve saliency detection performance. In Section 2.6, we have also described the methods that are able to reduce noise in pseudo labels. The Research Methodology will be formulated in the next chapter.

Chapter 3

Research Methodology

3.1 Introduction

Having discussed the background and related work, we formally define the research aim and hypothesis in this chapter. Section 3.2 outlines the research aim. Section 3.3 gives the description of the datasets. The proposed method, that will accomplish the research objective, is discussed in Section 3.4.

3.2 Research Aim

There are a handful of unsupervised salient object detection models, as discussed in Section 2.6. In this study, we modify an existing algorithm that is designed for refining noisy labels for image classification task, so that it can be applied for refining noisy pseudo labels for the task of salient object detection. We then evaluate the effectiveness of our approach by comparing the performance of the proposed deep learning model trained on refined noisy pseudo labels against the same model but trained on pseudo labels without refinement.

3.3 Dataset Description

Three datasets were used during training and testing, namely MSRA-B [Wang *et al.* 2017], DUT [Lijun *et al.* 2017] and ECCSD [Shi *et al.* 2015]. MSRA-B [Wang *et al.* 2017] consists of 2500 images, with the dataset split into 500 validation images and 2000 training images. The labels in MSRA-B [Wang *et al.* 2017] consist of saliency masks. While these labels, also known as ground truth labels, are used during training of supervised approaches, they were not used during the training of our model. Our model generates its own labels which are called pseudo labels and these generated pseudo labels are used similarly to how ground truth labels would be used for supervised approaches. After training, the ground truth labels were used to calculate the prediction error and evaluate the model performance during testing. A few samples from MSRA-B dataset are shown in Figure 3.1.

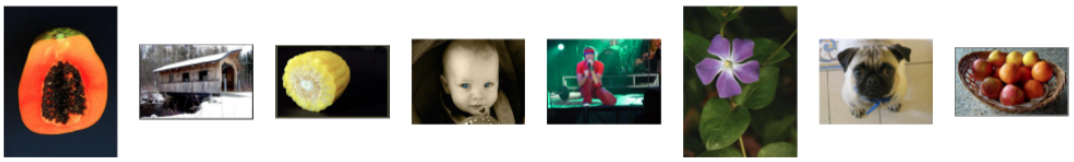

Figure 3.1: Random sample images from the MSRA-B salient object detection dataset [Wang *et al.* 2017]

The DUT [Lijun *et al.* 2017] dataset contains 5168 images, see Figure 3.2 for some sample images. Both MSRA-B and DUT datasets contain objects in various classes, with a single salient object in each image.

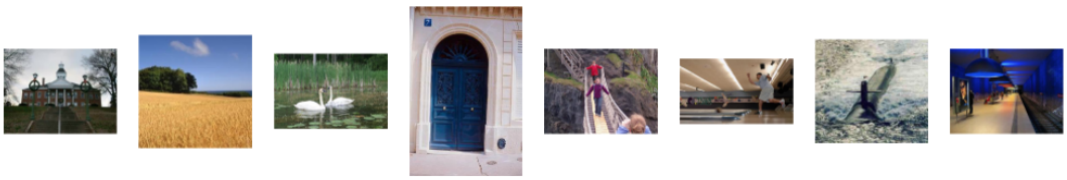

Figure 3.2: Random sample images from the DUT salient object detection dataset [Lijun *et al.* 2017]

Compared to the previous two datasets, the ECCSD [Shi *et al.* 2015] dataset consists of images with multiple objects. For model performance evaluation ECCSD was also used. Figure 3.3 contains random images from the ECCSD dataset.

Figure 3.3: Random samples from the ECCSD salient object detection dataset [Shi *et al.* 2015]

To evaluate the trained model, MAE and F_β-score are calculated using the model predictions and the ground truth labels. MAE is defined as

$$MAE = \frac{\sum_{i=1}^{n} |(y_i - x_i)|}{n},$$

where y_i is the predicted label and x_i is the ground truth. MAE measures the average error of the predictions versus the ground truth labels. F_β-score is defined as

$$F_\beta = \frac{(1+\beta^2)\textit{Precision} \times \textit{Recall}}{(\beta^2)\textit{Precision} + \textit{Recall}},$$

where

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives},$$

and

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}.$$

Precision and Recall are easier to understand with the aid of diagram shown in Figure 3.4. The hyperparameter β represents how much more important *recall* is with respect to *precision*, and is commonly set to $\beta^2 = 0.3$ for salient object detection networks. Saliency detection networks should aim to achieve higher F_β-scores and the lower MAE scores.

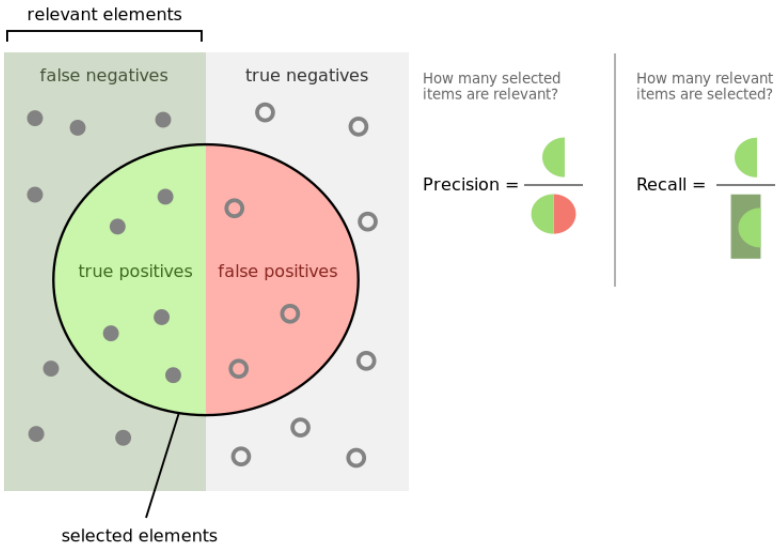

Figure 3.4: Precision and recall visualised [Walber 2023]

3.3.1 Learning curriculum

The images from the dataset are used to generate the pseudo labels as per Deep Robust Unsupervised Saliency Prediction [Nguyen *et al.* 2019a]. The generated pseudo labels are then further refined according to DivideMix [Li *et al.* 2020]. At this stage, the pseudo labels are partitioned into two partitions, namely clean and noisy. Each batch of pseudo labels is drawn randomly from the dataset and is comprised of both noisy and clean pseudo labels. The noisy pseudo labels are further refined and the total loss for the batch is made up of the loss for the noisy and clean pseudo labels. The batch that is drawn is random, that is we do not draw harder and harder samples as we progress through the dataset.

3.4 Implementation Details

We have implemented a novel framework that will detect salient objects in images. The framework consists of a pseudo label generator (Module A), followed by a noise-aware training module (Module B). An overview of the framework is shown in 3.5. The network entry point is a two-dimensional RGB 432×432 colour image with a saliency

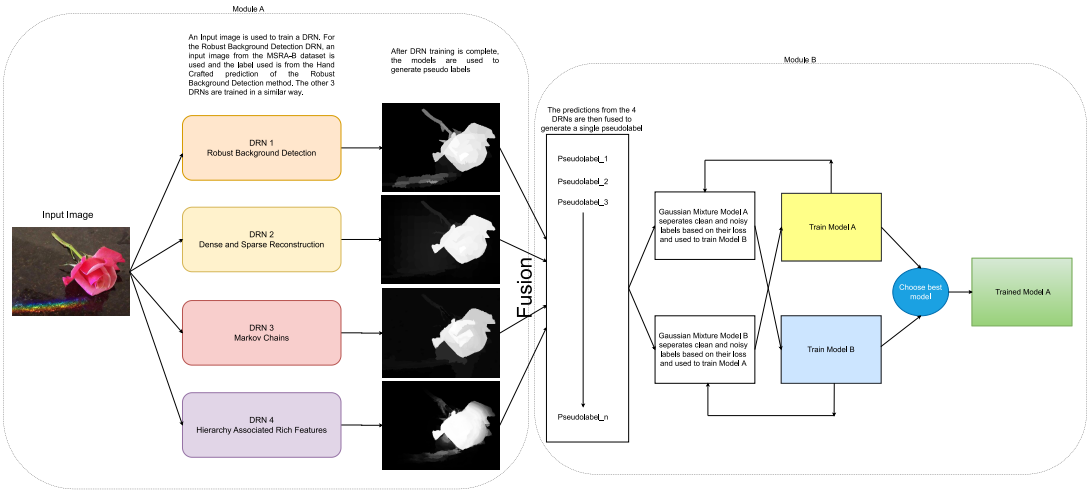

Figure 3.5: An overview of the implemented model

prediction as output. During training, four DNNs were learnt, one for each handcrafted method. Each handcrafted DNN (HC-DNN) was trained by using the predictions from the original handcrafted methods as targets. The MSRA-B [Wang *et al.* 2017] dataset (excluding the labels) was used during this process. The selected handcrafted methods were Robust Background Detection [Zhu *et al.* 2014], Dense and Sparse reconstruction [Li *et al.* 2013], Absorbing Markov Chains [Jiang *et al.* 2013] and Hierarchy associated rich features [Zou and Komodakis 2015] which were discussed in Section 2.6. After training the HC-DNNs (four in total), each HC-DNN used the MSRA-B [Wang *et al.* 2017] dataset as inputs in order to generate pseudo labels, that is, one set of pseudo labels for each HC-DNN. The input dimension of the dataset was [1 x 2500] while the pseudo label output was [4 x 2500], four, as there are four HC-DNNs. The four pseudo labels were then fused together, by taking their average, to generate the final pseudo labels [1 x 2500] that were used during training in Module B. At this point the pseudo label dataset contains both clean and noisy pseudo labels. As part of the DivideMix [Li *et al.* 2020] algorithm, the pseudo labels are required to be classified as either clean or noisy and was classified by modelling the loss, as calculated by the model, with a Gaussian Mixture Model (GMM). The probability for each image belonging to either the clean or noisy class is then predicted by the GMM and results in clean pseudo labels if the probability ≥ 0.5 or noisy pseudo labels otherwise. The probability returned by the GMM was also used as a weighting factor during label refinement, in order to combine the model's prediction with the clean pseudo labels. After training, performance was calculated by recording the MAE (Mean Absolute Error or L1-loss) and the F_β-score.

The majority of the changes were made in the DivideMix [Li *et al.* 2020] (Module B) algorithm as it was adapted from an image classification network to a saliency detection network. An image classification algorithm uses an input image to output a single class e.g. dog, bird, man, or woman whereas a pixel map of the main object is output in a saliency detection algorithm. The Loss function was changed from a Cross Entropy Loss to F_β-score. Augmentations (Noise, Jitter and Blur) were applied to the input and the data loader was modified to handle pixel-wise labels in an image instead of per image classes.

To better explain the complete algorithm, we have taken snapshots of the outputs at different parts of the algorithm, which is visible in Figure 3.6. In Figure 3.6, the input images, ground truth saliency maps, handcrafted method target predictions and finally the model's refined pseudo labels are shown. The blue bounding box represents pseudo labels that the algorithm generated as noisy and consequently discarded in favour of the model's own predictions. The red bounding box depicts pseudo labels that are classified as clean and are minimally refined.

Figure 3.6: Input, Ground Truth, handcrafted predictions, final refined label. The first row shows a random sample of input images. In the second row, we see the ground truth masks (which are only used during evaluation). The handcrafted predictions are shown in the next four rows, with noisy samples in the blue bounding box and clean samples in the red bounding box. Comparing the final row to the ground truth row gives an indication of model performance.

Algorithm 1 and Algorithm 2 show the implemented pseudo code. Algorithm 2 depicts the outer loop of the training iteration. Here, we iterate through the specified number of epochs, and if it is less than W (where $W = 10$) the model is considered to be warmed-up. The warmup period is required as the model's own predictions are used during label refinement. If there was no warmup period, then the models' initial predictions would be close to random, which would result in poor label refinement. After the warmup period, a GMM is used to create a model based on the loss. The GMM

is then used to make a prediction based on it's modelled probability on whether each pseudo label is noisy or clean. A data loader for the noisy and clean pseudo labels is then generated and is used during the training procedure.

In Algorithm 1, pseudo label refinement occurs. The level of refinement depends on whether the model classifies the current label as clean or noisy. A GMM uses the loss per label to predict clean and noisy class probabilities at each epoch.

Algorithm 1 DeepDivide - Train

1: **procedure** TRAIN($\theta^{(1)}, \theta^{(2)}, L, U$)
2: **for** $l, u, w_l \in \mathcal{L}, U$ **do** ▷ Draw batch from datasets, w_l are weights from GMM
3: $output \leftarrow 0$
4: **for** `m=0; m++; m < M` **do** ▷ Noisy label part
5: $A[m] \leftarrow Augment[u]$ ▷ Augmentations = Jitter, Blur and Noise
6: $output \leftarrow \theta^{(1)}(A[m])$ `+ output` ▷ Accumulate model1 predictions
7: $output \leftarrow \theta^{(2)}(A[m])$ `+ output` ▷ Accumulate model2 predictions
8: **end for**
9: $output \leftarrow \frac{1}{2M} \times output$ ▷ Average output over Augmentations and models
10: $ptu \leftarrow output^{\frac{1}{2T}}$ ▷ Temperature sharpening, ptu is the refined label
11: $Lu \leftarrow loss(\theta^{(1)}(x), ptu)$ ▷ Noisy loss between input and refined label
12: **for** `m=0; m++; m < M` **do** ▷ Labelled part
13: $A[m] \leftarrow Augment[l]$ ▷ Augmentations = Jitter, Blur and Noise
14: $output \leftarrow \theta^{(1)}(A[m])$ `+ output` ▷ Accumulate model1 predictions
15: **end for**
16: $output \leftarrow \frac{1}{M} output$ ▷ Average output over Augmentations
17: $output \leftarrow w_l y + (1 - w_l) output$ ▷ Combine model prediction with clean labels based on probability from GMM
18: $ptl \leftarrow output^{\frac{1}{2T}}$ ▷ Temperature sharpening, ptl is the refined label
19: $Ll \leftarrow loss(\theta^{(1)}(x), ptl)$ ▷ Clean loss between input and refined label
20: $L \leftarrow \frac{len(labelled-dataset)}{len(entire-dataset)} Ll + \frac{len(unlabelled-dataset)}{len(entire-dataset)} Lu$
21: $\theta^{(1)} \leftarrow ADAM(L, \theta^{(1)})$
22: **end for**
23: **end procedure**

The probability returned by the GMM, $\mathrm{GMM_p}$, is used to partition the dataset into clean and noisy pseudo labels, where noisy samples have a $\mathrm{GMM_p} < 0.5$ and clean samples have a $\mathrm{GMM_p} \geq 0.5$. The threshold was set to 0.5 as per [Li *et al.* 2020]. If the pseudo label is classified as clean, then the refined label is generated by combining the model's prediction with the hand-crafted label. The probability provided by the GMM is used to weight the label and the model prediction in the ratio of $\mathrm{GMM_p}$ and $1 - \mathrm{GMM_p}$ respectively, as seen in Line 17 of Algorithm 1. Noisy labels are completely discarded in favour of the model's own prediction. Two models are trained simultaneously in order to mitigate confirmation bias, such that model 1 splits the dataset for model 2 into clean and noisy pseudo labels, and model 2 splits the dataset for model 1 into clean and noisy pseudo labels. A warm-up period of 10 epochs is required in order to pre-train the models so that their outputs are not random when using the model's

predictions during label refinement. The loss of

$$1 - F_\beta$$

where

$$F_\beta = \frac{(1+\beta^2)\textit{Precision} \times \textit{Recall}}{(\beta^2)\textit{Precision} + \textit{Recall}}$$

for the noisy and clean pseudo labels is weighted based on the length of the respective dataset and optimised using ADAM. The training runs for a total of 200 epochs.

Algorithm 2 DeepDivide - Main

Require: $\theta^{(1)}, \theta^{(2)}$, pseudo label dataset (X, Y), number augmentations M, Temperature sharpening T, Warmup epochs W, Clean label threshold p, Augmentations M

procedure MAIN($args$)
  while $epoch \leq EPOCHS$ **do**
    if $epoch \leq W$ **then**
      $\theta^{(1)}, \theta^{(2)} \leftarrow warmup(X, Y, \theta^{(1))}, \theta^{(2)})$ ▷ Warmup both models for W epochs
    else
      $prob1, prob2 \leftarrow GMM(X, Y, \theta^{(1)}, \theta^{(2)})$ ▷ Model the loss with a GMM
      $L \leftarrow (x_i, y_i, p_i, \theta^{(2)})|p_i \geq p$ ▷ Clean dataset is a subset of (X,Y)
      $U \leftarrow (x_i, y_i, p_i, \theta^{(2)})|p_i < p$ ▷ Noisy dataset is the remainder of (X,Y)
      $\theta^{(1)} \leftarrow TRAIN(\theta^{(1)}, \theta^{(2)}, L, U)$
      $L \leftarrow (x_i, y_i, p_i, \theta^{(1)})|p_i \geq p$ ▷ Clean dataset is a subset of (X,Y)
      $U \leftarrow (x_i, y_i, p_i, \theta^{(1)})|p_i < p$ ▷ Noisy dataset is the remainder of (X,Y)
      $\theta^{(2)} \leftarrow TRAIN(\theta^{(2)}, \theta^{(1)}, L, U)$
    end if
  end while
end procedure

3.5 Research Hypothesis

Our hypothesis is stated as follows

H1. Quantify the effect that refining pseudo labels has on network performance by using the refined labels to train an existing deep salient object detection network Nguyen *et al.* [2019a].

3.6 Conclusion

This chapter has formally defined the research aim and given an overview of the datasets and how they are used. We have also described the implementation details that were used during model training, as well as the performance metrics. The high level algorithm was also discussed, and thereafter the research hypothesis was formally defined. In Chapter 4, a discussion of the experimental setup and results is presented.

Chapter 4

Experiments and results

Following from the previous chapter, two benchmark algorithms were implemented, namely Deep Robust Unsupervised Saliency Prediction [Nguyen *et al.* 2019a] and DivideMix [Li *et al.* 2020]. No modification was made to DeepUSPS [Nguyen *et al.* 2019a]. The DivideMix algorithm was adapted from an object classification network to a saliency detection network.

4.1 Experimental setup

The CNN network DRN-105-D [Yu *et al.* 2017] was used for both the pseudo label generator (HC-DNNs) and the salient object prediction network. Inputs are resized to 432×432 pixels. The pseudo label generation network is trained on the MSRA-B [Wang *et al.* 2017] dataset for 25 epochs for each hand-crafted method with $\beta^2 = 0.3$ with $1 \times e^{-6}$ as the initial learning rate. The conditional random field (CRF) is computed for every prediction and the historical moving average is computed as

$$MVA(x, p, k) = (1 - \alpha) \times CRF(y^j(x, p)) + \alpha \times MVA(x, p, k - 1)$$

where $\alpha = 0.7$, $y(x, p)$ is the pixel output for p in image x and $MVA(x, p, k - 1)$ is the previous accumulated moving average. The learning rate is doubled at each iteration, for a maximum of 3 iterations during the self supervision phase. The refined pseudo labels are written to disk and used as input to the DivideMix [Li *et al.* 2020] implementation. During the final training, $1 \times e^{-4}$ is used as the learning rate and $\beta^2 = 0.3$. Training is performed for 200 epochs with a batch size of 8. Warming-up the network is performed for the first 10 epochs so that the GMM (2 components, with maximum iterations of 10, tolerance of $1 \times e^{-6}$ and a regularisation covariance of $5 \times e^{-4}$) can model the loss and provide probabilities on whether the pseudo label is clean or noisy. A probability threshold of $P_t = 0.5$ is used to decide whether the pseudo label falls into the clean or noisy partition after each epoch. A temperature sharpening of $T = 0.5$ is used to sharpen the edges of the labels.

4.2 Results and Ablation Study

An average of three runs was taken in order to generate all results. The variance and standard deviation across the three runs are shown in Table 4.1. As can be seen from the standard deviation, all three runs produced similar results and there were no outliers in both the F_β-scores and the MAE. Table 4.2 shows that the proposed method shows a negligible improvement across some of the datasets and metrics and later we perform the t-test to verify this. The results published in DeepUSPS [Nguyen *et al.* 2019a] were attempted to be reproduced based on the provided code by the authors [Nguyen *et al.* 2019a], however, with much effort, the provided code did not yield the same results as published in the original paper Nguyen *et al.* [2019a]. The authors of DeepUSPS [Nguyen *et al.* 2019a] also provided a pre-trained model, and this yielded the published results (see Table 4.2). When training the model, using the same parameters and training data that were presented in the paper Nguyen *et al.* [2019a], the reproduced results were observed. We expected our reproduced results to be similar to the published results, but they yielded poorer performance. The following three reasons could be attributed to the variation in results. Firstly, we used a batch size of 8 while the authors used a batch size of 20. Secondly, our GPU hardware and software libraries were different. Lastly, the published code of the authors may not be the same code that was used to generate the released trained model. Hence, both the reproduced results and the published results are given, while the reproduced results are used as benchmark for fair comparison.

	MSRA-B		ECCSD		DUT	
	F_β	MAE	F_β	MAE	F_β	MAE
Variance	0.043	0.0014	0.09	0.0064	0.048	0.031
Standard Deviation	0.21	0.038	0.30	0.08	0.22	0.18
Mean	90.1	04.03	87.70	06.18	72.10	06.73

Table 4.1: The variance, standard deviation and mean of the three runs, per dataset. The mean results are used as the model's performance.

To test the significance of the results, the *t-test* was done. Group 1 was the reproduced results for DeepUSPS while Group 2 was our DeepDivideMix results. The calculated *P-value* was 0.9948 with a difference in means between group 1 and group 2 equalling 0.1667.

The above results show that there is a negligible performance improvement when comparing our method to the reproduced results of the baseline DeepUSPS [Nguyen *et al.* 2019a] method. This implies that further refinement of the pseudo labels is not possible using the current implementation. The results of DeepUSPS [Nguyen *et al.* 2019a] suggest that the generated pseudo labels, when used for training, result in similar performance to fully supervised methods. This means that the pseudo labels generated by their method are very close to human-annotated labels for the MSRA-B [Wang *et al.* 2017] dataset. To determine the performance impact of the various stages in the method, an ablation study was performed and the following components were analysed.

Model	MSRA-B		ECCSD		DUT	
	$F_\beta \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	MAE $\downarrow$
SBF	-	-	78.70	08.50	58.30	13.50
USD	87.70	05.60	87.30	07.04	71.56	08.60
DeepUSPS (published)	90.31	03.96	87.42	06.32	73.58	06.25
DeepUSPS (reproduced)	89.90	04.12	**87.80**	06.31	**72.30**	06.87
DeepDivideMix (OURS)	**90.10**	**04.03**	87.70	**06.18**	72.10	**06.73**

Table 4.2: The results of the implemented method as well as other unsupervised methods. $\uparrow$ indicates that bigger values are preferred, while $\downarrow$ indicates that lower values are preferred. **Bold** font refers to the best metrics per column, DeepUSPS (published) is excluded from the comparison.

- **Augmentations**: Adds colour jitter, Gaussian blur and Gaussian noise to the input training data. All three augmentations were performed on the input image, while no change was made to the pseudo labels. These augmentations were used to provide consistency regularisation. During training, both clean and noisy pseudo labels were used and their corresponding input images were augmented.
- **Refinement**: During label refinement, the clean labels were refined by combining the model's prediction and the clean pseudo labels with a weighting factor provided by the GMM. In the case of noisy pseudo labels, the noisy pseudo labels were discarded and replaced completely by the model's own prediction. This refinement process in the ablation study refers to refining both the clean and noisy pseudo labels while also including the augmentation component.
- **Clean pseudo labels**: In this study, the noisy pseudo labels were completely discarded and only the clean pseudo labels were used during training. Both the augmentation component and the refinement component were included during this test. A breakdown of the results when each component was turned on is seen in Table 4.3.

Model	MSRA-B		ECCSD		DUT	
	$F_\beta \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	MAE $\downarrow$	$F_\beta \uparrow$	MAE $\downarrow$
Augmentations	89.10	04.51	86.00	07.05	69.60	07.84
Refinement	89.90	04.49	85.80	07.33	70.90	07.22
Clean psuedo labels	90.1	04.03	87.70	06.18	72.10	06.73

Table 4.3: Ablation study results. $\uparrow$ indicates that bigger values are preferred, while $\downarrow$ indicates that lower values are preferred.

The results in Table 4.3 show that augmenting the input data resulted in a performance degradation in all metrics and across all datasets, when compared to the baseline. Refining the pseudo labels improved performance on the MSRA-B [Wang *et al.* 2017] and DUT [Lijun *et al.* 2017] datasets, but yielded degraded performance on the ECCSD [Shi *et al.* 2015] dataset. The augmentation and refinement components did not contribute positively to network performance when compared to the baseline results. Performance was only improved from the baseline when the noisy labels were discarded completely

from the dataset during training. We can infer from this that the noisy labels could not be refined successfully by the proposed method and that the noisy pseudo labels suffered degradation after refinement.

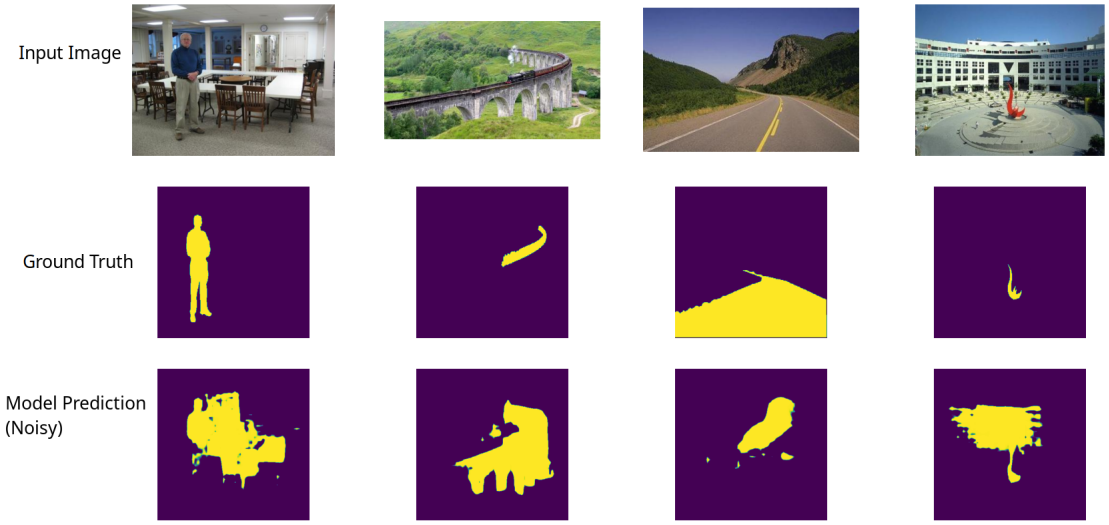

Figure 4.1: The first row shows sample input images for which noisy labels were predicted. In the second row, we see the ground truth masks. And the final row shows noisy predictions, i.e., pseudo labels that are classified as noisy.

As seen in Figure 4.1, the pseudo labels that the model are predicted as noisy result in performance degradation when included in the training dataset. Thus an alternate use of this work can be to partition pseudo labels into noisy and clean.

4.3 Summary

In this section, we presented and discussed our results. We showed that refining the noisy pseudo labels led to a statistically negligible performance improvement, that is there was no contribution towards increasing the average F_β-score and reducing the average MAE of the dataset. It was noted that the algorithm may be better used as an identifier of noisy labels which can be used for other methods.

Chapter 5

Conclusion

5.1 Summary

A label refinement algorithm for object classification was modified into a pseudo label refinement algorithm. This research focuses on the changes made to the pseudo label refinement algorithm and its effect on performance for unsupervised saliency object detection tasks. Changes were only made to the DivideMix [Li *et al.* 2020] algorithm in order to convert it from a classification-based label refinement framework into a salient object label refinement framework. Changes were made to the augmentation, data loader, and loss function components as these were configured for object classifications tasks. The label refinement module was added to the Nguyen *et al.* [2019a] method, and the results were analysed against the reproduced results of the benchmark DeepUSPS [Nguyen *et al.* 2019a] method. An ablation study was performed, and it was noted that in some cases both the augmentation and refinement components contributed negatively to network performance, while discarding the noisy pseudo labels resulted in a positive effect on performance. Thus, the hypothesis of the study (quantifying the effect of pseudo labels) resulted in the conclusion that the noisy pseudo labels could not be refined further.

The changes that were required to be made to the DivideMix [Li *et al.* 2020] method resulted in the novel network being more computationally and memory intensive in the following ways,

- Due to the additional model that is required to be trained, in order to reduce confirmation bias, the training time and GPU memory usage has more than doubled from the work in [Nguyen *et al.* 2019a].
- A warm-up period for both models is required, which further increases training time.
- In order to partition the dataset into clean and noisy samples, the entire dataset has to be evaluated at every epoch, which increases training time.
- There is no time or memory impact after the model is trained and used for salient object predictions.

5.2 Future Work

Learning a salient object detection network in an unsupervised way is a challenging task. This work focused on pseudo label refinement in order to reduce pseudo label noise as opposed to modelling the pseudo label noise. We found that the pseudo label noise was not adequately reduced to make a significant improvement to the final saliency predictions. In fact, discarding the noisy pseudo labels instead of refining them yielded better results. As such, a few areas of immediate further investigation could be explored:

- Investigate the performance impact of the classification threshold for the clean/noisy pseudo labels.
- Discard noisy labels during training of the DRNs and analyse the effect on the final predictions.

References

[Achanta *et al.* 2008] Radhakrishna Achanta, Francisco Estrada, Patricia Wils, and Sabine Süsstrunk. Salient region detection and segmentation. In *Computer Vualisation systems*, volume 5008, 05 2008.

[Alpert *et al.* 2007] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[Arpit *et al.* 2017] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. *A Closer Look at Memorization in Deep Networks*, 2017.

[Berthelot *et al.* 2019] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. *MixMatch: A Holistic Approach to Semi-Supervised Learning*, 2019.

[Borji *et al.* 2019] Ali Borji, Ming-Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *Comput. Vis. Media*, 5(2):117–150, 2019.

[Chen *et al.* 2019] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. *Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels*, 2019.

[Chen *et al.* 2020] Zuyao Chen, Qianqian Xu, Runmin Cong, and Qingming Huang. *Global Context-Aware Progressive Aggregation Network for Salient Object Detection*, 2020.

[Dingwen *et al.* 2017] Zhang Dingwen, Han Junwei, and Zhang Yu. Supervision by fusion: towards unsupervised learning of deep salient object detector. In *ICCV*, 2017.

[Everingham *et al.*] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[Everingham *et al.* 2010] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[Han *et al.* 2016] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. *Alternating Back-Propagation for Generator Network*, 2016.

[Han *et al.* 2018] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. *Co-teaching: Robust Training of Deep Neural Networks with Extremely Noisy Labels*, 2018.

[He *et al.* 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*, 2015.

[He *et al.* 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Identity Mappings in Deep Residual Networks*, 2016.

[Hou *et al.* 2016] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip H. S. Torr. Deeply supervised salient object detection with short connections. *CoRR*, abs/1611.04849, 2016.

[Ioffe and Szegedy 2015] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015.

[Itti *et al.* 1998] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[Jiang *et al.* 2013] Bowen Jiang, Lihe Zhang, Huchuan Lu, Chuan Yang, and Ming-Hsuan Yang. Saliency detection via absorbing markov chain. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, page 1665–1672, USA, 2013. IEEE Computer Society.

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

[Lafferty *et al.* 2001] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Lee 2013] Dong-Hyun Lee. *Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks*, 2013.

[Li *et al.* 2013] Xiaohui Li, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via dense and sparse reconstruction. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 12 2013.

[Li *et al.* 2018] Guanbin Li, Yuan Xie, and Liang Lin. *Weakly Supervised Salient Object Detection Using Image Labels*, 2018.

[Li *et al.* 2020] Junnan Li, Richard Socher, and Steven C.H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2020.

[Lijun *et al.* 2017] Wang Lijun, Lu Huchuan, Wang Yifan, Feng Mengyang, Wang Dong, Yin Baocai, and Ruan Xiang. Learning to detect salient objects with image-level supervision. In *CVPR*, 2017.

[Lin *et al.* 2022] Xiangru Lin, Ziyi Wu, Guanqi Chen, Guanbin Li, and Yizhou Yu. A causal debiasing framework for unsupervised salient object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36:1610–1619, 06 2022.

[Liu and Han 2016] Nian Liu and Junwei Han. Dhsnet: Deep hierarchical saliency network for salient object detection. In *CVPR2016*, pages 678–686, 06 2016.

[Liu *et al.* 2007] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):353–367, 2007.

[Liu *et al.* 2018] Nian Liu, Junwei Han, and Ming-Hsuan Yang. *PiCANet: Learning Pixel-wise Contextual Attention for Saliency Detection*, 2018.

[Lou *et al.* 2014] Jing Lou, Mingwu Ren, and Huan Wang. Regional principal color based saliency detection. *PloS one*, 9:e112475, 11 2014.

[Lowe 2004] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[Mohammadi *et al.* 2020] Sina Mohammadi, Mehrdad Noori, Ali Bahri, Sina Ghofrani Majelan, and Mohammad Havaei. Cagnet: Content-aware guidance for salient object detection. *Pattern Recognition*, 103:107303, Jul 2020.

[Nguyen *et al.* 2019a] Duc Tam Nguyen, Maximilian Dax, Chaithanya Kumar Mummadi, Thi-Phuong-Nhung Ngo, Thi Hoai Phuong Nguyen, Zhongyu Lou, and Thomas Brox. Deepusps: Deep robust unsupervised saliency prediction with self-supervision. *CoRR*, abs/1909.13055, 2019.

[Nguyen *et al.* 2019b] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. *SELF: Learning to Filter Noisy Labels with Self-Ensembling*, 2019.

[Pang *et al.* 2020] Youwei Pang, Xiaoqi Zhao, Lihe Zhang, and Huchuan Lu. *Multi-scale Interactive Network for Salient Object Detection*, 2020.

[Peng *et al.* 2017] Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. *Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network*, 2017.

[Permuter *et al.* 2006] H. Permuter, J. Francos, and I.H. Jermyn. A study of gaussian mixture models of color and texture features for image classification and segmentation. *Pattern recognition.*, 39(4):695–706, April 2006.

[Qin *et al.* 2019] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7471–7481, 2019.

[S. Mohamed 2017] Ihab S. Mohamed. *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques*. PhD thesis, Indiana University Bloomington, 09 2017.

[Shi *et al.* 2015] Jianping Shi, Qiong Yan, Li Xu, and Jiaya Jia. *Hierarchical Saliency Detection on Extended CSSD*, 2015.

[Shin *et al.* 2022] Gyungin Shin, Samuel Albanie, and Weidi Xie. Unsupervised salient object detection with spectral cluster voting. In *CVPRW*, 2022.

[Simonyan and Zisserman 2015] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2015.

[Srivastava *et al.* 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[Tarvainen and Valpola 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1195–1204. Curran Associates, Inc., 2017.

[Walber 2023] By Walber. *Precision and Recall*, 08 2023.

[Wang *et al.* 2017] Jingdong Wang, Huaizu Jiang, Zejian Yuan, Ming-Ming Cheng, Xiaowei Hu, and Nanning Zheng. Salient object detection: A discriminative regional feature integration approach. *International Journal of Computer Vision*, 123(2):251–268, 2017.

[Wang *et al.* 2018] Tiantian Wang, Lihe Zhang, Shuo Wang, Huchuan Lu, Gang Yang, Xiang Ruan, and Ali Borji. Detect globally, refine locally: A novel approach to saliency detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3127–3135, 2018.

[Wang *et al.* 2019] Wenguan Wang, Shuyang Zhao, Jianbing Shen, Steven C. H. Hoi, and Ali Borji. Salient object detection with pyramid attention and salient edges. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[Wei *et al.* 2019] Jun Wei, Shuhui Wang, and Qingming Huang. *F3Net: Fusion, Feedback and Focus for Salient Object Detection*, 2019.

[Whitehill *et al.* 2009] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, volume 22, pages 2035–2043, 01 2009.

[Xie and Tu 2015] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *CoRR*, abs/1504.06375, 2015.

[Xin *et al.* 2018] Li Xin, Yang Fan, Cheng Hong, Liu Wei, and Shen Dinggang. Contour knowledge transfer for salient object detection. In *ECCV*, 2018.

[Yan *et al.* 2022] Pengxiang Yan, Ziyi Wu, Mengmeng Liu, Kun Zeng, Liang Lin, and Guanbin Li. Unsupervised domain adaptive salient object detection through uncertainty-aware pseudo-label learning. *arXiv preprint arXiv:2202.13170*, 2022.

[Yang *et al.* 2013] Chuan Yang, Lihe Zhang, and Lu. Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3166–3173. IEEE, 2013.

[Yu *et al.* 2017] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[Zhang *et al.* 2017] Pingping Zhang, Dong Wang, Huchuan Lu, Hongyu Wang, and Xiang Ruan. *Amulet: Aggregating Multi-level Convolutional Features for Salient Object Detection*, 2017.

[Zhang *et al.* 2018a] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. *mixup: Beyond Empirical Risk Minimization*, 2018.

[Zhang *et al.* 2018b] J. Zhang, T. Zhang, Y. Daf, M. Harandi, and R. Hartley. Deep unsupervised saliency detection: A multiple noisy labeling perspective. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9029–9038, 2018.

[Zhang *et al.* 2020] Jing Zhang, Jianwen Xie, and Nick Barnes. *Learning Noise-Aware Encoder-Decoder from Noisy Labels by Alternating Back-Propagation for Saliency Detection*, 2020.

[Zhao and Wu 2019] Ting Zhao and Xiangqian Wu. *Pyramid Feature Attention Network for Saliency detection*, 2019.

[Zhou *et al.* 2022] Huajun Zhou, Bo Qiao, Lingxiao Yang, Jianhuang Lai, and Xiaohua Xie. Appearance-guided attentive self-paced learning for unsupervised salient object detection. In *arXiv*, 2022.

[Zhu *et al.* 2014] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 2814–2821, USA, 2014. IEEE Computer Society.

[Zou and Komodakis 2015] Wenbin Zou and Nikos Komodakis. Harf: Hierarchy-associated rich features for salient object detection. In *HARF: Hierarchy-Associated Rich Features for Salient Object Detection*, pages 406–414, 12 2015.