

https://dspace.org/

The impact of encoded features from action classifiers on dense video captioning performance

Bhugwan, Dhruv

2021
https://hdl.handle.net/10539/33029

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

Dissertation

The Impact of Encoded Features from Action Classifiers on Dense Video Captioning Performance

Author:
Dhruv Bhugwan
1037363

Supervised by:
Prof. Terence van Zyl
Dr. Richard Klein

September 30, 2021



University of the Witwatersrand, Johannesburg

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 in Computer Science.

Declaration

I, Dhruv Bhugwan (1037363), declare that this dissertation titled, "The Impact of Encoded Features from Action Classifiers on Dense Video Captioning Performance" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this dissertation has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this dissertation is entirely my own work.
- I have acknowledged all main sources of help.
- Where the dissertation is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 30 / 09 / 2021

Contents

Preface
Declaration i
Table of Contents ii
List of Figures v
List of Tables vi
Abstract 1
Acknowledgements 2

1 Introduction **3**
1.1 Dense Video Captioning 3
1.2 Problem Statement 5
1.3 Significance and Motivation 6
1.4 Research Aims and Objectives 6
1.5 Research Hypothesis 7
1.6 Research Questions 8
1.7 Nomenclature 8
1.8 Outline 9
1.9 Conclusion 9

2 Background **10**
2.1 Introduction 10
2.2 Convolutional Neural Networks 10
2.2.1 Mobilenet 11
2.3 Recurrent Neural Networks 13
2.3.1 Long-Short Term Memory Networks 14
2.4 Video Captioning 15
2.5 Datasets 15
2.5.1 Sports-1M 16
2.5.2 Kinetics 16
2.5.3 ActivityNet Captions 16
2.6 Principal Component Analysis 16
2.6.1 Incremental Principal Component Analysis 17
2.7 K-Means Clustering 18
2.8 Conclusion 18

3 Related Work **19**
3.1 Introduction . . . 19
3.2 Action Recognition and 3D Convolutional Neural Networks . . . 20
3.2.1 C3D . . . 20
3.2.2 3D MobileNet . . . 20
3.3 Temporal Action Proposals . . . 21
3.4 Dense Video Captioning . . . 22
3.5 Self-Attention . . . 23
3.5.1 Attention Estimator Module . . . 23
3.5.2 Squeeze and Excitation . . . 25
3.6 MobileNetV3 . . . 25
3.7 Conclusion . . . 27

4 Research Methodology **28**
4.1 Introduction . . . 28
4.2 Data Preparation . . . 28
4.2.1 Kinetics-368 . . . 28
4.2.2 ActivityNet Captions . . . 29
4.3 Action Classifiers . . . 34
4.4 Temporal Action Proposal Module . . . 35
4.5 Caption Generation . . . 36
4.5.1 Temporal Dynamic Attention . . . 36
4.5.2 Context Gating . . . 37
4.6 Evaluation Metrics . . . 37
4.6.1 BLEU . . . 37
4.6.2 METEOR . . . 38
4.6.3 ROUGE-L . . . 39
4.6.4 CIDEr . . . 40
4.6.5 Temporal Action Proposals . . . 40
4.6.6 PCA: Reconstruction Error . . . 40
4.6.7 Action Classifiers . . . 41
4.7 Training Method . . . 41
4.7.1 Action Classifiers . . . 41
4.7.2 Dimensionality Reduction . . . 41
4.7.3 Temporal Action Proposals . . . 42
4.7.4 Captions . . . 42
4.7.5 Total Loss . . . 43
4.8 Inference . . . 43
4.9 Conclusion . . . 44

5 Results and Analysis **45**
5.1 Introduction . . . 45
5.2 Action Classifiers . . . 45
5.2.1 Temporal Depth Effect on Results . . . 46
5.2.2 Attention Estimator Effect on the Networks and their Results . . . 47
5.3 Dense Video Captioning . . . 48

5.3.1 Impact of Dataset Choice . . . 48
5.3.2 Temporal Action Proposals . . . 49
5.3.3 Captions . . . 51
5.4 Summary and Answers to Research Questions . . . 54
5.5 Qualitative Analysis . . . 55
5.6 Conclusion . . . 59

6 Conclusion **61**
6.1 Discussion . . . 61
6.2 Future Work . . . 62

A Appendix **63**
A.1 Classifier Architectures . . . 63

References **69**

List of Figures

2.1 Linear Bottleneck Block. *Taken From* Sandler *et al.* (2018) 13
2.2 Unrolled RNN 14
2.3 Image Caption Generator 16

3.1 Generic Dense Video Captioning Pipeline. *Adapted from* Krishna *et al.* (2017) 22
3.2 Implementation of the Attention Estimator Module proposed by Jetley *et al.* (2018). *Taken From* Jetley *et al.* (2018) 24
3.3 Implementation of the Squeeze and Excitation Module proposed by Hu *et al.* (2018). *Taken From* Hu *et al.* (2018) 25
3.4 A plot of the h-swish activation function. *Taken From* Howard *et al.* (2019) 26

4.1 BiDirectional SST Pipeline. *Taken from* Wang *et al.* (2018). (a) A video is encoded by an action classifier and the features are reduced to a fixed size using PCA. (b) The encoded features are passed through a bidirectional LSTM encoder. (c) The forward and backward LSTM hidden states are processed by a proposal prediction ANN and are jointly used to generate temporal proposal predictions. (d) The hidden states at the start and end of an event are concatenated with the encoded visual feature vectors. (e) The concatenated features are passed through a caption generation LSTM decoder to generate sentences. 35

5.1 Precision@1000 vs tIoU 49
5.2 Recall@1000 vs tIoU 50
5.3 A video of a man mowing the lawn. Each pair of images represent the start and end frame for each event in the video. The ground truth caption for each event is provided in the caption below each image pair. . . 56

List of Tables

4.1 ActivityNet Captions Data Structure 29

5.1 Architecture - Action Classifier, ACC@1 - Video Level Top-1 accuracy, ACC@5 - Video Level Top-5 accuracy, ACC@10 - Video Level Top-10 accuracy, ACC@20 - Video Level Top-20 accuracy, Temporal Depth - Number of Frames processed in a single pass and NUM_PARAMS - The Total Number of Trainable Parameters 45
5.2 Architecture - Action Classifier, iPCA Reconstruction Error - The Mean squared error of the ActivityNet Captions features after transforming the data back to the original subspace. 50
5.3 Caption Results on the ActivityNet Captions Dataset. All Scores are Presented as Percentage Values. 51
5.4 C3D-8 Proposal sentence results 57
5.5 C3D-16 Proposal sentence results 57
5.6 C3D-8 Attn Proposal sentence results 57
5.7 C3D-16 Attn Proposal sentence results 58
5.8 3D MobileNetV3 Proposal sentence results 58
5.9 3D MobileNetV3 Attn Proposal sentence results 59

A.1 C3D_8 Architecture 63
A.2 C3D_16 Architecture 64
A.3 C3D_8 Attn Architecture 64
A.4 C3D_16 Attn Architecture 65
A.5 3D MobileNetV3 Architetcure 66
A.6 3D MobileNetV3 Attn Architetcure 67

Abstract

Action classification is the process of identifying what action is being performed in a video. Dense video captioning is the process of identifying time segments within a video where individual events occur and describing each event with a caption. 3D Convolutional Neural Networks (3D CNNs) pretrained on an action classification dataset are used to encode a video from a video captioning dataset into a set of features. These features are then passed to a temporal action proposal and captioning modules to identify the time segments and their respective captions. For our work, we consider only the 3D CNNs used to generate features on the ActivityNet Captions dataset to investigate their impact on the dense video captioning performance. Features generated on the C3D architecture pretrained on the Sports1M dataset are commonly used due to their availability. To improve feature performance we utilize the Kinetics-368 dataset which does not only contain sporting action classes such as the Sports1M dataset but contains everyday action classes as well. We propose a 3D version of the MobileNetV3 architecture to investigate the performance of resource efficient action classifiers as feature extractors. Furthermore, we propose a 3D attention estimator module that can be integrated into existing 3D CNN architectures. We integrate the 3D attention estimator module into the C3D and 3D MobileNetV3 architectures and train the C3D, 3D MobileNetV3 and their attention augmented counterparts on the Kinetics-368 dataset. Thereafter we generate a set of features on the ActivityNet Captions dataset using each of the aforementioned 3D CNNs. We empirically show that our classification accuracy on the Kinetics-368 dataset can improve when integrating the 3D attention estimator module into the C3D architecture where we found gains of 8.87%. The 3D MobileNetV3 architecture obtained the best classification accuracy on the Kinetics-368 dataset, but the structure of the network bottlenecked the amount of information provided to the attention estimator module resulting in a performance decrease of 2.63% indicating that the base architecture structure must be considered before integrating the 3D attention estimator module. We find that the features generated using 3D MobileNetV3 have the largest loss of information due to PCA being applied to them but they achieve higher scores in half of our temporal action proposal and caption metrics. This indicates that classification performance is indicative of temporal proposal and captioning performance. Furthermore, we show that the scale of the dataset used to pretrain the 3D CNNs used for feature extraction is a large contributing factor to dense video captioning performance. The Sports1M dataset has approximately 6 times more videos per action class compared to the Kinetics-368 dataset. Across all metrics, we find that our features obtained temporal action proposal and caption scores far lower than the features generated on the C3D architecture pretrained on the Sports1M dataset.

Acknowledgements

There are quite a few people that helped me cope with this project and resources that were made available to me throughout the course of this long long journey as such I would just like to give a few words of thanks for all of their contributions.

This dissertation was supported in part by two main organizations. Firstly we were supported by the National Research Foundation of South Africa (Grant number: 118075). Secondly, we were supported by the Centre for High Performance Computing (CHPC), South Africa, which provided us with computational resources to do this research project (Project number: CSCI1340).

I would like to thank both of my supervisors Richard Klein and Terence van Zyl without whom I would not have managed to complete or even had the confidence to continue with this project, especially when components or implementations of this project were broken. They did not give up and continually advised me throughout this project. They organized access to computational resources without which I would not have been able to train half of the networks. The meetings and advice they gave me helped me focus on individual components instead of worrying about the project as a whole.

I would like to thank the Wits University School of Computer Science and Applied Maths. The RAIL lab was instrumental in this project as I was provided not only with a computer but a lab with peers who were always willing to help and consult with me when I had setbacks.

I would also like to thank my friends and family who always motivated and believed in me to continue during the current COVID-19 pandemic even when lockdown became very stressful with cabin fever. Knowing that there were people behind me and motivating me especially near the end telling me that I could do anything I wanted as soon as I was done was a great factor in me managing to finish this dissertation. I would like to thank two people, in particular, first I want to thank my colleague Kyle who was also completing his dissertation at the same time as me. The hours we spent debugging each other's code trying to understand why our networks were not training was of great help as I was not the only one who struggled at points when completing a dissertation. Second is my sister, Uma without the energy drinks and emotional support that you continually provided me with this project would have most likely been impossible. You listening to me talk about my research even when you did not understand what I was doing was immensely valuable.

Lastly, there are some open-source projects that I would like to acknowledge:

- Python the language and community were invaluable.
- Pytorch an impressive deep learning library that has many functions and guides that made life easy.
- Scikit-Learn which made the evaluation and data manipulation very simple.

Chapter 1

Introduction

1.1 Dense Video Captioning

Dense video captioning is the process of identifying time segments within a video where events occur and generating a caption describing each event within each identified time segment [Krishna *et al.* 2017]. Dense video captioning can be a useful tool in teaching people new skills, in a cooking video, the steps to replicate the recipe can be detailed with the relevant time segments. In sports or exercise, the motions to replicate a technique can be identified and broken down and described to people who are unfamiliar with the required steps. In situations such as these dense video captioning can be a useful tool in teaching people new skills.

The field of dense video captioning is currently an area of focus due to the availability of the first large-scale temporally segmented and captioned video dataset, the ActivityNet Captions dataset created by Krishna *et al.* (2017). The ActivityNet challenge [Ghanem *et al.* (2018)] is an activity recognition challenge that spans a variety of action recognition topics including dense video captioning. Dense video captioning is the process of identifying time frames and providing captions for each event in a video. There exists a common "pipeline" from which these models are constructed, see Figure 3.1. In this work we break down this pipeline and it should be noted that Krishna *et al.* (2017) used the C3D architecture [Tran *et al.* 2015] pretrained on the Sports1M dataset to generate a set of encoded features on the ActivityNet Captions dataset. The features are commonly used in temporal action proposal and captioning networks for dense video captioning. Krishna *et al.* (2017) provide the C3D features for this purpose on the ActivityNet Captions dataset. We also consider alternative features based on the C3D [Tran *et al.* 2015] and new 3D MobileNetV3 architectures pretrained on the Kinetics-368 dataset. The aforementioned features are used in the dense video captioning pipeline proposed by Wang *et al.* (2018), and we do not modify the temporal action proposal or captioning module implemented by them.

Video encoding is the process of extracting information and producing a vector representing the important features from a sequence of video frames. 3D Convolutional Neural Networks (CNN) are used to encode video data. 3D CNNs encode videos by

partitioning them into T segments of δ frames per segment. 3D convolutional filters can be considered volumes of dimensions $(D \times W \times H)$ where D is the temporal depth or how many consecutive frames will be considered by that filter. By utilizing the temporal depth of the 3D convolutional filters, temporal dependencies may be identified. 2D CNNs can process videos, but they will do so on a frame by frame basis. This disregards temporal dependency between frames and therefore incorrect classifications may be obtained.

Many methods utilise the classical Convolutional 3D (C3D) architecture proposed by Tran *et al.* (2015) that has shown to perform well on action recognition datasets such as UCF101 and Sports1M provided by Soomro *et al.* (2012) and Karpathy *et al.* (2014) respectively. The datasets such as the Kinetics [Kay *et al.* 2017] and HMDB [Kuehne *et al.* 2011] has proven to be far more difficult to categorize for the C3D architecture and these datasets have introduced the need for deeper or more sophisticated architectures for action recognition. One such architecture proposed by Kopuklu *et al.* (2019) is the 3D MobileNetV2 architecture.

Kopuklu *et al.* (2019) proposed adaptations of resource efficient (having minimal impact on computational resources and requiring a minimal number of mathematical operations in a forward pass) CNNs such as MobileNet [Howard *et al.* 2017], MobilenetV2 [Sandler *et al.* 2018] and ShuffleNetV2 [Ma *et al.* 2018] to perform classification on video datasets, by converting their 2-Dimensional (2D) convolutional filters to 3-Dimensional (3D). These resource efficient architectures have fewer parameters than classical architectures, but manage to outperform them by making use of more complex logic when processing information. MobileNetV2 implements blocks akin to those found in ResNet [He *et al.* 2016] whereby there exist residual connections between the input and output of specific blocks. The residual connections aid the network in predicting the input of the block value after performing operations on the input in a block [He *et al.* 2016]. These residual connections were introduced by He *et al.* (2016) as a means of overcoming the vanishing gradient problem in deeper architectures. Inspired by the 3D MobileNet V2 architecture [Kopuklu *et al.* 2019] we propose a 3D implementation of the more recent MobileNetV3 architecture introduced by Howard *et al.* (2019).

Soft-attention is a general term for a differentiable technique of weighting input values as a means of assigning importance to different parts of the input tensor. Soft-attention aids the network in filtering out noise by assigning a larger weight to common features which are learned during training. Self-attention is a term for trainable soft-attention modules included in deep learning architectures. Xie *et al.* (2018) and Hu *et al.* (2018) implement similar techniques for self-attention whereby, they calculate different weights for each of the input's channels. Each channel having a different weight associated with it aids the network in filtering out feature maps from convolution kernels that have little relevance to the current input. Jetley *et al.* (2018) implemented a different version of self-attention; (attention estimator module); whereby they extract feature maps from strided intervals in the network architecture and weight each of the pixels instead of channels. Thereafter, they perform a sum over the pixels in each individual channel, concatenate the outputs of each of these weighted feature maps

and use them to perform a prediction. Inspired by these various attention implementations we propose an augmentation of the C3D and the 3D MobileNetV3 by utilizing a 3-Dimensional adaptation of the attention mechanism used by Jetley *et al.* (2018).

Temporal action proposal frameworks process the encoded vectors obtained using video action recognition architectures. They output temporal proposals which are effectively time intervals in a video where separate actions occur. These are usually implemented using some sort of Recurrent Neural Network (RNN). We used the Bidirectional Single Steam Temporal (Bidirectional SST) Action action proposal framework introduced by Wang *et al.* (2018). Each feature vector $v_t \in [1, 2, ..., T]$ is passed through this framework. The framework outputs K binary proposals indicating if an action ended in that segment, where it began, a confidence score, and the hidden state of the RNN. SST and by extension the Bidirectional SST framework use C3D as their primary manner of video processing. We implemented attention augmented networks such as 3D MobileNetV3 and attention augmented C3D as an alternative video processing architecture due to the increase in classification performance when applying such techniques with a minimal impact on the overall size of the architecture.

1.2 Problem Statement

The field of dense video captioning focuses most of its work on context improvement between adjacent event captions. While this may be a very significant area for improvement in the field, many methods use classical visual models from the field of action recognition such as C3D for feature extraction. While the C3D architecture [Tran *et al.* 2015] performed well on the UCF101 [Soomro *et al.* 2012] and Sports1M datasets [Karpathy *et al.* 2014], there exist other methods which utilize different concepts such as self-attention. Self-attention has shown to improve results with a minimal impact on the size of the overall architecture as seen in Jetley *et al.* (2018) and Hu *et al.* (2018). We acknowledge that there are architectures that excel in action classification tasks such as I3D [Carreira and Zisserman 2017], ResNet-101, and ResNeXt-101 [Kopuklu *et al.* 2019], but these architectures have a large number of parameters and need hardware to support loading and training them which is not always readily available.

Good action classifiers require architectures that excel at feature extraction, and in turn enable better action proposals and caption generation. However, a perfect action recognition architecture does not yet exist, since current methods of dense video captioning focus on contextual improvement between captions. Krishna *et al.* (2017), Buch *et al.* (2017) and Wang *et al.* (2018) use the easily available C3D architecture pretrained on the Sports1M dataset for feature extraction, which is not necessarily the best that currently exists.

In response to this problem, we proposed to implement and evaluate the accuracy difference between dense video captioning networks that use the C3D architecture and a new 3D MobileNetV3 architecture as feature extractors. Furthermore, we will explore the efficiency of the implemented feature extractors when augmented with the attention module proposed by Jetley *et al.* (2018) adapted for 3D convolutions.

1.3 Significance and Motivation

The availability of many large scale video action recognition competitions and datasets has resulted in the construction of architectures that outperform classical methods such as 3D MobileNet V2 [Kopuklu *et al.* 2019] and deeper architectures such as I3D [Carreira and Zisserman 2017]. The construction of these architectures which perform well in video action recognition competitions allows us to explore alternatives to classical methods of temporal action proposals and caption generation. The improvement of context between subsequent video segments does not solely depend on the temporal proposal and caption generation module of a dense captioning network. Features are fed from the video encoder to a temporal action proposal module and finally to the captioning module. Thus features are propagated through the network and while new techniques are created to improve the context between captions, feeding them features obtained from better performing encoders may produce more accurate results.

1.4 Research Aims and Objectives

Dense video captioning is a fairly new topic in the computer vision and natural language processing domains. As such a variety of techniques to improve the performance of current methods are being explored from action recognition and action proposal networks to context integration between detected actions. This research aims to improve the Bidirectional SST framework by utilizing 3D CNNs pretrained on the Kinetics-368 dataset as feature extractors. We constructed the 3D MobileNetV3 architecture, utilizing the methods Kopuklu *et al.* (2019) used to convert MobileNetV2 to the 3D MobileNetV2 architecture. Futhermore, we constructed a 3D Attention Estimator Module and integrated it into the C3D and 3D MobileNetV3 architectures. C3D, 3D MobileNetV3, and subsequent self-attention augmented architectures thereof were used as feature extractors for the Bidirectional SST temporal action proposal framework. We analyzed the performance between our implemented methods using 2018 ActivityNet challenge metrics: CIDEr [Vedantam *et al.* 2015], METEOR [Banerjee and Lavie 2005], BLEU [Papineni *et al.* 2002] and ROUGE-L [Lin 2004].

We trained our networks on a subset of the Kinetics-600 dataset namely the Kinetics-368 dataset. While there does exist a Kinetics-400 dataset that contains 400 unique action classes with approximately 400 videos per action class, we constructed the Kinetics-368 dataset from the intersection of the Kinetics-400 and Kinetics-600 classes. The Kinetics-400 dataset is not a subset of the Kinetics-600 dataset. The kinetics-600 dataset has 600 unique action classes with approximately 600 videos per action class. We constructed the Kinetics-368 dataset using the 368 classes that are common between the Kinetics-400 and Kinetics-600 datasets whereby all video IDs were taken from the Kinetics-600 dataset. Thus the Kinetics-368 dataset has 368 unique classes with each class containing a minimum of 600 videos. While we had access to Nvidia GTX 1060 GPU's, the training time for a model to converge on the Kinetics-600 dataset was substantially longer than that of the Kinetics-400 dataset. Therefore, we constructed this dataset due to restrictions in available compute, while still trying to maintain a feature-rich dataset with a large variety of videos per action class.

The Sports1M dataset has 487 classes with approximately 2500 videos per action class but the action classes are specifically targeted towards sports and sporting actions [Karpathy *et al.* 2014]. All Kinetics datasets consist of not only sporting actions but also generalized actions such as braiding hair or headbanging among other classes [Carreira *et al.* 2018].

Using C3D and 3D MobilenetV3 models pretrained on the Kinetics-368 dataset we encoded the videos in the ActivityNet Captions dataset to a set of embedded feature vectors where the number of vectors per video will differ depending on its length. Using these feature vectors we trained the temporal proposal and caption modules. We compared temporal proposal and caption results of the features obtained from networks pretrained on the Kinetics-368 dataset to the available features generated from a C3D network pretrained on the Sports1M dataset. This comparison allowed us to evaluate if features obtained by networks pretrained on a more general set of action classes can result in improved results.

The above aims of this research project were achieved through the following objectives:

- Extract the 368 action classes common between the Kinetics-400 and Kinetics-600 datasets to obtain the Kinetics-368 dataset;
- construct the 3D MobileNetV3 architecture following the adaptations Kopuklu *et al.* (2019) used on the MobileNetV2 architecture to construct 3D MobileNetV2;
- construct an augmented 3D MobileNetV3 model using an adaptation of the self-attention technique mentioned in Jetley *et al.* (2018) so it can accept 3D tensors;
- train the 3D MobileNetV3 and the 3D MobileNetV3 augmented with self-attention models on the Kinetics-368 dataset;
- replicate C3D model and train it on the Kinetics-368 dataset;
- augment C3D model with an adaptation of the self-attention technique mentioned in Jetley *et al.* (2018) so it can process 3D tensors, and train it on the Kinetics-368 dataset;
- train the attention augmented C3D model on the Kinetics-368 dataset;
- replicate and modify the Bidirectional SST framework from Wang *et al.* (2018) to utilize the 3D MobileNetV3 models for feature extraction;
- train the models on the ActivityNet Captions dataset;
- analyze and discuss the results of the evaluation metrics.

1.5 Research Hypothesis

As discussed in Chapter 3, C3D is a commonly occurring visual encoder or feature extractor in the field of dense video captioning even in recent works. But there exist more recent works which perform comparably with significantly fewer parameters on

existing action classification datasets such as 3D MobileNetV2 on the UCF-101 dataset [Kopuklu *et al.* 2019; Tran *et al.* 2015]. Furthermore, attention mechanisms have recently been used as a means of improving the performance of deep neural networks while increasing the number of computations and number of neurons by a minimal amount. The field of dense captioning has been progressing largely in the region of integrating and improving contextual information between the captions. We explore feeding temporal action proposal frameworks and subsequently caption generators features from networks that perform better on the Kinetics-368 dataset. Furthermore, we explore if a network that has been trained on a more general set of action classes such as daily activities instead of only sporting classes will produce a better set of feature vectors.

We hypothesize that providing temporal action proposal and captioning networks with features from networks that performed better on the Kinetics-368 dataset will result in higher METEOR, CIDEr, ROUGE-L, and BLEU scores and better temporal action proposals.

1.6 Research Questions

1. By how much does the top-1 and top-5 accuracy differ between 3D MobileNetV3 and C3D on the Kinetics-368 dataset?
2. By how much does the top-1 and top-5 accuracy differ between C3D and 3D MobilenetV3 augmented with a 3D adaptation of the attention module from Jetley *et al.* (2018) and their base configurations?
3. By how much do the CIDEr, METEOR, BLEU, and ROUGE-L scores differ when features extracted from C3D and 3D MobileNetV3 pretrained on the Kinetics-368 dataset are used to train the Bi-Directional SST temporal action proposal module and the subsequent caption module from Wang *et al.* (2018) compared to the available features obtained from the C3D network pretrained on the Sports1M dataset?
4. To what degree are the CIDEr, METEOR, BLEU, and ROUGE-L scores affected by using features from the C3D and 3D MobileNetV3 architectures when augmented with the attention module from Jetley *et al.* (2018)?
5. To what extent are the temporal action proposal windows improved on the ActivityNet Captions dataset by utilizing features obtained from 3D MobileNetV3, C3D and their attention augmented architectures when pretrained on the Kinetics-368 dataset compared to the available features obtained from the C3D network pretrained on the Sports1M dataset?

1.7 Nomenclature

Let $\mathbb{I}$ be the domain of image pixels $[0, 1, 2, ..., 255]$.

1.8 Outline

In the rest of this paper, we include a detailed background in Chapter 2 and related works chapters in Chapter 3. In Chapter 4 we outline our research methodology, specifically the tools and data that we use for our analysis. Thereafter in Chapter 5 we discuss our findings our interpretation of the results and possible future work in chapter 6.

1.9 Conclusion

In this chapter, we provided some background knowledge into the field of video captioning and topics needed to understand the standard pipeline followed to implement these models. We identified similarities between existing works namely C3D as a visual encoder and spoke about recent alternatives. Furthermore, we spoke about SST which is a temporal action proposal framework for identifying actions in a video. We spoke about our research problem and how we will try and improve existing works using attention augmented architectures and more recent architectures such as 3D MobilenetV3 compared to the classical C3D architecture. We then outlined our objectives and research questions which we will be using to evaluate our modifications to the Bi-Directional SST framework.

Chapter 2

Background

2.1 Introduction

Dense video captioning requires an encoding of a video to generate temporal segments and captions for each event. This can be described as an encoder-decoder network. This chapter describes the foundation of the networks that we will use for the action classification segment of the dense video captioning pipeline as seen in Figure 4.1. We give a brief description of CNN's and how traditional convolutional layers operate. We describe MobileNet and MobileNetV2, their depthwise separable convolutions, and linear bottleneck blocks respectively. We describe how depthwise separable convolutions are an approximation to traditional convolutions but use significantly fewer parameters when a large number of output channels are desired. We describe RNN's and LSTM's as they are the foundation of the decoder segment for all image and video captioning pipelines. We outline the generic process of image and video captioning where temporal segmentation is not performed and a single caption is generated for a single image or video. We describe common action classification and dense caption datasets that we will use or modify for this dissertation because of restrictions in the computational resources available to us. We outline the process of using PCA for dimensionality reduction on an input set of data and we outline the process of clustering data using the K-Means algorithm. The aforementioned algorithms are described as they are used to structure the data which will be used as input to the temporal segmentation and captioning modules.

2.2 Convolutional Neural Networks

Convolutional neural networks have become a popular image classification tool after the excellent classification performance obtained by AlexNet [Krizhevsky *et al.* 2012] in the ImageNet challenge. CNN's consist of a series of convolutional and pooling layers followed by a series of fully connected layers. At each convolutional layer, matrices (kernels) are learned which have dimensions ($k \times k$), and a stride s associated with each layer. These kernels represent commonly occurring features in the dataset. At the low levels, these kernels are often analogous to Gabor filters according to Yosinski

et al. (2014). As we approach higher levels the kernels represent abstract concepts such as parts of or entire objects. Therefore, these networks excel at capturing spatial information. These kernels are convolved with an input matrix with dimensions $(c \times w \times h)$ where c, w, and h represent the number of channels, width, and height of the input image respectively. The more similar a patch in an image is to the feature map the higher the returned value. As these values propagate through the network objects are identified and subsequently, they are used for classification [Yosinski *et al.* 2014].

2.2.1 Mobilenet

Depthwise separable convolutions

Depthwise separable convolutions are an approximation of the classical convolution. Consider a classical convolutional filter with respect to CNNs of dimensionality $(k \times k)$ (while it is not necessary to have a square matrix representing a convolutional filter it is a common practice) we refer to this as a 2D filter as it is described as having 2 dimensions. But images also have a 3$^{\text{rd}}$ dimension, this dimension isn't the height or width of the kernel but rather the number of channels from an image. Given an input image of dimensionality $(c \times w \times h)$ the depth of the kernel of dimensions $(k \times k)$ will be c, therefore, a more accurate representation of the kernel can be described as a tensor of dimensionality $(c \times k \times k)$. Applying a kernel of dimensionality $(k \times k)$ to an image of dimensionality $(c \times w \times h)$ with a stride $s = 1$ we will obtain an output tensor of dimensions $(1 \times w \times h)$, applying n such convolutional filters we obtain an output tensor of dimensionality $(n \times w \times h)$. If we consider the number of parameters required to obtain an output tensor of dimensionality $(1 \times w \times h)$ we have $(1 \cdot c \cdot k \cdot k)$ trainable parameters when applying a single convolutional kernel to an input image. Consequently, if we consider the number of parameters required to obtain an output tensor of dimensionality $(n \times w \times h)$ we have $(n \cdot c \cdot k \cdot k)$ trainable parameters when applying n convolutional kernels to an input image.

Depthwise separable convolutions are implemented to reduce the number of trainable parameters while still maintaining an approximation of classical convolutional filters. Depthwise separable convolutions [Sifre and Mallat 2014] are a modification of the process mentioned above. Instead of applying a convolutional filter to an image of dimensionality $(c \times w \times h)$, they break down a classical convolutional kernel of dimensionality $(c \times k \times k)$ into c individual kernels of dimensionality $(k \times k)$. The c kernels are applied to the c channels in the input image respectively obtaining c $(1 \times w \times h)$ feature maps per convolution filter. Therefore, each channel is assigned its own convolutional filter. This process is called depthwise convolution and was inspired by edge detection and colour filtering. Thereafter, they reduce these c feature maps using pointwise convolution, to perform pointwise convolution, a filter of dimensionality $(c \times 1 \times 1)$ is constructed and applied to the feature maps obtained from the depthwise convolution stage obtaining a $(1 \times w \times h)$ feature map. If we apply n such pointwise convolution filters to the depthwise feature maps we obtain a new image tensor of dimensionality $(n \times w \times h)$. The benefit to this process is not obvious when only using one pointwise filter but it becomes apparent when using n pointwise convolution filters. If we consider the number of parameters required to obtain an out-

put tensor of dimensionality $(1 \times w \times h)$ using depthwise separable convolutions we have $(c \cdot k \cdot k + (1 \cdot c))$ trainable parameters which are more than the number of parameters used in the classical convolution operation on CNNs. If we consider the number of parameters required to obtain an output tensor of dimensionality $(c \times w \times h)$ using depthwise separable convolutions we have $(c \cdot k \cdot k + (n \cdot c))$ trainable parameters which are significantly less than the classical convolution operation on CNNs as the number of output feature maps n grows.

Depthwise separable convolutions are the building block of the MobileNet architecture described in Howard *et al.* (2017). Due to the reduction in the number of trainable parameters and consequently a reduction in the number of multiplication operations. The MobileNet family of architectures can be used on devices with low computational resources and storage while still being able to maintain comparable or results surpassing existing architectures such as VGG-16 [Simonyan and Zisserman 2014] and AlexNet [Krizhevsky *et al.* 2012] on ImageNet [Russakovsky *et al.* 2015] as seen in Howard *et al.* (2017). MobileNet stacks successive depthwise separable convolutions allowing deeper networks to be constructed while maintaining a minimal number of trainable parameters.

MobileNet V2

MobileNet reduces the number of trainable parameters by replacing classical convolutional filters with depthwise separable convolutional filters. Sandler *et al.* (2018) addressed the problem where the total number of convolutions performed in a single forward pass can be reduced. By decreasing the number of channels that need to be processed by each convolutional block, the number of depthwise filters is reduced. However, operating on lower dimensionality data results in less information being extracted from the input, and applying non-linear activation functions to such data results in poorer performance [Sandler *et al.* 2018].

Sandler *et al.* (2018) solved the aforementioned problem through two modifications to the original depthwise separable convolution blocks. The first is a direct modification to the operations performed in the depthwise separable convolution operations. Initially a set of m $(c \times 1 \times 1)$ convolution filters are applied to the input image (where $m > c$) yielding an image of dimensionality $m \times w \times h$. This layer is called an expansion layer with an expansion factor m. The expansion layer is applied to the image before the depthwise filters in an effort to feed a more feature rich image to these filters. Thereafter the depthwise filters are applied to the image to obtain a tensor of dimensionality $(m \times w \times h)$. Finally, the last direct change to the depthwise separable convolution blocks is the concept of a projection layer with an expansion factor c. Similarly to the expansion layer, the projection layer is analogous to the pointwise convolution layer mentioned earlier. The projection layer consists of c $(m \times 1 \times 1)$ convolution filters. By applying the projection layer to the output of the depthwise layer, an output image of dimensionality $(c \times w \times h)$ is obtained. The second modification to the depthwise separable convolution blocks is the inclusion of a residual connection similar to those proposed by He *et al.* (2016). These connections act as a means of accessing activations that were not affected by the depthwise separable con-

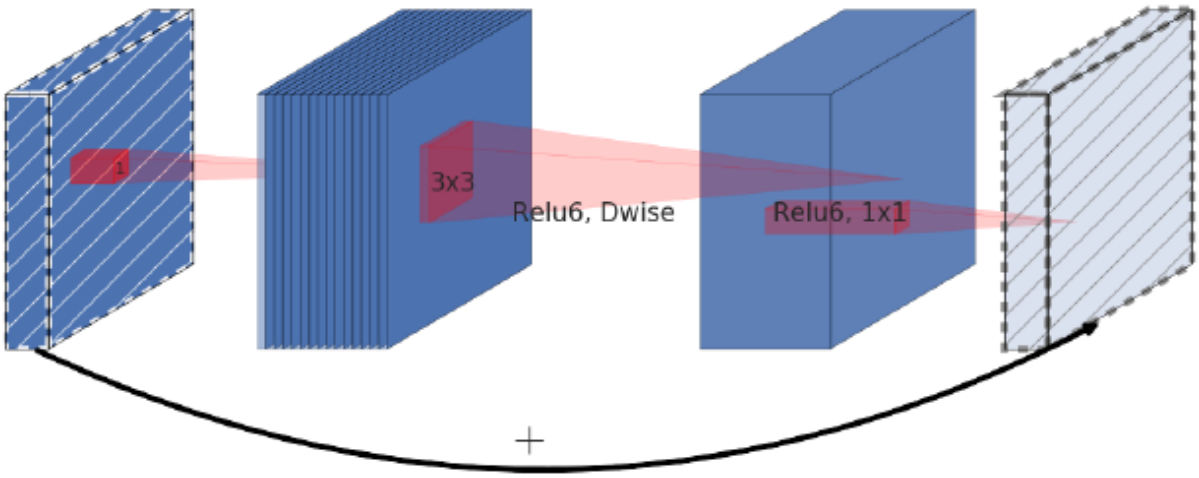


Figure 2.1: Linear Bottleneck Block. *Taken From* Sandler *et al.* (2018)

volution block. This has been shown to aid in the reduction of vanishing gradients in larger networks [He *et al.* 2016]. In MobileNetV2 these residual connections combine the input and output of the depthwise separable convolution block. Unlike He *et al.* (2016) the modified depthwise separable convolution blocks proposed by Sandler *et al.* (2018) does not downscale the input and expand it but they rather expand the image using the expansion layer and contract it at the end of the block using the projection layer. Therefore, this new type of depthwise separable convolution block is termed as an inverted residual block. These inverted residual blocks are used to increase the number of channels from the input internally allowing the block to process higher dimensional data while still maintaining a minimum number of channels passed to subsequent blocks. Sandler *et al.* (2018) implemented the above modifications and constructed a new block namely the linear bottleneck block. An example of a linear bottleneck block can be seen in Figure 2.1.

2.3 Recurrent Neural Networks

Recurrent neural networks (RNN) are neural networks that are primarily used in sequential data processing. They learn conditional probability distributions. Given a sequence they learn to predict the next element [Cho *et al.* 2014]. RNN inputs consist of two components, an item in a sequence and a hidden state. The hidden state of an RNN is the result of processing an element in the sequence together with the hidden state of the previous element see Figure 2.2. The hidden states are a type of memory mechanism which stores important information about previous inputs. Subsequently, this information is used when processing the following elements in the sequence to obtain a new hidden state. As entire sequences are processed by RNN's the hidden state aggregates values from the preceding elements which are used to predict the next element.

The hidden state of an RNN aggregates information from the past states and the present input to the network. As a consequence of processing long sequences, the hid-

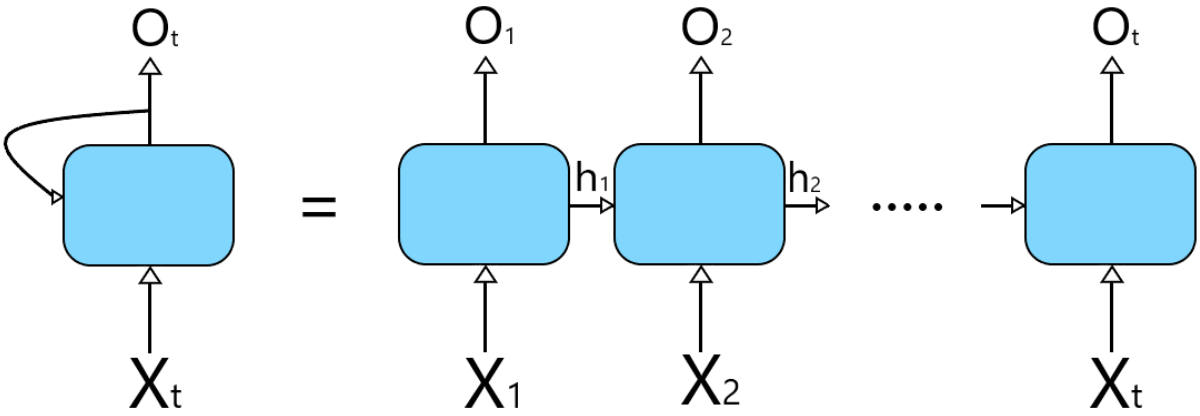


Figure 2.2: Unrolled RNN

den state will encapsulate less information from an older state than recent ones. When training the network through backpropagation (specifically backpropagation through time), vanishing gradients are encountered due to the memory being insufficient and not remembering information about older states [Hochreiter and Schmidhuber 1997].

2.3.1 Long-Short Term Memory Networks

The problem of vanishing gradients was solved by Hochreiter and Schmidhuber (1997) who introduced Long Short-Term Memory (LSTM). LSTM's are an input gating mechanism that can decide whether to store or delete information based on learned weights. This mechanism is called an LSTM cell where the outputs are the previous cell state, hidden state, and the next element in the sequence. The cell consists of three gates, namely the input, forget, and output gates.

$$\begin{aligned} i_t &= \sigma\left(W^i[x_t, h_{t-1}] + b^i\right) \\ f_t &= \sigma\left(W^f[x_t, h_{t-1}] + b^f\right) \\ \tilde{C}_t &= \tanh\left(W^C[x_t, h_{t-1}] + b^C\right) \end{aligned} \tag{2.1}$$

$$\begin{aligned} C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma\left(W^o[x_t, h_{t-1}] + b^o\right) \\ h_t &= o_t * \tanh C_t \end{aligned} \tag{2.2}$$

i_t: Input Gate, f_t: Forget Gate, $\tilde{C}_t$: Candidate State, C_t: Current Cell State,

o_t: Output Gate, h_t: Current Hidden State, x_t: Input Tensor,

h_{t-1}: Previous Hidden State , C_{t-1}: Previous Cell State

The input, forget and output gates as seen in Equations 2.1 perform the same procedure on the input value and hidden state using different weight tensors W^i, W^f or W^o

respectively. The gates form a weighted sum of the input values i_t and hidden state h_{t-1} using traditional single layer fully connected neural networks. The input gate learns what new information to store in the new cell state, as such the network does not learn redundant information which will take preference over older information if it was seen recently. The output gate learns what information to convey from the current cell after the new element has been processed. Lastly, the forget gate allows the network to forget information such as patterns that may have been relevant earlier in a sequence but hold no value later in the sequence. These networks are more sophisticated than traditional RNN's as they gate the inputs and learn what to retain over longer periods of time without replacing it with irrelevant information. While individually all the same operations are applied to the input and previous hidden states by each of the three gates, the manner in which they are combined to form new cell and hidden states assign different weights to each of their values. The new cell state C_t is a combination of the previous cell state and the current candidate state weighted by the forget and inputs gate respectively. The forget gate re-weights the values in the previous cell state whereby values learned to have a high correlation to the current input and previous hidden state values will receive a higher weight. A similar procedure is applied to the current candidate state by the input gate. Finally, the new hidden state is a re-weighting of the current cell state based on the output gate. The gates are given their respective names as the forget gate operates on information from the previous state of the LSTM, the input gate operates on new information passing through the LSTM and the output gate operates on information passing to the next state of the LSTM.

2.4 Video Captioning

Image and video caption generation follows the same pipeline as shown in Figure 2.3. A visual encoder is used to obtain a context vector. These visual encoders are traditionally CNNs used for classification tasks. The context vectors are a representation of the original image by a latent space of one of the fully connected layers in the CNN as seen in Ghanem *et al.* (2018). The context vector can be modified for specific tasks by means of dimensionality reduction using techniques such as PCA or an additional fully-connencted layer. The context vector is then fed to an LSTM network, the hidden state of each cell or unit is a predicted word, words are predicted until an $" < end > "$ symbol is predicted. Vinyals *et al.* (2015) implemented an image caption generator using the aforementioned pipeline. See Figure 2.3 for an example.

2.5 Datasets

The field of video captioning could not have progressed without commonly available large-scale datasets. These datasets each serve their own purpose varying from standard pre-training datasets to common performance evaluation datasets.

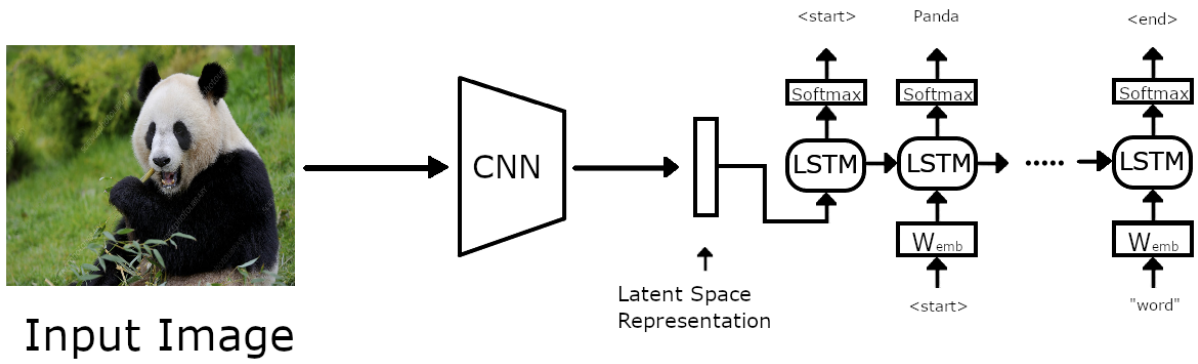


Figure 2.3: Image Caption Generator

2.5.1 Sports-1M

The Sports-1M dataset [Karpathy *et al.* 2014], is a video dataset that consists of approximately 1 million Youtube videos. These videos have been annotated with 487 sport classification labels with approximately 2500 videos per class.

2.5.2 Kinetics

Kinetics-400

The Kinetics-300 dataset [Kay *et al.* 2017] is a high-quality collection of YouTube videos that contain a range of 400 different human actions. The dataset consists of at least 400 clips for each action with an annotation of the class that each clip belongs to with a duration of 10 seconds.

Kinetics-600

The Kinetics-600 dataset [Carreira *et al.* 2018] is a high-quality collection of YouTube videos that contain a range of 600 different human actions. The dataset consists of approximately 500 000 video clips with at least 600 clips for each action with an annotation of the class that each clip belongs to with a duration of 10 seconds.

2.5.3 ActivityNet Captions

The ActivityNet Captions dataset [Krishna *et al.* 2017] consists of 20000 videos with a total length of 849 hours and 100 000 total descriptions each with its own start and end times. The descriptions can vary from sporting activities to mundane activities. On average a video will contain approximately three annotated events with an average of 13.5 words per sentence.

2.6 Principal Component Analysis

In Section 2.4 we described that we apply dimensionality reduction to a context vector obtained from a CNN. Principal Component Analysis (PCA) [Wold *et al.* 1987] is

a technique of feature extraction whereby a subset of components from the original context vector are selected using variance as a measure of uniqueness between each element in the vector. Consider a set of $n \in \mathbb{R}^{(1\times k)}$ context vectors of dimensionality each representing a different datapoint from our dataset. Stacking the context vectors a matrix $D \in \mathbb{R}^{(n\times k)}$ is formed. PCA is applied to a set of data using the following steps:

1. Calculate the mean vector $m \in \mathbb{R}^{(1\times k)}$ for matrix D
2. Subtract m from each context vector in D. $Z = D - m$
3. Calculate the covariance matrix $\Sigma \in \mathbb{R}^{(k\times k)}$. $\Sigma = Z^\intercal Z$
4. Calculate the eigenvectors and corresponding eigenvalues (eigendecomposition) of Σ. $Z^\intercal Z = PDP^{-1}$
    - Note: the eigendecomposition can be calculated using Singular Value Decomposition (SVD). $Z^\intercal Z = PDP^{-1} = U\Sigma V^*$
    - P is a matrix where each column represents the eigenvectors of $Z^\intercal Z$
    - D is a diagonal matrix where each non-zero element is the eigenvalue of the eigenvectors corresponding columnwise to P
5. Sort the eigenvalues according to the magnitude and the corresponding eigenvectors in the same order to obtain the sorted eigenvector matrix $P^* \in \mathbb{R}^{(k\times k)}$
    - To perform dimensionality reduction we selectively keep the first p columns from P^* to form $C \in \mathbb{R}^{(k\times p)}$. The selected columns are referred to as principle components.
    - The eigenvectors can be removed in various ways but the most common methods include arbitrarily selecting p or by selecting components until the total variance described by the principal components surpasses a percentage threshold.
6. Calculate the reduced context vector matrix $Z^* = ZC$, $Z^* \in \mathbb{R}^{(n\times p)}$.

2.6.1 Incremental Principal Component Analysis

Incremental Principal Component Analysis (iPCA) is an adaptation of PCA introduced by Ross *et al.* (2008). This method follows almost the same steps as PCA except it uses an iterative method to update the covariance matrix Σ and its corresponding eigendecomposition. This process was introduced as a means of calculating covariance matrices for large datasets whereby the user is constrained by computational resources (specifically memory). The method used to calculate the covariance matrix at each step is called the Karhunen–Loève theorem Levy and Lindenbaum (1998). Given a dataset of dimensions $(n \times k)$ where n is large. A subset of datapoints are selected at each iteration and the covariance matrix and its subsequent eigendecomposition are updated until all n datapoints have been processed. It is important to note that this procedure does not guarantee that the covariance matrix will be the same as the one

calculated in normal PCA but it has been shown to be a very accurate approximation [Ross *et al.* 2008].

2.7 K-Means Clustering

K-Means Clustering [MacQueen *et al.* 1967] is an unsupervised clustering algorithm. Given a set of n data points $\mathrm{D} \in \mathbb{R}^{(n \times m)}$ where each row corresponds to a separate point in space, the algorithm randomly initializes k, m dimensional points in space named centroids. Iterating through D for each point the distance between said point and each centroid is calculated. Each point is assigned to a single centroid where the distance between the point and centroid is less than the distance from all the other centroids. Thereafter the centroid's coordinates are updated, each centroids new coordinates are set to be the mean of the points assigned to the centroid. This process is repeated until an arbitrary number of iterations have been completed or until the updates to a centroids value are less than a threshold.

2.8 Conclusion

This Chapter described convolutional neural networks, how traditional convolution filters operate and how Mobilenet uses depthwise separable convolutions to reduce the number of trainable parameters. We spoke about the MobileNetV2 architecture and how it modified the depthwise separable convolution blocks to form the linear bottleneck block. We describe recurrent neural networks, long-short term networks, and how their gates operate. The process of generating a single caption for a video was outlined and how the standard pipeline functions. Common datasets which we use or reference when evaluating our results are described. Principal component analysis and the steps used to apply it to a dataset were outlined. We also discussed an approximation to the technique which can be used when computational resources are scarce. Finally, we explained the k-means clustering algorithm and how it operates on a dataset which is used when structuring data for the temporal action proposal and captioning modules for our dense captioning architectures.

Chapter 3

Related Work

3.1 Introduction

Video description is a narration of events that occur in order. This task is useful in a variety of ways, such as in cooking, where obtaining a set of instructions to replicate a task is invaluable. Yu *et al.* (2016) explored a solution to this problem by first using a 3D convolutional neural network (3D CNN) to encode a video then passing the encoded features to an LSTM decoder. To maintain context they took advantage of Hierarchical RNN's, which can be thought of as a stacked RNN whereby the output of an RNN will be used as input to another. The captions were generated for actions in a clip. These captions were passed to the second LSTM which processed the caption and adjusted the context so as to maintain contextual consistency between multiple sentences in a paragraph.

In this Chapter we describe the task of action recognition and how 3D CNN's are used to process videos in clips whereby they identify patterns by considering multiple time segments instead of processing a video on a frame by frame basis. We describe the widely popular C3D network and we outline modifications made to the MobileNetV2 architecture and the linear bottleneck block to process video clips instead of single images.

We discuss previous work in identifying temporal segments in videos and thereafter provide a description of the full dense video captioning pipeline introduced by Krishna *et al.* (2017).

We outline the concept of self-attention and explain two separate implementations, the Attention Estimator module which we use to augment models for our research, and the Squeeze and Excitation module which is used to extend the capabilities of the MobileNetV2 architecture to form the basis of the MobileNetV3 architecture.

3.2 Action Recognition and 3D Convolutional Neural Networks

Action recognition is the process of identifying actions in a video given a series of frames. A predominant technique for action recognition in videos are 3D CNN's due to their ability to capture spatio-temporal information. In the case of dense captioning, the Convolutional 3D (C3D) network pretrained on the Sports1M dataset is frequently compared to other architectures due to the ActivityNet Captions features being readily available. C3D was first used for feature extraction in dense video captioning by Krishna *et al.* (2017). While the first dense video captioning paper may have used C3D for feature extraction, more recent architectures have used the same network for feature extraction [Aafaq *et al.* 2019, Yao and Li 2018, Shen *et al.* 2017, Wang *et al.* 2018, Li *et al.* 2018]. It should be noted that Yao and Li (2018) obtained second place in the 2018 ActivityNet Challenge dense video captioning task using C3D as a feature extractor [Ghanem *et al.* 2018].

3.2.1 C3D

Tran *et al.* (2015) proposed a 3D CNN architecture for spatiotemporal feature extraction from videos. The network consists of 5 convolutional layers, each followed by a max-pooling layer and two fully connected layers with 4096 neurons each. The 3D convolutional kernels have dimensions $(3 \times 3 \times 3)$, that is the kernels have a temporal depth of 3 and spatial width and height of 3. The temporal depth of 3 was chosen after experimentation whereby kernels of depth larger than 3 achieved diminishing returns on the UCF101 dataset [Tran *et al.* 2015, Soomro *et al.* 2012]. The comparison was made when considering kernels of depth $1, 3, 5$ and 7, with spatial dimensions of 3 and 5. They found a similar pattern for both space and time dimensions, that is after a depth, width or height of 3 the network will experience diminishing accuracies. Their results showed a maximum accuracy of 85.2% using only RGB channels and 90.4% using all available features such as optical flow [Horn and Schunck 1981] on the UCF101 dataset. On the Sport1M dataset, they achieved a maximum accuracy of 85.2%. *Note that C3D can only process* 16 *frames at a time as such most methods which implement C3D refer to this as their time resolution* δ.

3.2.2 3D MobileNet

3D Depthwise Separable Convolutions

In Section 2.2.1 we described depthwise separable convolutions as an alternative to classical convolution kernels by means of performing two distinct operations. First, apply a convolution filter to each channel independently (this operation is only performed once), thereafter we perform a pointwise convolution on each pixel across all channels (the number of pointwise convolution filters is equal to an arbitrary number of output channels). An image input can be represented by $I \in \mathbb{I}^{(c \times w \times h)}$ consequently we can represent a video sequence by $V \in \mathbb{I}^{(c \times d \times w \times h)}$ where d represents the number

of frames in the video sequence. Adapting depthwise separable convolutions to operate on 3D data such as video sequences requires a minimal amount of modification to the process. The pointwise convolution filters for 3D input tensors are given by $(c \times 1 \times 1 \times 1)$, it is important to note that the temporal dimension in these pointwise convolution filters are only considered as individual points and no temporal dependency is maintained. The depthwise convolutional filters for 3D input tensors are given by c individual $(1 \times k \times k \times k)$, unlike the pointwise convolutions these depthwise convolutions consider k time segments at each pass thus temporal dependency between frames in the video are preserved only in this step of the depthwise separable convolution procedure.

3D MobileNetV2

Kopuklu *et al.* (2019) adapted the original MobileNetV2 architecture proposed by Sandler *et al.* (2018). The changes made to the MobileNetV2 architecture by Sandler *et al.* (2018) were minimal, the first change was adapting all the linear bottleneck blocks first by extending the expansion and projection layers to perform the same operations as the pointwise convolution filters, secondly, the depthwise convolution filters are expanded to include a temporal depth dimension as described above for 3D convolutions. This architecture is designed to operate on video clip inputs with temporal depth $d = 16$ frames. The initial convolution layer is not a depthwise separable convolution layer but rather a traditional convolution layer with stride $(1 \times 2 \times 2)$ where there is a single step in the temporal dimension but two steps in the spatial dimensions.

The 3D MobileNetV2 architecture they implemented had 3.12 million parameters. They found that on the UCF101 and Kinetics-600 action recognition datasets that the implemented architecture achieved top-1 accuracies of 81.60% and 50.65% on the datasets. These scores are an improvement from the 3D MobileNet architecture which achieved scores of 70.95% and 40.07% on the UCF101 and Kinetics-600 datasets respectively.

3.3 Temporal Action Proposals

The encoded features obtained from the feature extractor (action classifier, we will use these terms interchangeably) are then passed through an event proposal framework (temporal action proposal framework). Given a video, these networks identify windows in which actions occur which are subsequently passed to a captioning network. A common technique to identify these windows at varying scales is to apply multiple windows at differing scales [Shou *et al.* 2016]. Another method is to use a fixed window but to perform calculations between each of the considered windows to identify if actions start and end times occur between the windows [Escorcia *et al.* 2016]. The technique of considering windows at different scales may lead to windows being processed multiple times [Buch *et al.* 2017].

A commonly used event proposal network is the single-stream temporal action proposals (SST) framework proposed by Buch *et al.* (2017). SST is an improvement on

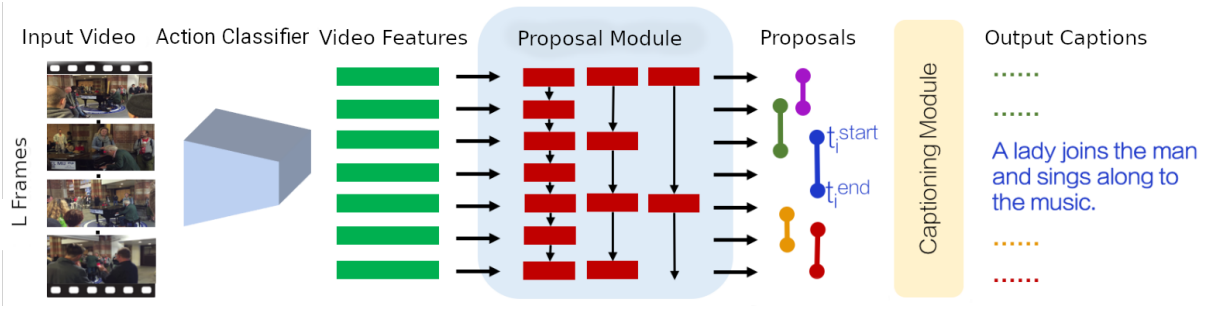


Figure 3.1: Generic Dense Video Captioning Pipeline. *Adapted from* Krishna *et al.* (2017)

the Deep Action Proposals for Action Understanding (DAPs) framework [Escorcia *et al.* 2016]. The SST framework identifies windows in which actions occur without the need of processing overlapping windows separately [Buch *et al.* 2017]. SST uses C3D as a feature extractor to obtain spatial and temporal information from videos. Given a video with N frames and the temporal resolution of $\delta = 16$ frames the number of extracted feature vectors is $T = \frac{N}{\delta}$. Thereafter the sequence of these feature vectors $\{f_t\}_{t=1}^{T}$ is processed by a RNN-based sequence encoder [Buch *et al.* 2017]. This encoder provides k action proposals at each time step along with a confidence score for each of the k proposals [Buch *et al.* 2017]. Each of these proposals starts at the beginning of the time step and span some time interval between $[t, t + \delta]$. To identify actions between time intervals from these proposals they implement thresholding by confidence and non-maximum suppression so as to remove overlapping actions with lower confidence scores [Buch *et al.* 2017].

Context improvement between captions has become perhaps the largest area of dense video captioning. While the pipeline in Figure 3.1 may not change, the methods in which the features, temporal proposals, and captions are generated change significantly. Wang *et al.* (2018) proposed an improvement to the SST framework introducing the Bidirectional SST for better temporal action proposals, while there are minimal changes to the underlying framework they consider not only passing their extracted features through SST in the normal order but rather do a second pass and provide the data to SST in the reverse order. Now that they have two sets of proposals they perform element-wise multiplication among the vector pairs to form a fusion between the context vectors. Their results show that this bidirectional method improves caption quality significantly. On the ActivityNet Captions dataset they achieved scores that were double that of Krishna *et al.* (2017) on the METEOR metric.

3.4 Dense Video Captioning

Dense video captioning is the process of localizing and captioning all events that occur in a video (sequence of temporally ordered images). Each event has an associated time frame and a caption associated with it. The task was first proposed by Krishna *et al.* (2017). They identified a gap in current work at the time, that is video captioning only works on very short clips containing a single event and there is no overlap between

the frames used in the other events. CNN's or 3D CNN's generate feature vectors which are passed to a captioning module. These convolutional feature extractors can only process clips with a fixed number of frames. If the number of frames in a clip or video was longer than a fixed number of frames, they are partitioned into non-overlapping segments and processed individually. Thereafter they find the average of these context vectors and pass it to the captioning module. As a means of resolving this problem Krishna *et al.* (2017) introduced an intermediary module between the feature extractor and the captioning module. The module termed the event proposal module (we leave the term generic) receives a series of inputs $\{f_t = F(v_t, v_{t+\delta})\}_{t=0}^{n}$ where $F(\cdot)$ is the convolutional feature extractor and $\{v_t, v_{t+\delta}\}$ are a series of δ consecutive frames from the input video. The features $\{f_t\}_{t=0}^{n}$ are subsequently fed into the event proposal, module in their case an LSTM network, which aggregates evidence over time. Whenever the LSTM network detects an event that hidden state is passed to the captioning module where a caption is generated.

The second contribution Krishna *et al.* (2017) made was the addition of the ActivityNet Captions dataset. The availability of this dataset prompted the addition of the dense captioning task to the challenge. Since the inception of the dataset and techniques used for dense video captioning there have been a flurry of techniques and variants proposed as challenge submissions and independent papers exploring the domain [Shen *et al.* 2017, Aafaq *et al.* 2019, Li *et al.* 2018, Yao and Li 2018, Wang *et al.* 2018]. The full dense video captioning pipeline can be seen in Figure 3.1.

3.5 Self-Attention

Attention or feature gating is a technique that allows models to assign weights to individual input values. A mechanism that processes an input feature and identifies salient regions is referred to as self-attention by Xie *et al.* (2018). Xie *et al.* (2018) implemented context gating on their S3D network which applies separable convolutions on the Inception network [Szegedy *et al.* 2015]. They found that on the UCF101 action recognition dataset that the network they implemented had fewer parameters than the 3D inception network (I3D) and outperformed it by at least 3% on the top-1 accuracy achieving a score of 96.8%.

3.5.1 Attention Estimator Module

Jetley *et al.* (2018) proposed a method of self-attention whereby model predictions are conditioned on a subset of the extracted features extracted at different scales. They found that extracting features from different scales in a CNN helps inference whereby information from background features are used when they are traditionally filtered out by convolution filters as the input passes through the network. Jetley *et al.* (2018) did not provide a specific name for this technique as such we will refer to it as an attention estimator module for the duration of this dissertation. The attention estimator module is applied in the following manner:

- Extract all feature maps from layer L_i in a CNN

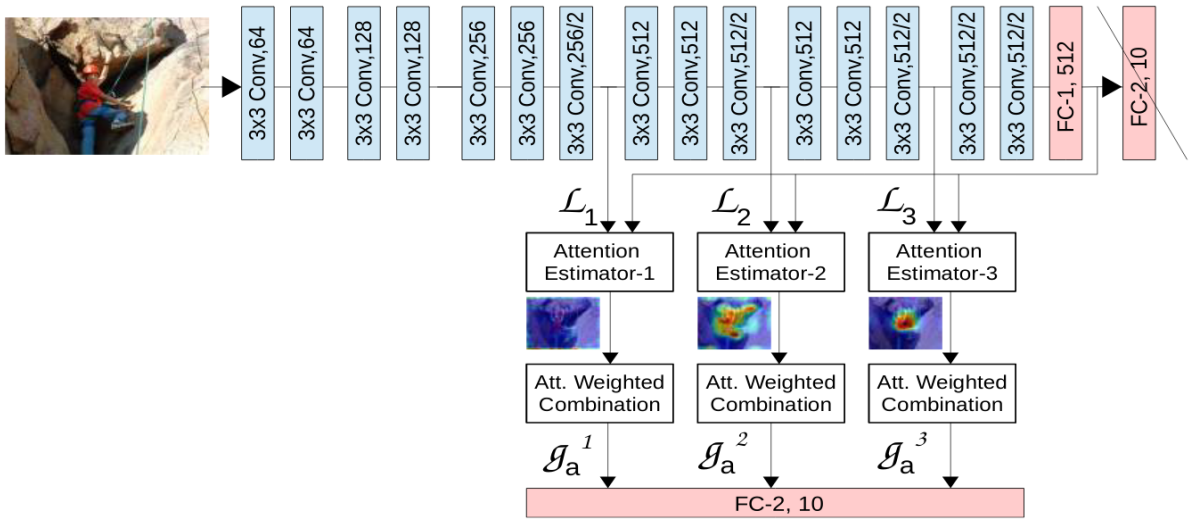


Figure 3.2: Implementation of the Attention Estimator Module proposed by Jetley *et al.* (2018). *Taken From* Jetley *et al.* (2018)

- Calculate the compatibility score C_i

$$C_i = \langle \mathbf{u}, L_i + \mathbf{g} \rangle = \mathbf{U}[L_i + \mathbf{g}] + \mathbf{b} \tag{3.1}$$

- Calculate the attention weights A_i, where σ is the Softmax function

$$A_i = \sigma(C_i) \tag{3.2}$$

- Calculate the final output of the attention mechanism g_a

$$g_{i,a} = [\Sigma_{j=1}^{n} A_{i,j} \cdot L_{i,j}^{c}]_{k=1}^{c} \tag{3.3}$$

Where $\mathbf{U}$ is a single convolutional layer with one output channel, $\mathbf{g}$ a global feature descriptor obtained from the output of the fully connected layer before the classification layer and $\mathbf{b}$ is the bias. The operations performed in Equation 3.1 is first the addition between the global feature descriptor and the output L_i from layer i in the CNN. The notation $< \cdot, \cdot >$ indicates the dot product between two values. Therefore $\langle \mathbf{u}, L_i + \mathbf{g} \rangle$ is the dot product between the added features $L_i + \mathbf{g}$ and a learned vector $\mathbf{u}$. The series of operations to obtain the compatibility score can be understood as adding the global feature descriptor and the output of layer i, thereafter passing them through a single layer CNN with one output channel.

If we consider the feature maps from $L_i \in \mathbb{R}^{(c \times w \times h)}$, the compatibility score $C_i \in \mathbb{R}^{(1 \times w \times h)}$ is a tensor that has a different score for every pixel in L_i, where the same score is representative of a pixel across all channels. The attention weights $A_i \in \mathbb{R}^{(1 \times w \times h)}$ is a tensor that has a probability for every pixel in L_i after applying the sigmoid function to C_i. Finally, the attention module output $g_{i,a} \in \mathbb{R}^{(c \times 1 \times 1)}$ are the representative values

of each channel where each value is the sum across all spatial dimensions of each channel respectively.

Jetley *et al.* (2018) found that the global feature vector g (the logits of the final layer of the CNN) was redundant when using the parametrized compatibility score in 3.1.

These Attention Estimator modules are placed at discrete layers in a CNN which have different spatial resolutions. This is done as a means of providing information to a classification layer with higher level information from earlier layers. Features from earlier layers in a CNN encompass more information related to the general scene of the input image. Whereas, features from later layers mainly provide information related to objects of interest to the classification task Jetley *et al.* (2018).

Given a set of $g_{i,a}$ attention vectors obtained from different layers in the CNN we concatenate them and pass them through an Artificial Neural Network (ANN) to obtain the logits used for inference. See Figure 3.2.

3.5.2 Squeeze and Excitation

Hu *et al.* (2018) introduced squeeze and excitation networks. This form of self-attention assigns a single scalar value (weight) associated with each channel of an input tensor. This technique was designed for ease of integration into existing networks while imposing a minimal overhead to the number of computations performed [Hu *et al.* 2018]. Squeeze and excitation networks implement squeeze and excitation blocks, consider a tensor $T \in \mathbb{R}^{(c \times w \times h)}$, these blocks calculate the average value for each channel and concatenates these values into a vector. This vector is subsequently passed through an ANN with two fully connected layers. The number of outputs for this two layer ANN is c, thereafter each predicted value is multiplied with each channel respectively assigning different weights to each channel. *Note: In traditional CNNs, each channel is assigned equal importance with a weight of 1.* An implementation of the squeeze and excitation module can be seen in Figure 3.3.

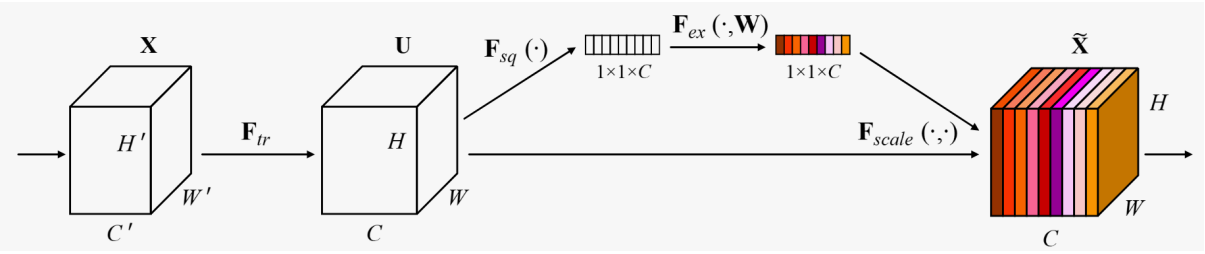


Figure 3.3: Implementation of the Squeeze and Excitation Module proposed by Hu *et al.* (2018). *Taken From* Hu *et al.* (2018)

3.6 MobileNetV3

Tan *et al.* (2019) introduced an improvement to the MobileNetV2 architecture by means of a neural architecture search. The MobileNetV2 architecture consists of linear bottleneck blocks, each linear bottleneck block consists of 3 main layers an expansion

layer, a depthwise convolutional layer, and a projection layer. Tan *et al.* (2019) modified the linear bottleneck blocks by adding a squeeze and excitation block between the depthwise convolutional layer and projection layer. This was done as a means of re-weighting the channels which were expanded after the expansion layer before they were projected down to the original number of channels.

The MobileNetV3 architecture was designed by means of a Neural Architecture Search (NAS) [Zoph and Le 2016] and NetAdapt [Yang *et al.* 2018] to construct the intermediate layers of the network [Howard *et al.* 2019]. Specifically the number of linear bottleneck blocks and the number of channels constructed by the expansion layers were designed using the above two techniques. The maximum number of layers and blocks were constrained by a maximum inference time $T = 80ms$. Secondly Howard *et al.* (2019) proposed a replacement for all ReLU nonlinearity activation functions specifically the h-swish activation function.

$$\text{h-swish}[x] = x \cdot \frac{\text{ReLU6}(x+3)}{6} \tag{3.4}$$

The second term in Equation 3.4 is an approximation to the sigmoid activation function. This approximation is computationally cheap compared to the sigmoid function. The swish activation function was shown to improve performance of neural networks when replacing all ReLU activations [Ramachandran *et al.* 2017, Elfwing *et al.* 2018, Hendrycks and Gimpel 2016]. A plot of the h-swish activation function can be seen in Figure 3.4.

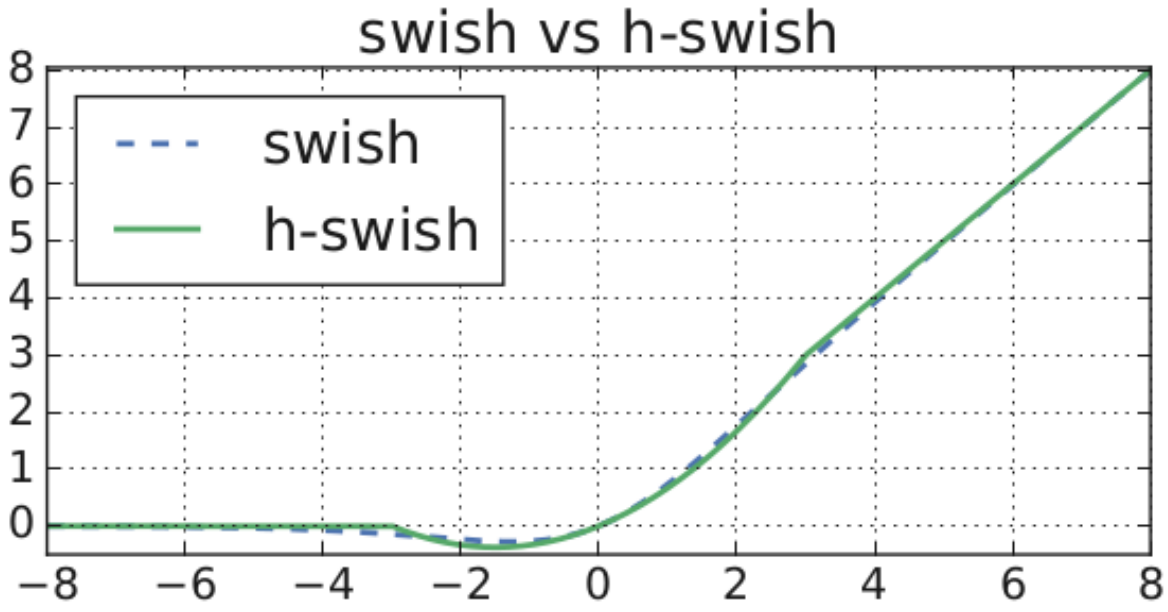


Figure 3.4: A plot of the h-swish activation function. *Taken From* Howard *et al.* (2019)

The final modification that Howard *et al.* (2019) proposed was a redesign to the expensive starting and final layers of the MobileNetV2 architecture. They placed the projection layer from the final linear bottleneck layer after an average pooling layer,

and the depthwise and projection layers of the previous linear bottleneck layer were removed. Finally, the number of filters in the first convolution layer was reduced from 32 to 16.

Howard *et al.* (2019) found that their network outperformed the MobileNetV2 architecture on ImageNet with top-1 accuracies of 75.2% compared to 72.0% while still maintaining a faster inference time of 51ms compared to 64ms respectively.

3.7 Conclusion

In this Chapter we discussed the 3D convolution operation, how 3D convolutional networks are used for video action classification. Furthermore, we discussed the 3D MobileNetV2 architecture and how the 3D linear bottleneck blocks operate. We went over the process of generating temporal action proposals and some methods of generating them. We described the process of dense video captioning and went over the generic pipeline that is still followed in current work. We described the concept of self-attention and some attention mechanisms. We described the self-attention mechanism that we will be modifying and the self-attention mechanism that was used in forming the MobileNetV3 architecture. Finally, we describe the MobileNetV3 architecture and how it was constructed.

Chapter 4

Research Methodology

4.1 Introduction

In this chapter, we outline the overall procedure which was used to train our models and achieve our results. We will describe the method which was used to prepare the Kinetics-368 dataset, with transformations applied to each downloaded video. We will outline the procedure in which we processed the ActivityNet Captions dataset, transformed the videos, timestamps, and captions respectively. We describe the action classifiers used, the modified attention estimator, and how it was integrated into our networks. We describe the temporal proposal and caption decoder LSTM networks and all transformations applied to the data that they process. We outline the evaluation metrics that are used to answer our research questions and to verify our hypothesis. Lastly, we explain the training and inference methods used for this project.

All the code written and used for this project can be found at the following GitHub Repository: Masters Code

4.2 Data Preparation

4.2.1 Kinetics-368

We constructed the Kinetics-368 dataset which is a strict subset of the Kinetics-600 dataset. This new subset contains 368 unique action classes taken from the intersection of the Kinetics-400 and Kinetics-600 datasets. This dataset was constructed from the Kinetics-600 dataset as each of our 368 classes contains a minimum of 600 videos per action class as opposed to the Kinetics-400 dataset which contains a minimum of 400 videos per action class. Furthermore, we used the Kinetics-600 dataset videos as a means to increase the variability in data per class while maintaining a subset of the classes in an effort to accommodate the computational resources available to us.

Each video is associated with a video ID which is used to download them from YouTube. All Kinetics datasets have start and end times associated where each video with a duration of 10 seconds per video. Furthermore following Tran *et al.* (2015) all videos are

resized to have spatial dimensions of (128×171).

The Kinetics-600 dataset is pre-split into three sets, training, validation, and test sets. The Kinetics-368 dataset is a strict subset of the Kinetics-600 dataset therefore all videos in each split are a subset of the original videos in the Kinetics-600 dataset's splits. The test set is reserved for evaluation on the evaluation server as such we do not have access to the class labels to evaluate our action classifiers. Therefore we split the Kinetics-368 dataset's validation set into two separate sets, the first validation, and second validation sets. Each of the first and second validation sets contain half the videos from the original validation set. The second validation set is reserved for evaluation as our test set which we use to report our results. The video IDs for each validation split are randomly selected and stored in separate lists, this is done as a means to avoid any bias towards a specific action class during evaluation as each class in the Kinetics-368 dataset is balanced. All results are reported on the second validation set while the training and first validation sets are used during the training phase.

4.2.2 ActivityNet Captions

The ActivityNet Captions dataset associates video IDs, start and end times with each video. The videos are not of uniform duration and may have multiple time segments. Each time segment is assigned a caption describing the event performed in the segment. The structure of the provided data is given in Table 4.1.

Table 4.1: ActivityNet Captions Data Structure

Video_id	v_12Ab67fG45D
Duration	D
Timestamps	[[s0, e0], [s1, e1]]
Sentences	["This is sentence 1.", "This is sentence 2."]

The data preparation steps and transformations applied to the videos and annotations were taken from Wang *et al.* (2018). If we consider a video V of duration D, utilizing an action classifier (AC) that processes a video sequentially in clips of length δ frames. If V has a framerate of 30 frames per second, the total number of frames in V is $L = 30 \cdot D$. Discretizing V we obtained $N_v = \frac{L}{\delta}$ clips. We reshaped each of these clips' spatial dimensions to (112×112) and passed each clip through an AC to obtain N_v feature vectors per video. *Note: N_v does not need to be the same for each video as they have differing durations D.* These feature vectors were not obtained from the final fully connected layer in the 3D CNN, following Krishna *et al.* (2017), we extracted them from previous layers in an effort to maintain a representation of the extracted features for each clip. *In Appendix A, see layers marked with "FV" in each model diagram. These layers were used to generate the feature vectors for the dataset.* The features extracted from the attention augmented models were obtained from the concatenation of the Attention Estimator outputs to provide the Temporal Action Proposal and Captioning modules with features from different temporal and spatial resolutions. Each feature vector represents δ frames from V. Thereafter we applied iPCA to each of these feature vectors to reduce each one to a vector of dimensionality (1×500). Furthermore we

averaged $R = \frac{64}{\delta}$ consecutive feature vectors where each new feature vector represents 64 consecutive frames in V which will be referred to as T (rem).

The ActivityNet Captions dataset is pre-split into four sets, the training, first validation, second validation and test sets. The test set is reserved for evaluation on the ActivityNet Captions evaluation server as such all results we report on are based on results obtained from the second validation set while we used the training and first validation set during the training phase.

Temporal Segments

Given a set of timestamps as seen in Table 4.1, they are not structured in a format that is feasible for usage during the training and inference phases. Videos may have a varying number of events and subsequently, the number of timestamps will vary. Furthermore, the start and end times do not necessarily have start and end times within the same region in time. Therefore we restructured the data in a trainable format. This process involved:

- Utilizing K-means to identify K duration cluster centroids where each centroid represents common durations for individual events in our entire dataset.
- Calculate the approximate start and end times utilizing the centroids.
- Form a matrix $M_T : (R \times K)$ and assign 0 to all values aside from those with small tIoU (threshold Intersection over Union) scores between the approximated and ground truth start and end times.

Obtaining all start and end timestamp pairs from the ActivityNet Captions dataset we calculated the duration for each event $d = \{e_i - s_i\}_{i=1}^{n}$. Utilizing the K-means algorithm we clustered the durations d. We specified $K = 120$ cluster centroids, where each centroid represents some average duration. We then sorted these cluster centroids by magnitude to obtain a list C.

Wang *et al.* (2018) proposed a bidirectional temporal proposal module, thus it involves passing events in the forward and reverse order. This is done to capture future context as well as sequential context as a means to improve start and end timestamp prediction. Below we outline the two algorithms used. In Algorithm 1 we show how the data is structured for features being passed to the temporal proposal module in the sequential manner. In Algorithm 2 we show how to structure the temporal proposals in the reverse order.

Algorithm 1: Generate Forward Time Segments

Result: Matrix Representing the Forward Time Segments
K-means Cluster Centroids: C ;
Video Feature Vectors: T ;
Duration: D ;
Video Timestamps: T_s ;
GT_Proposal_FW: Zeros:$(len(T) \times len(C))$;
while *stamp_id* $<$ *len(T_s)* **do**
    start, end = $T_s[stamp_id]$;
    end_id = $\frac{end}{D} \cdot len(T) - 1$;
    start_id = $\frac{start}{D} \cdot len(T) - 1$;
    mid_id = $\frac{0.5 \cdot (start+end)}{D} \cdot len(T) - 1$;
    i = mid_id;
    while *i* $<$ *len(T_s)* **do**
        overlap = false;
        while *anchor_id* $<$ *len(C)* **do**
            end_pred = $\frac{i+1}{len(T)} \cdot D$;
            start_pred = end_pred - C[anchor_id];
            curr_iou = calc_IoU([start, end], [start_pred, end_pred]);
            anchor_id += 1;
            if *curr_iou* $>$ *iou_thresh* **then**
                GT_Proposal_FW[i, anchor_id] = 1;
                overlap = true;
            else if *overlap* **then**
                break;
        end
        i += 1
    end
    stamp_id += 1
end

Algorithm 2: Generate Backwards Time Segments

Result: Matrix Representing the Backward Time Segments
K-means Cluster Centroids: C ;
Video Feature Vectors: T ;
Duration: D ;
Video Timestamps: T_s ;
GT_Proposal_BW: Zeros:$(len(T) \times len(C))$;
while *stamp_id* $<$ *len(T_s)* **do**
    start, end = $T_s[stamp_id]$;
    end_id = $\frac{end}{D} \cdot len(T) - 1$;
    start_id = $\frac{start}{D} \cdot len(T) - 1$;
    mid_id = $\frac{0.5 \cdot (start+end)}{D} \cdot len(T) - 1$;
    i = mid_id;
    while *i* $<$ *len(T_s)* **do**
        overlap = false;
        while *anchor_id* $<$ *len(C)* **do**
            end_pred = $\frac{i+1}{len(T)} \cdot D$;
            start_pred = end_pred - C[anchor_id];
            curr_iou = calc_IoU([start, end], [start_pred, end_pred]);
            anchor_id += 1;
            if *curr_iou* $>$ *iou_thresh* **then**
                i_bw = len(T) - 1 - (start_id + end_id - i);
                GT_Proposal_BW[i_bw, anchor_id] = 1;
                overlap = true;
            else if *overlap* **then**
                break;
        end
        i += 1
    end
    stamp_id += 1
end

Intepreting the Time Segment Data
Consider a matrix generated using Algorithm 1. Let $GT_Proposal_FW \in [0,1]^{(len(T)\times len(C))}$.

$$\text{GT_Proposal_FW} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 0 \end{bmatrix}$$

Each row in GT_Proposal_FW corresponds to a feature vector in T and each column corresponds to an anchor in C. In Algorithm 1 GT_Proposal_FW is initialized as a zero matrix, thereafter approximate start and end times are calculated utilizing anchors in C and indices from T. An approximate end time is calculated from a given index, thereafter the approximate start time is calculated by subtracting an anchor value from the end time. The IoU score between the approximate and true timestamps is calculated and a value of 1 is assigned to GT_Proposal_FW. This value is assigned to the feature index where the event approximately ended. For each timestamp, there is a possibility of multiple approximate timestamps for a single true timestamp. Using the calculated approximate ending indices the approximate start time is calculated using the K-means anchors. Each anchor represents an average duration of common event lengths in the ActivityNet Captions dataset.

Considering that there are approximately three timestamps on average per video we did not limit the approximate timestamps to the ones with the maximum IoU score but rather we used a threshold. There are many feature vectors T for a video and assigning a single value for each timestamp resulted in a very sparse matrix with most entries being 0 and approximately three values which are 1. This sparse representation makes it very difficult to learn the correct timestamps as it is easier for the LSTM to predict the value of 0 for all entries and obtain a very high accuracy.

Captions

We assigned a unique index to each word in the ActivityNet Captions dataset's vocabulary. We then tokenized each sentence and removed all punctuation marks aside from commas and apostrophes. Tokenization is the process of breaking a piece of text into individual linguistic units. Thereafter we encoded the tokenized sentence using the indices assigned to each word in the vocabulary.

We enforced that each sentence has a maximum of 30 words. If a sentence is shorter than 30 words we padded the missing values with a placeholder. This does not affect the training step as we constructed a binary mask indicating if a word is a valid part of the sentence or if it is a placeholder index.

Considering Algorithm 1, if a feature ID is considered a valid end given the true timestamp, we assign the given timestamp's encoded sentence to that feature ID.

4.3 Action Classifiers

Please see the Appendix A for visualizations of the specific architectures.

The goal of the action classifiers is to generate a set of T feature vectors per video, by leveraging a trained action classifier to extract features from an unseen video dataset.

We used the Wang *et al.* (2018) pipeline as our base configuration. Their temporal proposal and caption module are trained on features extracted from the ActivityNet Captions dataset on a C3D model pretrained on the Sports1M dataset. The C3D action classifier is trained using a temporal depth of 16 frames.

For this dissertation, we constructed or replicated 6 network architectures and evaluated the features extracted from them compared to the original results produced by Wang *et al.* (2018).

We replicated the C3D model first introduced by Tran *et al.* (2015), and trained 2 separate instances of this model. We trained the first instance using a temporal depth of 8 frames and the second using a temporal depth of 16 frames. We refer to these architectures **C3D-8** and **C3D-16** respectively for the duration of this dissertation.

We extended the MobileNetV3 (Large 1.0x) architecture introduced by Howard *et al.* (2019) by modifying the Linear BottleNeck blocks to operate on 3D data. We utilized the same technique as Kopuklu *et al.* (2019) described in Section 3.2.2. This architecture only works with input video clips with a temporal depth of 16. We refer to this architecture as **3D MobileNetV3** for the duration of this dissertation.

3D Attention Estimator Module

We now consider the Attention Estimator Module introduced by Jetley *et al.* (2018), which was described in Subsection 3.5.1. We extended this technique to operate on 3D data. In the first step of the Attention Estimator Module pipeline, we calculated the compatibility score C_i. This was implemented with a single 2D convolution kernel $c_k \in \mathbb{R}^{(1\times1)}$ yielding a single feature map with a representative value of each pixel. Our extension to this process only modified the convolution kernel used, consider an input tensor $I \in \mathbb{R}^{(c\times t\times w\times h)}$, we modified c_k and obtained our 3D convolution kernel $c_k \in \mathbb{R}^{(t\times1\times1)}$, where t represents the temporal depth of I. The application of the compatibility score and the calculation of the attention module output remains the same as described in Subsection 3.5.1.

Utilizing this modified Attention Estimator Module, we augmented the **C3D-8**, **C3D-16** and **3D MobileNetV3** respectively. In the **C3D-8** and **C3D-16** architectures we added three separate Attention Estimator Module at layers 6, 9 and 12 respectively. In the **3D MobileNetV3** we also added three separate Attention Estimator Module at layers 6, 12, and 15 respectively. For the duration of this dissertation we will refer to these architectures as **C3D-8 Attn**, **C3D-16 Attn** and **3D MobileNetV3 Attn** respectively. *Note: when implementing the Attention Estimator modules in existing networks we utilized the same number of fully connected layers as in the base architectures. We rescaled the*

number of output neurons in the new fully connected layers according to the dimensionality of the attention concatenated vector.

While the C3D model that we used for our baseline comparison is trained on the Sports1M dataset with a temporal depth of 16 frames, we trained C3D and its attention augmented variants using 8 frames and another set of the aforementioned architectures using 16 frames. We trained the C3D models using differing temporal depths to identify the impact temporal depth has on the temporal proposal and the caption predictions.

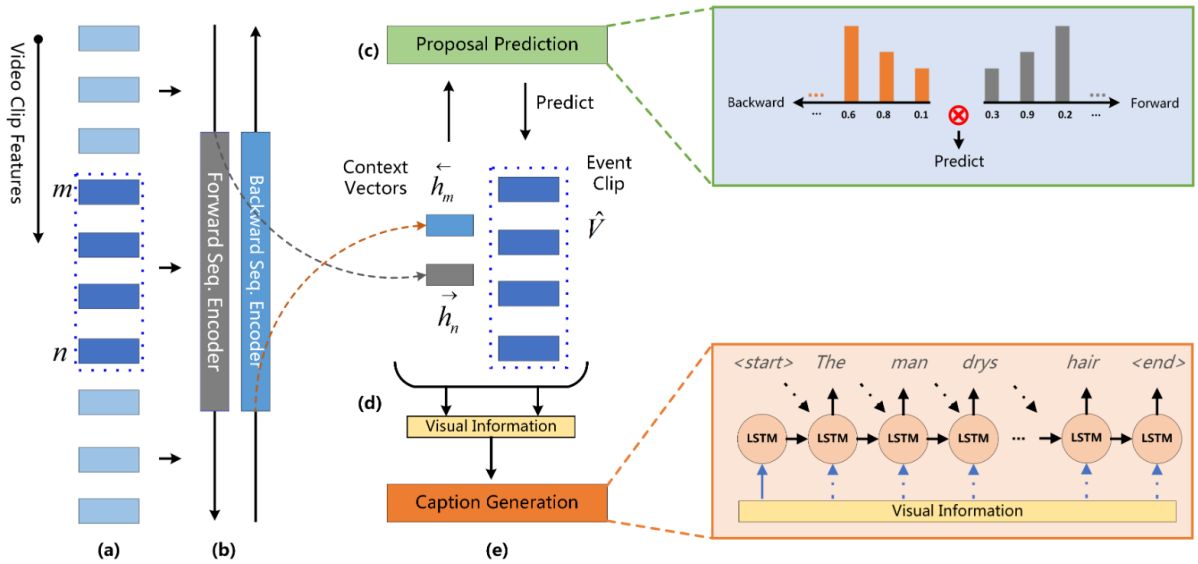


Figure 4.1: BiDirectional SST Pipeline. *Taken from* Wang *et al.* (2018). (a) A video is encoded by an action classifier and the features are reduced to a fixed size using PCA. (b) The encoded features are passed through a bidirectional LSTM encoder. (c) The forward and backward LSTM hidden states are processed by a proposal prediction ANN and are jointly used to generate temporal proposal predictions. (d) The hidden states at the start and end of an event are concatenated with the encoded visual feature vectors. (e) The concatenated features are passed through a caption generation LSTM decoder to generate sentences.

4.4 Temporal Action Proposal Module

The goal of the temporal action proposal module is to generate a set of valid timestamps which represent a region in time where events start and end in a given video. Considering a video V we form $T = \{\mathbf{t}_1, \mathbf{t}_1, ..., \mathbf{t}_N\}$ as described in Subsection 4.2.2. Each feature vector in T represents 64 continuous frames in V.

Segments (b) and (c) in Figure 4.1 represent the Bidirectional SST temporal action proposal module that was constructed by Wang *et al.* (2018), they passed T in both the forward and reverse order in an effort to capture future context from T in a single forward pass of the data.

Wang *et al.* (2018) used a single layer LSTM to encode the sequential data in T. This process isolates and retains features across the sequence of temporal data. The output which is obtained from the LSTM after passing the T in the forward order results in a set of N hidden states $\overrightarrow{\mathbf{H}} = \{\overrightarrow{\mathbf{h}_i}\}_{i=1}^{N}$. Each hidden state is passed through a single layer ANN with $K = 120$ output values (K is the number of K-means anchors), where each logit is a binary prediction (confidence score) representing whether an event ended at that feature at a specific anchor. This produces a matrix of confidence scores $\overrightarrow{\mathbf{C}} = \{\overrightarrow{\mathbf{c}_i}\}_{i=1}^{N}$, which is of the same shape as the returned matrix from Algorithm 1.

To obtain the reverse or backwards hidden states $\overleftarrow{\mathbf{H}} = \{\overleftarrow{\mathbf{h}_i}\}_{i=1}^{N}$ and confidence scores $\overleftarrow{\mathbf{C}} = \{\overleftarrow{\mathbf{c}_i}\}_{i=1}^{N}$, the same process as above was followed with the exception of the order of the context vector. The features from the context vector T were passed into a separate LSTM in the reverse order.

4.5 Caption Generation

Segments (d) and (e) in Figure 4.1 represent the caption module proposed by Wang *et al.* (2018). The caption module takes the proposal hidden state, feature vectors, and the previous hidden state as input to a two layer LSTM. Each predicted sentence has a limit of 30 words as such there are $t = 30$ timesteps in our sequence.

For the duration of this section, we will discuss the formulation for the concatenation of these individual components.

Considering a video V we have $T = \{\mathbf{t}_1, \mathbf{t}_1, ..., \mathbf{t}_N\}$ feature vectors, videos in the ActivityNet Captions dataset do not have a fixed duration thus the number of feature vectors per video is not consistent. Furthermore an event e in a video is assigned a subset of the feature vectors in T, $T_e = \{\mathbf{t}_m, \mathbf{t}_1, ..., \mathbf{t}_n\}; m < n, m \geq 0, n \leq N$ where the length of an event is given by $l = n - m + 1$.

The length of T_e can vary between events in videos as such Wang *et al.* (2018) introduced an attention module to fuse the event feature vectors T_e and the hidden states obtained in the temporal action proposal module. The proposal forward and reverse hidden states for event e are given by $\overrightarrow{\mathbf{h}_n}$ and $\overleftarrow{\mathbf{h}_m}$ respectively. For the duration of this dissertation we refer to the concatenation of these features as $\mathbf{h}$ where $\mathbf{h} = [\overrightarrow{\mathbf{h}_n}, \overleftarrow{\mathbf{h}_m}]$. The previous hidden state of the caption module LSTM is given by $\mathbf{H}_{t-1}$.

4.5.1 Temporal Dynamic Attention

Wang *et al.* (2018) proposed a mechanism to dynamically fuse an event's feature vectors T_e. Temporal dynamic attention is the process of constructing a weighted sum of all feature vectors in T_e at each timestep t. This is done as calculating the mean of T_e does not explore the relationship between the feature vectors [Wang *et al.* 2018]. Each

feature vector is assigned a weight and is calculated using a two layer ANN:

$$\alpha_i^t = \sigma(W_a^T \cdot \tanh(W_t \mathbf{t}_{i+m-1} + W_h \mathbf{h} + W_{\mathbf{H}} \mathbf{H}_{t-1} + \mathbf{b})) \tag{4.1}$$

This constructs an explicit relationship between the proposal hidden state, feature vector, and previous hidden state of the caption LSTM at each timestep t.

The weighted sum of the feature vectors in T_e are given by:

$$\widetilde{\mathbf{T}}^t = \sum_{i=1}^{l} \alpha_i^t \cdot \mathbf{t}_{i+m-1} \tag{4.2}$$

4.5.2 Context Gating

Wang *et al.* (2018) proposed a gating mechanism inspired by LSTM gates. The proposed gating mechanism is used to model the contributions of the feature vectors and contexts when generating a new word. The first step is to project $\widetilde{\mathbf{T}}^t$ into the same space as the hidden states of the current event to obtain $\dot{\mathbf{T}}^t$.

$$\dot{\mathbf{T}}^t = \tanh(\widetilde{W}\widetilde{\mathbf{T}}^t + \mathbf{b}) \tag{4.3}$$

Let $\mathbf{E}_t$ be the word embedding of the previously predicted word in the caption module. The context gate for the feature vectors and proposal hidden state are given by:

$$g_{ctx} = \sigma(W_g[\dot{\mathbf{T}}^t, \mathbf{h}, \mathbf{E}_t, \mathbf{H}_{t-1}]) \tag{4.4}$$

The input to the caption LSTM is given by:

$$\mathbf{F}(e) = f[(1 - g_{ctx}) \odot \dot{\mathbf{T}}^t, g_{ctx} \odot \mathbf{h}] \tag{4.5}$$

The output hidden state $\mathbf{H}_t$ of the caption LSTM is used as the input to a single layer ANN to predict the next word in the sequence.

4.6 Evaluation Metrics

All of the following temporal action proposal and captioning evaluation metrics are based on Krishna *et al.* (2017).

Given a sentence, a unigram refers to a single word and an *n-gram* refers to n adjacent words in a sentence. A unigram can be considered an *n-gram* where $n = 1$.

4.6.1 BLEU

Bilingual Evaluation Understudy (BLEU) proposed by Papineni *et al.* (2002) is a common metric in natural language processing which is used to evaluate generated text. BLEU evaluates the overlap between unigrams or n-grams. The evaluation returns a

score of 1 if a generated sentence is an exact match for the ground truth sentence. The log score is calculated using the following equation:

$$\log \text{BLEU} = min(1 - \frac{l_r}{l_c}, 0) + \sum_{n=1}^{N} w_n \log p_n \tag{4.6}$$

In Equation 4.6 $\frac{l_r}{l_c}$ describes the ratio between the lengths of the reference and generated captions. Generally, the weight terms w_n are obtained by the number of adjacent words we are considering. The term p_n is referred to as the modified n-gram precision. As mentioned earlier an *n-gram* referrers to the number of adjacent words in a sentence. All possible *n-grams* in a generated sentence and a reference sentence are considered. For each unique *n-gram* in the reference sentence the number of times they appear in the reference and generated sentences respectively are counted. Divide the number of occurrences in the reference sentence by that of the occurrences in the generated sentence and calculate the geometric mean to obtain each p_n term. The first term in Equation 4.6 penalizes the descriptions if they are too short and the second term calculates the match between the sentences.

4.6.2 METEOR

Metric for Evaluation of Translation with Explicit Ordering (METEOR) proposed by Banerjee and Lavie (2005) was created as a means of overcoming certain limitations of BLEU such as context and arrangement of phrases in a proposed sentence. Instead of evaluating generated sentences for exact matches to reference sentences, semantic matching was introduced. METEOR uses WordNet which is a lexical database for English. It uses this database to account for inexact but relevant unigrams as sentences can be rephrased by means of exact word, stemmed word (same base word with different suffixes), synonym, and paraphrase matching [Aafaq *et al.* 2018]. Using WordNet it generates a mapping between all words that exist in the reference sentence to words in the generated sentence. Thereafter the alignment between common *n-grams* in the generated and reference sentences is calculated. The METEOR score is calculated in two distinct steps, first, the F-score is calculated using the following equation:

$$F_{mean} = \frac{10PR}{R + 9P} \tag{4.7}$$

Similarly to BLEU, P represents the precision score given by $P = \frac{m_{gr}}{m_{gt}}$ where m_{gr} represents the number of unigrams common between both generated and reference sentences and m_{gt} represents the total number of unigrams in the generated sentence. R represents the recall score given by $R = \frac{m_{gr}}{m_{rt}}$ where m_{rt} represents the total number of unigrams in the reference sentence.

While these metrics measure unigram similarity, *n-grams* are not accounted for. As a means of integrating common *n-gram* groupings between the reference and generated sentences into the score, a penalty score is calculated. The penalty score is influenced by adjacent *n-grams* in the reference and generated sentences. That is if we consider the

position of each word in the reference sentence, a chunk is defined as an *n-gram* that exists in both the reference and generated sentence. Given the generated and reference sentences, obtain the minimum number of chunks common between both sentences where the order of unigrams in the chunks are adjacent (found in the same order of appearance between the proposal and reference sentence) to the order of unigrams in the reference sentence. The penalty is calculated as follows:

$$p = \frac{1}{2}(\frac{N_c}{N_u})^3 \tag{4.8}$$

Where N_c is the number of chunks that exist in the reference sentence which occur in the generated sentence in the same order as the reference sentence and N_u is the number of unigrams in the reference sentence. Lastly, the METEOR score is obtained by the following:

$$M = F_{mean}(1 - p) \tag{4.9}$$

Note: If the reference sentence occurs in the generated sentence N_c is 1, but if there exists an extra word in the generated sentence which does not exist in the reference sentence then N_c is 2 as the extra word serves as a partition between two chunks of the reference sentence. The maximum number of chunks is the number of unigrams in the reference sentence. Chunks cannot change order. That is if we concatenate all chunks we will obtain the reference sentence.

4.6.3 ROUGE-L

Recall-Oriented Understudy for Gisting Evaluation - LCS (ROUGE-L) proposed by Lin (2004) is an evaluation metric for natural language processing. The evaluation metric considers considers the longest common sequence (lcs) of unigrams in a generated sentence when compared to a reference sentence. Similar to the METEOR metric the ROUGE-L score is an F-score calculated using the following equations:

$$\begin{aligned} R_{lcs}(c_i, s_i) &= \frac{LCS(c_i, s_i)}{len(s_i)} \\ P_{lcs}(c_i, s_i) &= \frac{LCS(c_i, s_i)}{len(c_i)} \\ \beta &= \frac{P_{lcs}(c_i, S_i)}{R_{lcs}(c_i, S_i)} \\ ROUGE\text{-}L(c_i, s_i) &= \frac{(1+\beta^2)R_{lcs}(c_i, s_i)P_{lcs}(c_i, s_i)}{R_{lcs}(c_i, s_i) + \beta^2 P_{lcs}(c_i, s_i)} \end{aligned} \tag{4.10}$$

Where c_i is the generated sentence and s_i is a possible reference sentence.

Note: The longest common subsequence $LCS(\cdot, \cdot)$ does not require the unigrams to be consecutive but it does require them to be in sequence. That is if the reference sentence does exist in the generated sentence, even if there are extra unigrams between words that occur consecutively in the reference sentence the $LCS(\cdot, \cdot)$ function will return the length of the reference sentence.

4.6.4 CIDEr

Consensus based Image Description Evaluation (CIDEr) proposed by Vedantam *et al.* (2015) is an evaluation metric for image captioning. The evaluation only considers the root word of each unigram. Each sentence is considered to be a set of *n-grams* between 1 and 4 words each. The frequency of *n-grams* common between both generated and reference sentences are measured to encode the consensus between both sentences. Common *n-grams* between the generated and reference sentences are given a lower weight than those occurring less frequently. The weighting of each word is given by:

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{\omega_l \in \Omega} h_l(s_{ij})} \log\left(\frac{|I|}{\sum_{I_p \in I} min(1, \sum_q h_k(s_{pq}))}\right) \tag{4.11}$$

The weighting for the generated sentence is given in the same manner.

The CIDEr score for a set of *n-grams* is given by:

$$\text{CIDEr}_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i).g^n(s_{ij})}{\parallel g^n(c_i) \parallel . \parallel g^n(s_{ij}) \parallel} \tag{4.12}$$

Where $g^n(\cdot)$ is the vector of weights formed by Equation 4.12 for *n-grams* of length n for the reference and generated sentences respectively.

The final CIDEr score which considers all sets of *n-grams* for $N = 4$ is given by:

$$\text{CIDEr}(c_i, S_i) = \sum_{n=1}^{N} \frac{1}{N} \text{CIDEr}_n(c_i, S_i) \tag{4.13}$$

Where c_i is the generated sentence and S_i is a set of possible reference sentences. $g^n(c_i), g^n(S_i)$ represents all *n-grams* with n words in the generated and reference sentences respectively and $||g^n(\cdot)||$ represents the magnitude or number of *n-grams* in the generated and reference sentences respectively.

4.6.5 Temporal Action Proposals

We used Precision@1000 and Recall@1000 (precision and recall calculated for the top 1000 proposals for each video) averaged at IoU thresholds $\{0.3, 0.5, 0.7, 0.9\}$ to compare the ground truth and proposed timestamps.

4.6.6 PCA: Reconstruction Error

PCA is the process of projecting a set of data points $D \in \mathbb{R}^{(n \times k)}$ to a lower dimensional subspace $Z^* \in \mathbb{R}^{(n \times p)}$. *See Section 2.6*. The PCA reconstruction error is a projection of Z^* back to the original subspace to form $D^* \in \mathbb{R}^{(n \times k)}$. Thereafter we the Mean Squared

Error (MSE) to compare the original set of data points to the reconstructed datapoints. We define the reconstruction error as:

$$D^* = Z^* C^\intercal + m \tag{4.14}$$

$$PCA_{re} = \frac{1}{n} \sum_{t=1}^{n} (D - D^*)^2 \tag{4.15}$$

Where m is the mean of the original set of data points D and the Mean Squared Error is used to calculate the error between the original and reconstructed points in Equation 4.15.

4.6.7 Action Classifiers

Following Tran *et al.* (2015), all action classification results are presented in the form of top-1 and top-5 accuracies. Where the top-1 accuracy is the probability of the correct prediction occurring in the largest predicted logit and the top-5 accuracy is the probability of the correct prediction occurring in the five largest predicted logits. Furthermore we present the top-10 and top-20 accuracies for further comparisons and analysis.

4.7 Training Method

4.7.1 Action Classifiers

The 6 action classifiers described in Section 4.3 were trained on the Kinetics-368 dataset. Random clips were extracted from each video during the training phase. The length of each extracted clip varied with each model, the C3D-8 and C3D-8 Attn models were trained on clips with a length of 8 frames whereas the remaining four models were trained on clips with a length of 16 frames. Each clip's spatial dimensions were randomly cropped into frames of dimension (112×112) and were randomly flipped horizontally with a probability of 50%. A batch size of 16 was used. The Stochastic Gradient Descent (SGD) optimizer was used with Nesterov momentum. The initial learning rate was set to 0.01, when the validation loss between successive epochs had a difference less than 0.001 the learning rate was stepped down with a ratio of 0.1. The training phase ended when the validation loss between successive epochs had a difference less than 0.001 and the learning rate was 0.0001.

4.7.2 Dimensionality Reduction

In Subsection 4.2.2, we described how each video in the ActivityNet Captions dataset was reduced to a set of feature vectors $T = \{\mathbf{t}_1, \mathbf{t}_1, ..., \mathbf{t}_N\}$. For the temporal proposal and captions module we only used the feature vectors.

4.7.3 Temporal Action Proposals

The durations of all timestamps were collected and clustered into $K = 120$ anchors. Each set of feature vectors had a corresponding set of ground truth vectors $\{g_t\}_{i=1}^{N}$. Each ground truth vector g_t was a K dimensional binary vector. An entry in g_t was assigned a value of 1 if the temporal Intersection-over-Union (tIoU) score between the estimated and ground truth timestamps was larger than 0.5 [Wang *et al.* 2018] as shown in Algorithm 1.

At each training iteration we calculated the ratio between the number of 0's and 1's vs the total number of elements in T for that specific batch. To calculate our proposal loss L_p we adopted the weighted binary cross-entropy loss function. We have a large imbalance between the number of 0 and 1 values in g_t, as such we used a weighted loss function to balance the negative and positive proposal predictions. Given all feature vectors T representing a video V the loss is calculated:

$$L_p(c, T, g_t) = \sum_{t=1}^{N} - \sum_{j=1}^{K} w_0^j g_t^j \log c_t^j + w_1^j (1 - g_t^j) \log(1 - c_t^j) \tag{4.16}$$

Where w_0 and w_1 are the calculated ratios of 0's and 1's and c_t^j are predictions from the temporal action proposal module. We calculated the loss for the normal and reverse features in the same manner. As the number of feature vectors may differ from video to video we averaged L_p across all timesteps:

$$L_p = \frac{L_p}{N} \tag{4.17}$$

We pretrained the Temporal Action Proposal module until the validation loss between successive epochs had a difference less than 0.001.

4.7.4 Captions

Following Wang *et al.* (2018) we only assigned a sentence to a feature vector if the tIoU score of the ground truth and estimated timestamps had a score larger than 0.8. Furthermore, while we imposed a strict size on our sentences (30 words), there filler words were used for shorter sentences. We constructed a mask as a means of removing those extra words before the loss was calculated.

Deviating from Wang *et al.* (2018) which used a cross-entropy loss function, we noticed that there is a severe imbalance in the frequency in which words appear in the dataset. As such to accommodate this imbalance we implemented a modified cross-entropy loss function, namely the Focal loss function introduced by Lin *et al.* (2017). The Focal loss function was designed to aid in predicting "difficult" results. Specifically, the loss of a prediction is weighted according to the magnitude of the predicted probability. If the correct value is predicted with a high probability, the loss contribution will be dynamically scaled down compared to a prediction with a poor probability. We implemented this loss function as some words appear frequently in a sentence

whereas others only appeared a few times in the dataset. We defined the caption loss L_p as the Focal loss of the *correct word* in a sentence with M words:

$$L_c = -\sum_{t=1}^{M} \alpha(1-p_i)^{\gamma}\log(p_i) \tag{4.18}$$

Where p_i is the probability of the i'th word in the ground truth sentence, α is the ratio of the frequency of the i'th word compared to the total number of words in the batch and $\gamma = 5$.

4.7.5 Total Loss

The total loss used during training is given by a weighted sum of the proposal and caption loss:

$$L = \lambda \cdot L_c + L_p \tag{4.19}$$

Where $\lambda = 5$ is the balancing factor.

For both the Temporal Action Proposals and Captions we used the ADAM optimizer [Kingma and Ba 2014]. Adopting the same hyperparameters as Wang *et al.* (2018), we used an initial learning rate of 0.001 and implemented weight decay with a value of 0.000001 on all layers except the word embedding layer. The Temporal Action Proposal and Caption networks were trained until the validation loss between successive epochs had a difference less than 0.001.

Using the pretrained Temporal Action Proposal module we jointly trained the Temporal Action Proposal and Caption modules. We used a pretrained Temporal Action Proposal module as we found that the module converged to a lower precision when we used an untrained Temporal Action Proposal module during joint training. To accommodate for the trained Temporal Action Proposal module and the untrained Caption module we used λ as a balancing factor when the total loss was calculated.

4.8 Inference

The dense video captioning pipeline has two decoder modules, the temporal proposal, and caption LSTM decoders. We utilized output predictions from both networks to rank proposals and their generated captions respectively. To identify if proposals and captions were accurately generated, the proposal and caption modules must have a high confidence score respectively. Following Wang *et al.* (2018), we used their method of ranking generated proposals and their respective captions:

$$C_c = \sum_{i=1}^{M} \log(p(w_i)) \tag{4.20}$$

Where C_c is the confidence score of a predicted sentence with M words ${w_i}_{i=1}^{M}$, and $p(w_i)$ is the probability of the correct logit generated by the caption decoder. Combining the caption confidence scores with the confidence scores calculated in the proposal

module, the final score is given by:

$$C = \gamma C_p + C_c \tag{4.21}$$

Where C_p is the proposal confidence score defined in Section 4.4 and γ is the balancing factor. This balancing factor is used because the proposal confidence score represents the combined value of the forward and backward confidence scores for that prediction, whereas the caption confidence score is representative of M words [Wang *et al.* 2018]. Following Wang *et al.* (2018), we set $\gamma = 10$.

This combined score was used to rank each of the proposal caption pairs, for evaluation we selected the top 1000 proposals for each video and report our results on them.

Furthermore, we implemented an extra heuristic which filtered out extra words in a sentence before the evaluation metrics were applied to the ground truth and proposal sentences during inference. We removed words from the proposed sentences if they occurred more than once in succession and keep only the first instance of the identified duplicate word in the sequence. Furthermore we identified that there was a severe imbalance in the number of occurrences of words in the entire dataset, to combat this we implemented a weight to the loss function calculated per batch during the training phase.

4.9 Conclusion

In this section, we discussed how we would prepare the Kinetics-368 and ActivityNet Captions dataset for training and inference. We discussed the action classifiers that we will use and how we integrated the attention estimator into their architectures. We discussed the temporal action proposal and captioning modules provided by Wang *et al.* (2018). We discuss the metrics that we will use when evaluating our action classifier, temporal action proposal, and captioning networks. We discuss the method we used to train our networks and how we will use them for inference.

Chapter 5

Results and Analysis

5.1 Introduction

In this chapter, we discuss the results of our experiments and the impact the various architectures and the manner in which they process videos affected the output of the temporal action proposal and caption modules. We observe the performance of the action classifiers that were implemented and discuss why scores tend to vary and anomalous results occur. We discuss the effect of applying PCA to the outputs of our action classifiers. We analyze the performance of temporal action proposal and captioning modules, and how their results varied according to the amount of information lost due to applying dimensionality reduction to the action classifier outputs. We analyze the captioning modules performance and discuss trends which occurred between caption evaluation metrics and their individual components based on how they are calculated and what they analyze. Lastly, we perform a qualitative analysis and explain the relative quantitative performance in relation to the generated captions.

5.2 Action Classifiers

Table 5.1: Architecture - Action Classifier, ACC@1 - Video Level Top-1 accuracy, ACC@5 - Video Level Top-5 accuracy, ACC@10 - Video Level Top-10 accuracy, ACC@20 - Video Level Top-20 accuracy, Temporal Depth - Number of Frames processed in a single pass and NUM_PARAMS - The Total Number of Trainable Parameters

Architecture	ACC@1	ACC@5	ACC@10	ACC@20	Temporal Depth	NUM_PARAMS
C3D-8	47.08754	73.33939	81.34439	87.85057	8	79, 496, 696
C3D-16	50.78914	74.65652	82.03702	87.69161	16	79, 493, 696
C3D-8 Attn	56.62541	80.40195	86.93085	92.23345	8	29, 573, 164
C3D-16 Attn	59.65709	82.40036	88.40695	92.98285	16	29, 575, 724
3D MobileNetV3	**59.70251**	**82.95674**	**89.00874**	93.44839	16	4, 996, 104
3D MobileNetV3 Attn	57.06824	82.15056	88.72488	**93.63007**	16	3, 705, 544

Table 5.1, shows the top-1, top-5, top-10, and top-20 video level accuracies for each of our implemented architectures. The top-10 and top-20 accuracies are not a part of our research questions but are used for further analysis. The temporal depth (δ) is the number of frames processed by the architecture in a single pass. The NUM_PARAMS is the total number of trainable parameters for each architecture.

It should be noted that the C3D-8 and C3D-16 architectures have the exact same layers and configurations, the only difference between them is the shape of the last convolutional layer's max-pooling kernel shape.

Consider a video V with L frames, we split V into $N_v = \frac{L}{\delta}$ clips. In our test step, we passed each clip into our action classifier, thereafter we identified the top-5 modal predictions for all clips in V. The video level accuracy was calculated using these modal predictions.

We note that Hara *et al.* (2017) found that the C3D architecture was too shallow to fit correctly onto the Kinetics-400 dataset. We found similar results when considering our attention augmentations and the implementation of 3D MobileNetV3 which has fewer parameters but is a deeper architecture with a more refined design compared to the base C3D architectures.

5.2.1 Temporal Depth Effect on Results

For this subsection, we consider the four C3D architecture results from Table 5.1. We consider the C3D-8 and the C3D-8 Attn models as baseline models used for comparison to their counterparts which process 16 consecutive frames in a single pass.

Observing our top-1 accuracies on the C3D and C3D attention architectures, we found an increase of approximately 3% when utilizing a temporal depth of 16 as opposed to 8. This occurs because a clip is a short set of consecutive frames from a video, very little movement occurs on a frame by frame basis. Therefore by incorporating more frames into the input, these changes in movement become more apparent. This is especially prominent when considering the C3D-8 and C3D-16 architectures as they are identical but incorporating more frames into the input resulted in better accuracies.

Observing the top-5 accuracy on the C3D attention architectures we found that there was a greater increase in accuracy compared to the non-attention augmented C3D architectures. This occurred due to the temporal depth being a factor in each attention estimator. The input to the attention estimator is dependent on what layer it extracts information from; at the earlier layers, the temporal depth is larger, but at later layers the temporal depth has been decreased due to either pooling layers or an increased stride. Therefore attention estimators at the earlier layers capture more information from the temporal depth whereas attention estimators at the later layers capture information from the spatial context after the temporal depth has decreased.

Observing the top-10 and top-20 accuracies we found that the trends that occurred in the top-1 and top-5 accuracies were not as apparent. The small difference between the top-10 and top-20 accuracies across the implemented architectures indicate that the

networks which had a lower accuracy in the top-1 and top-5 scores identified features of interest but were not able to distinguish the class of interest.

There is no 8 frame implementation of the 3D MobileNetV3 architecture as the temporal depth is decreased to one in a very early layer in the forward pass, we experienced very poor performance with this model and as such have excluded the results.

5.2.2 Attention Estimator Effect on the Networks and their Results

The attention estimator described in section 4.3, was implemented in 3 of our base configurations namely the C3D-8, C3D-16, and the 3D MobileNetV3 architectures. We found increases in the overall accuracies of the C3D augmented architectures. As explained above this is attributed to obtaining information from differing spatial and temporal resolutions from attention estimators at different layers in the network.

After incorporating the attention estimator module into the 3D MobileNetV3 (3D MobileNetV3 Attn) architecture we found a decrease in performance. In section 3.6, we explained that the MobileNetV3 and subsequently the 3D MobileNetV3 architectures use linear bottleneck blocks whereby channels outside a block are kept to a minimum and are expanded by an expansion layer at the beginning of the linear bottleneck block. The Attention Estimators in the 3D MobileNetV3 Attn architecture obtain their input after a linear bottleneck block, therefore after a projection layer has reduced the number of input channels. The linear bottleneck layers constrain the amount of information that the Attention Estimator receive, and subsequently, the usage of Attention Estimators has resulted in a lower accuracy compared to the base 3D MobileNetV3 architecture.

The architecture diagrams are presented in Appendix A. In the the 3D MobileNetV3 attention architecture the output of the 3 attention estimators are vectors of dimensions $40, 112$, and 960 respectively. The information obtained from the earlier layers is constrained due to the low number of channel outputs from the linear bottleneck blocks, whereas most of the information is obtained from the final linear bottleneck block which has significantly more channels. For the C3D architectures, the dimensions of the attention estimator outputs are $256, 512$ and 512 which balance the contribution of information from different spatial and temporal resolutions, as the C3D architectures only use traditional 3D convolutional layers and not linear bottleneck blocks. This indicates that the performance of the attention estimators in classification is dependant on the balance between the individual contributions of information from each estimator. We also see that Attention Estimators for the 3D MobileNetV3 Attn architecture have a small number of channels as input for the first two estimators while the C3D Attn architectures have a larger number of channels, indicating that the information that was filtered out in the earlier convolutional layers did contribute to the classification performance of the attention augmented architectures.

Reduction in Number of Parameters

Due to the Attention Estimators utilizing the fully connected layers which were rescaled according to the dimensionality of the concatenated attention features, we achieved a

significant decrease in the number of parameters, especially in the C3D architectures. 3D MobileNetV3 was designed with the intention of maintaining a minimal number of trainable parameters. When the Attention Estimators were integrated into the network, the final expansion layer before the classification layer was not used. This resulted in a reduction in the number of trainable parameters and thus a reduction in network capacity. This is seen in the decrease in performance between the 3D MobileNetV3 architecture and its attention augmented counterpart.

5.3 Dense Video Captioning

5.3.1 Impact of Dataset Choice

We acknowledge that our results are poorer than the baseline scores obtained by Wang *et al.* (2018) as seen in Figures 5.1, 5.2 and Table 5.3. Wang *et al.* (2018) used a C3D architecture which was trained on the Sports1M dataset with a temporal depth of 16. We used the Kinetics-368 dataset as it does not only contain sports action classes but it also contains more general action classes that involve everyday actions such as ”painting nails” among others including sporting classes such as ”throwing a football”. When comparing the two datasets the Sports1M dataset has 487 action classes with approximately 2500 videos per class, whereas the Kinetics-368 has 368 action classes with approximately 600 videos per action class. The Kinetics-368 dataset was used as a means of maximizing the number of videos per action class but reducing the number of classes due to limitations in computational resources available to us specifically the time taken to train on larger datasets took too long as we implemented six action classifiers. Furthermore, we chose a subset of the Kinetics-600 dataset as the classes are more general and not solely focused on sporting actions, this was done as a means to test if caption performance will improve if we utilized action classifiers trained on a dataset with a more general set of action classes.

We found that we obtained poorer results than the C3D architecture trained on the Sports1M dataset given by Tran *et al.* (2015).Their architecture achieves a video level top-1 accuracy of 60% and a top-5 accuracy of 84.4%. These scores are similar to the ones we obtained on our action classifiers over the Kinetics-368 dataset as seen in Table 1.1. The network architectures that we implemented were trained on the Kinetics-368 dataset which is approximately six times smaller than the Sports1M dataset and has 119 fewer classes. This resulted in the networks being exposed to far fewer features than the C3D architecture trained on the Sports1M dataset. Rather than resulting in better features being generated due to the generality of the Kinetics-368 action classes, we were hindered by the smaller size of our dataset. Therefore when extracting features on the ActivityNet Captions dataset using our action classifiers we obtained poorer results. This problem could be solved by training on the Kinetics-600 or even the newer and larger Kinetics-700 dataset [Carreira *et al.* 2019] which has far more classes than the Sports1M dataset even if there are fewer videos per class.

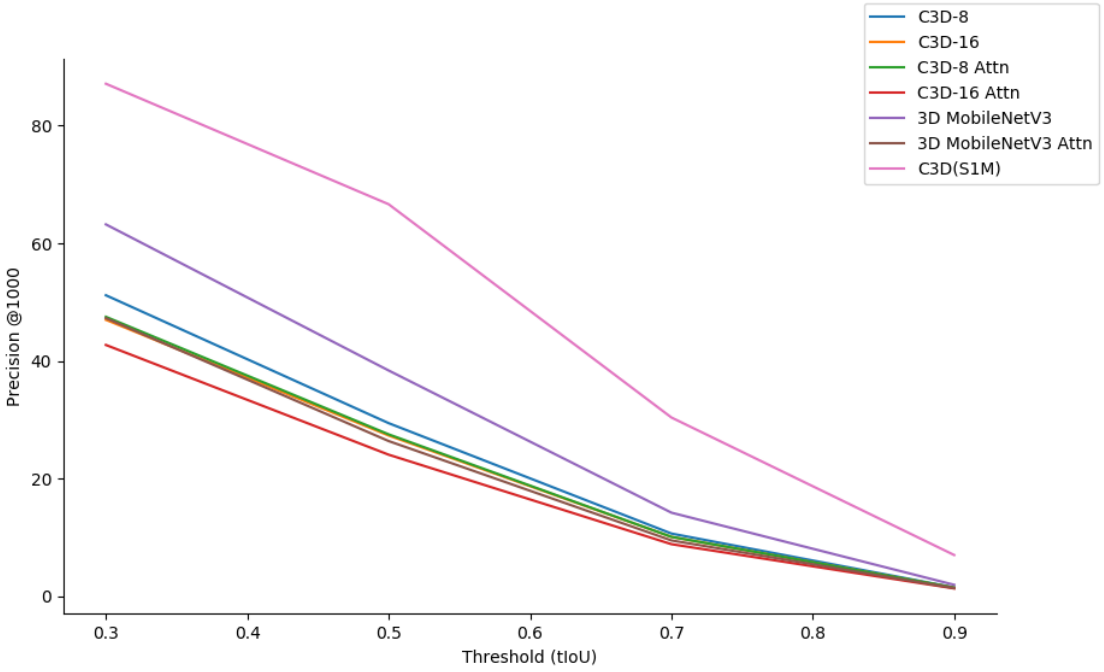


Figure 5.1: Precision@1000 vs tIoU

5.3.2 Temporal Action Proposals

When calculating the temporal proposal precision, recall, and caption metrics, we selected the top 1000 proposal, caption pairs for each video where each pair was ranked according to their confidence score C given by Equation 4.21.

As defined in the evaluation metrics given by Krishna *et al.* (2017), the recall of our temporal proposals is the percentage of ground truth timestamps which can be represented by proposals that have an IoU score larger than a specified threshold when compared to the ground truth timestamps. The precision of our temporal proposals is a percentage of the total proposals which have an IoU score larger than a specified threshold when compared to the ground truth timestamps.

Observing Figure 5.1, we found that the 3D MobileNetV3 features yielded the most amount of valid proposals at each IoU threshold. This is an interesting result as the 3D MobileNetV3 architecture has the largest reconstruction error as seen in Table 5.2. Considering a sequence of events in a video, different events in a video have significant differences in their frames, whereas frames within an event will have minor changes on a frame-by-frame basis.

Observing our 3D MobileNetV3 classification results in Table 5.1, it yielded the best results compared to our other classifiers. But the reconstruction error is also the largest indicating a large loss of information compared to the other features after applying PCA. Consequently, this resulted in minor changes or fine details between successive frames being more difficult identify as information was lost. Classifier features that had a lower reconstruction error retained more of the information compared to the 3D

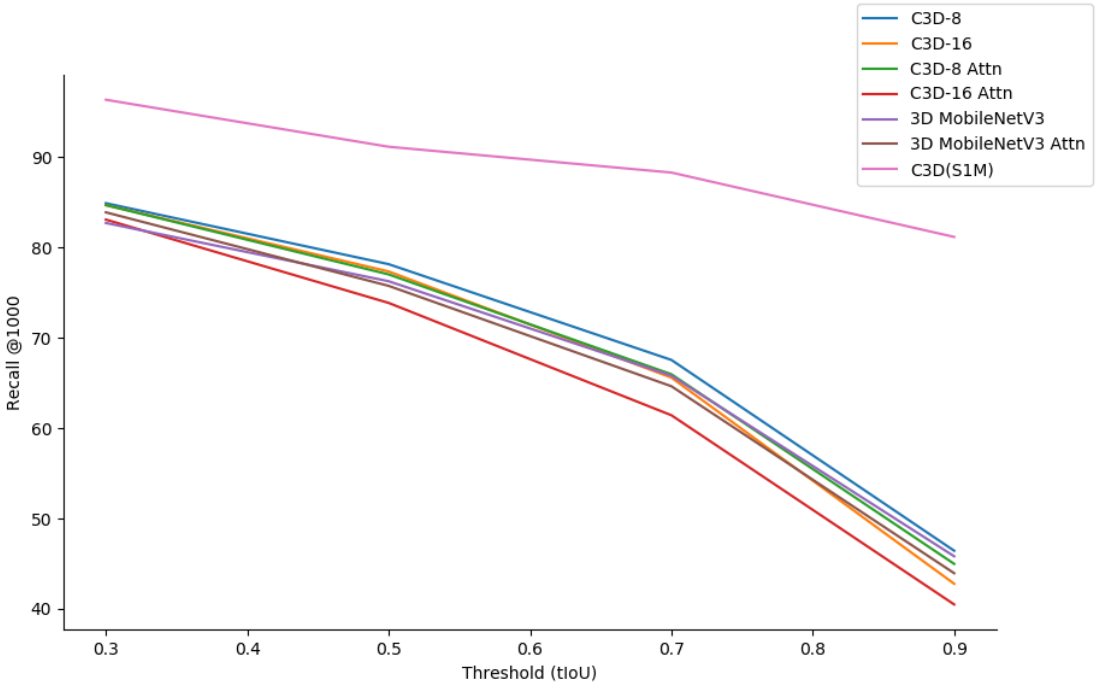


Figure 5.2: Recall@1000 vs tIoU

MobileNetV3 architecture. Consequently, they had representations for more minute changes between successive frames which led to a larger number of false positive temporal predictions, whereas the 3D MobileNetV3 architecture lost such information which resulted in a reduction of false positive predictions.

This is seen to a lesser degree in the C3D-8 architecture features, which achieved a slightly better precision than the other architecture features aside from the 3D MobileNetV3 architecture features. The classifier performed poorly and as a result, it did not represent the small changes between frames well but can identify the larger changes between different events.

Architecture	iPCA Reconstruction Error
C3D-8	0.00638
C3D-16	0.01455
C3D-8 Attn	0.00225
C3D-16 Attn	0.01166
3D MobileNetV3	0.67547
3D MobileNetV3 Attn	0.00602

Table 5.2: Architecture - Action Classifier, iPCA Reconstruction Error - The Mean squared error of the ActivityNet Captions features after transforming the data back to the original subspace.

Considering the non-attention augmented architectures, we obtained the features from a fully connected layer where the input to the layer was projected to a subspace. Ob-

serving the architecture diagrams for the attention augmented architectures, we obtained the feature vectors from a concatenation of attention vectors $g_{i,a}$ from three individual attention estimator modules. Observing the scores for the features obtained using the attention estimators, specifically the C3D-8 Attn, C3D-16 Attn, and the 3D MobileNetV3 Attn architectures we found that they resulted in poorer precision and recall scores than their non-attention augmented counterparts. While their reconstruction errors were low, we found that the features obtained from the concatenation of the three attention estimator vectors resulted in lower scores. We may find better results if we add another fully connected layer to project the concatenated vector to a subspace and thereafter use those features for the temporal proposal and caption modules.

Observing the C3D-8 and C3D-8 Attn feature results in figures 5.1 and 5.2, we found that the 8 frame action classifiers outperformed their 16 frame counterparts. In Subsection 4.2.2 we explained that each feature vector passed to the temporal action proposal module and the captioning module represents 64 consecutive frames in a video. In the 8 frame action classifiers we constructed these 64 frame representations using 8 individual feature vectors extracted from the action classifier, whereas in their 16 frame action classifier counterparts we only used 4 feature vectors to construct the 64 frame feature vector. Therefore, we see that at smaller strides we have better temporal consistency within each 64 frame feature vector representation.

5.3.3 Captions

In this section we discuss the caption results that we obtained and why certain achitectures outperformed others. The caption results can be found below in Table 5.3:

Table 5.3: Caption Results on the ActivityNet Captions Dataset. All Scores are Presented as Percentage Values.

Feature_Architecture	BLEU-1	BLEU-2	BLEU-3	BLEU-4	Meteor	Rouge-L	CIDEr
C3D (S1M) [Wang *et al.* 2018]	18.99	8.84	4.41	2.30	9.60	19.10	12.68
C3D-8	*11.6442*	*2.5937*	0.2749	0.0316	5.8599	*13.5622*	*1.9133*
C3D-16	12.5589	**2.8752**	**0.3259**	**0.0322**	**6.1645**	14.3995	2.4833
C3D-8 Attn	12.1530	2.7985	0.2758	**0.0322**	5.9732	14.0412	2.2712
C3D-16 Attn	12.2863	2.7935	0.2798	*0.0252*	5.9976	14.1880	2.5116
3D MobileNetV3	**14.6587**	2.7912	*0.1582*	*0.0124*	*5.7459*	**15.5055**	**3.6011**
3D MobileNetV3 Attn	12.1915	2.8060	0.2830	0.0246	6.0250	14.0861	2.3410

Action Classifier Performance and Reconstruction Error

Observing the reconstruction errors in Table 5.2, we found that the C3D-8 and C3D-8 Attn architectures obtained reconstruction errors approximately 5 times less than that of their 16 frame counterpart architectures. This indicates that feature vectors representing more frames did not lie on a linear subspace after applying PCA to the original feature vectors.

In Table 5.1, we found that the C3D-8 Attn, C3D-16 Attn and 3D MobileNetV3 Attn architectures achieved far better results on the Kineics-368 dataset than the C3D-16 architecture, but in Table 5.3 see that the features extracted from the ActivityNet Captions dataset using the C3D attention augmented architectures resulted in poorer scores than the features extracted using the C3D-16 architecture. In the previous section we explained that the manner in which we extracted feature vectors from the attention augmented architectures resulted in lower temporal proposal scores than their non-attention augmented counterparts, this result holds to some degree for our caption scores. We found that the C3D-8 Attn features outperform the C3D-8 features in every metric but this can be attributed to the large difference in classification performance. Furthermore this indicates that even with the manner in which features were extracted from the attention augmented architectures, the feature vectors still to represent the video features to a limited degree.

Observing the scores that the C3D-16 features obtained in Table 5.3, we see that while the C3D-16 classification performance was poorer than that of the attention augmented architectures for C3D and 3D MobileNetV3, the features obtained higher scores across all metrics except for the CIDEr score that the C3D-16 Attn features. Considering the 3D MobileNetV3 feature scores and the substantially larger reconstruction error of its features, we found that it outperformed all other features on certain metrics such as the BLEU-1, ROUGE-L and CIDEr metrics. The reconstruction error is an average score for the error in all our features, but this does not imply that every feature had that same error. The error for each feature can vary; features with a large reconstruction error lost some information, which reduces how well they describe the frames which they represent. These features are assigned a lower weight in the Temporal Dynamic Attention module as they describe less of the event than features with a lower reconstruction error which still maintain a large amount of information from the original feature vectors before applying PCA. Therefore, this indicates that even with a substantially larger reconstruction error, the performance of the classifier is still indicative of captioning performance.

Caption Metric Evaluation

Observing the results for the action classifier features that we trained in Table 5.3, the BLEU-n metrics represent the percentage of *n-grams* in the proposal sentences that occur ground truth sentence. While observing the BLEU-n scores we found that the caption results obtained using the 3D MobileNetV3 network obtained the best results with approximately 14.66% of the predicted words occurring in the ground truth sentence. Considering larger *n-grams* scores we found that the 3D MobileNetV3 features were not able to identify longer consecutive words as opposed to features obtained from our other architectures. But regardless of which set of features we consider, we do see a large dropoff when considering the baseline results from Wang *et al.* (2018), this indicates that across all sets of features we struggled to generate the same unigrams in our proposal sentences, and furthermore we struggled to generate longer *n-gram* matches.

The Rouge-L score is a measurement that represents the longest common in-sequence

unigrams that are in the proposal and ground truth sentences. *Note: The unigrams do not have to be consecutive but the unigrams that occur in the ground truth must occur in the same order in the proposal sentence.* From the BLEU-1 scores, when compared to the ground truth our predicted sentences did not contain a large portion of the ground truth unigrams, and furthermore when observing the BLEU-3 and BLEU-4 scores our models struggled to identify longer consecutive sequences of words, therefore there exist many words between unigrams in the ground truth sentence. While the features obtained using the 3D MobileNetV3 architecture returned the best score when considering unigrams and the C3D-16 features returned the best higher order *n-gram* scores, it is apparent that regardless of the architecture used to extract the features none were able to sufficiently generate consecutive unigrams that occurred in the ground truth sentences.

The CIDEr score is where we experience a large difference when comparing our scores to the baseline obtained in the Wang *et al.* (2018). The CIDEr score is a measurement of how often common *n-grams* occur between the ground truth and proposal sentences (common is not a reference to how often *n-grams* occur in a single sentence but rather indicating that they exist in both ground truth and proposal sentences). Lower weights are assigned to frequently occurring *n-grams* when calculating the score as they yield the least contextual information. Considering that the CIDEr score is an average calculated based on individual unigram, *2-gram*, *3-gram*, and *4-gram* scores it is directly related to the BLEU-n metrics. The low CIDEr scores indicate that we had a large number of words in our proposal sentences that did not exist in the reference sentences, this can be seen especially well in the BLEU-3 and BLEU-4 scores as no set of features could successfully generate consecutive *n-grams* which occur in the ground truth sentences. Furthermore, the BLEU-1 score is not an indication of uniqueness, it simply indicates that words that are in the proposal sentence are also in the ground truth. Therefore scores such as the BLEU-1 score can be considered high even if the same word is predicted many times. The low CIDEr score indicates that we had a large amount of frequently occurring words in our proposal sentences such as pronouns or articles but words that do not occur frequently such as nouns were not always present.

The METEOR score is a weighted F-score specifically an F-3 score whereby we find a weighted representation of our precision and recall between a ground truth and proposal sentence. The precision and recall are calculated by finding occurrences between common individual unigrams between the ground truth and proposal sentences. The F-score is weighted using a penalty term, this penalty term is higher if the word order is not maintained, specifically if we have a ground truth sentence and it occurs in the proposal sentence but has words that don't occur in the ground truth, the number of chunks N_c is larger and therefore the penalty for our unigram matches is higher as the context has been lost due to extra words. Considering the Rouge-L, CIDEr and BLEU-n scores we found that there was a large discrepancy in the common words between the proposal and ground truth sentences. The maximum penalty in the METEOR metric is 0.5, if we assume that N_c is the same as the number of unigrams in the ground truth sentence and disregard this we found that our METEOR scores without a penalty achieve values of approximately 12%, this shows us that there was a very small overlap between the proposal and ground truth unigrams. This analysis is com-

mon between all features extracted from all action classifiers that we implemented and correlates directly with the other scores that we obtained.

5.4 Summary and Answers to Research Questions

In this section, we briefly summarize our analysis in the form of directly answering our research questions.

1. By how much does the top-1 and top-5 accuracy differ between 3D MobileNetV3 and C3D on the Kinetics-368 dataset?
    - We found the 3D MobileNetV3 architecture outperformed the C3D architecture when trained using a temporal depth of both 8 and 16 on the Kinetics-368 dataset by a significant margin. The 3D MobileNetV3 model achieved a top-1 and top-5 accuracy of 12.62% and 9.58% higher than the C3D-8 model respectively. It also achieved a top-1 and top-5 accuracy of 8.02% and 8.2% higher than the C3D-16 model respectively.
2. By how much does the top-1 and top-5 accuracy differ between C3D and 3D MobilenetV3 augmented with a 3D adaptation of the attention module from Jetley *et al.* (2018) and their base configurations on the Kinetics-368 dataset?
    - We found that the C3D-8 and C3D-16 architectures benefited greatly when augmented with the attention estimator but the 3D MobileNetV3 architecture achieved lower accuracies when augmented with the attention estimator.
    - The C3D-8 architecture achieved a top-1 and top-5 accuracy increase of 9.54% and 7.17% respectively when augmented with the 3D attention estimator.
    - The C3D-16 architecture achieved a top-1 and top-5 accuracy increase of 8.87% and 7.74% respectively when augmented with the 3D attention estimator.
    - The 3D MobileNetV3 architecture achieved a top-1 and top-5 accuracy decrease of 2.63% and 0.81% respectively when augmented with the 3D attention estimator. This anomalous behavior is attributed to the structure of the linear bottleneck layers in the 3D MobileNetV3 architecture.
3. By how much do the CIDEr, METEOR, BLEU, and ROUGE-L scores differ when features extracted from C3D and 3D MobileNetV3 pre-trained on the Kinetics-368 dataset are used to train the Bi-Directional SST temporal action proposal module and the subsequent caption module from Wang *et al.* (2018) compared to the available features obtained from the C3D network pre-trained on the Sports1M dataset?
    - Across all generated features for all implemented action classifiers we found a net loss for all evaluation metric scores when compared to the baseline scores in Wang *et al.* (2018). This is attributed to the scale and variability

of data within each class as the Kinetics-368 dataset is significantly smaller than the Sports1M dataset.

4. To what degree are the CIDEr, METEOR, BLEU, and ROUGE-L scores affected by using features from the C3D and 3D MobileNetV3 architectures when augmented with the 3D adaptation of the attention module from Jetley *et al.* (2018)?

    - Features generated from attention augmented classifiers did not necessarily result in better classification scores, with the only exception being the C3D-8 and the C3D-8 Attn features, the C3D-8 Attn features outperformed the C3D-8 features in every metric. The 3D MobileNetV3 features outperformed the features obtained from its attention augmented counterpart in some metrics where it achieved scores higher than our other models but the large reconstruction error indicated that the features had a large loss of information and therefore the 3D MobileNetV3 Attn features obtained better scores for longer consecutive words in the generated sentences.

5. To what extent are the temporal action proposal windows improved on ActivityNet Captions dataset by utilizing features obtained from 3D MobileNetV3, C3D and their attention augmented architectures when pre-trained on the Kinetics-368 dataset compared to the results obtained by Wang *et al.* (2018) which were trained on available features obtained from the C3D network pre-trained on the Sports1M dataset?

    - Similarly to the caption results when compared to the baseline results from Wang *et al.* (2018), we found that we achieved a decrease in temporal proposal performance. This result is consistent across all features generated by the action classifiers which we trained.

5.5 Qualitative Analysis

In this section, we analyze ground truth sentences with our proposal sentences. A qualitative analysis is important when considering performance, while it may not be something that can be quantified as easily by mathematical evaluations, it helps us understand if the proposed sentences are random or if they are generating different words for an object or scene which are not quantified in the evaluation metrics.

Following Krishna *et al.* (2017) and Wang *et al.* (2018), all proposal sentences are selected where they have the highest IoU score with the ground truth. It is important to note that the overall sentence structure is fairly poor and this will not be considered when analyzing the proposal captions. The ground truth captions with the start and end frames of each event for a man mowing the lawn can be seen in Figure 5.3.

Considering the captions in Table 5.4, we found that in the first sentence, the subject is correctly identified as ”a man” but the rest of the sentence misrepresents the scene. The second sentence does not represent the actual scene or gender of the subject. In the third sentence however, the action ”turns” is correctly identified but the incorrect gender of the subject and scene are still maintained.




(a) Event 1: A man is outside in his front yard.




(b) Event 2: He is using a push mower to mow the grass.




(c) Event 3: He lifts the mower and turns it at every corner.

Figure 5.3: A video of a man mowing the lawn. Each pair of images represent the start and end frame for each event in the video. The ground truth caption for each event is provided in the caption below each image pair.

The C3D-16 features provided the best scores in most of our metrics, observing the captions for the features in Table 5.5 we find that in the first sentence the subject is correctly identified. The mower is also identified by the generic term "equipment". The second sentence retained the correct subject but the actions taken in the event are incorrect. In the third sentence, we find that the subject is correctly retained and the mower is still identified as "equipment" but the actions themselves are not generated.

In Table 5.6 we have the caption results for the C3D-8 Attn features. In sentence one, the subject is incorrectly identified as "a person" and the mower is correctly identified, but it is identified early just as it was in the C3D-16 caption results. The second

Table 5.4: C3D-8 Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	a man is down on a who shaving and shaving in the camera and it the way the
He is using a push mower to mow the grass.	a lady she on the on the camera is to her to the camera while the camera to the camera while the
He lifts the mower and turns it at every corner.	she the girls then playing, face with while, turns to the camera speaking it camera and it the it and it

Table 5.5: C3D-16 Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	a man is up the equipment quickly and he down in the pole and the pole and the and
He is using a push mower to mow the grass.	a man is and again the camera again to the to again to the to
He lifts the mower and turns it at every corner.	a man is up over the equipment at the and

Table 5.6: C3D-8 Attn Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	a person is sliding in a machine mower on around a inside by the rough along and the person in the room by the front room the
He is using a push mower to mow the grass.	a man is seen to a onto a onto a the set and the young passes the passes and the
He lifts the mower and turns it at every corner.	there lassos the mower from the mower in the water of the water of water in the and the

sentence corrects the gender of the subject. The action of mowing the grass is referred to by the action ”passes” which is incorrect but does maintain some meaning as it is a repetitive task that involves little change. The third sentence correctly identifies the object but no action or subject is correctly identified. ”Water” is identified and this is due to the water in the street gutter which can be seen in the last frame of the third event in Figure 5.3.

Considering the caption results for the C3D-16 Attn features in Table 5.7, we find that the subject is misidentified as ”a person”. The actions, objects, and subjects are all

Table 5.7: C3D-16 Attn Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	a person a ball with the man and the ice is shown it the ice in the in
He is using a push mower to mow the grass.	they are across the ice of ice to ice of ice and a ice the ice
He lifts the mower and turns it at every corner.	are up sun screen and the ice is it the

misidentified in the second and third sentences. These results are expected when considering that the features extracted using the C3D-16 Attn architecture obtained the poorest scores as seen in Table 5.3.

Table 5.8: 3D MobileNetV3 Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	a man is seated a in a room with a and his a hands and the and hands
He is using a push mower to mow the grass.	a woman is down the woman and begins to and door down to the door the door of the end of the end and to the
He lifts the mower and turns it at every corner.	the man then at the floor and another and a on a machine and it the board and the head the of and

In Table 5.8, we have the caption results for the 3D MobileNetV3 Features. The first sentence correctly identifies the subject but the scene is misidentified. The second sentence misidentifies the subject's gender and the scene. The third sentence correctly identifies the gender again, the mower is identified as a "machine" but the actions are not correctly identified. The poor representation of the overall events and scenes in the video are attributed to the large loss of information due to applying PCA to the 3D MobileNetV3 features as seen in Table 5.2.

In Table 5.9, we have the proposal captions for the 3D MobileNetV3 architecture. In the first sentence, subject is correctly identified but the scene is incorrect. In the second sentence the subject, action, and scene are correctly represented but the mower is not identified, but the ground truth represented the scene in the first sentence whereas our model could not. In the third sentence, the turning action is correctly identified and the gender of the subject is maintained.

Table 5.9: 3D MobileNetV3 Attn Proposal sentence results

Ground Truth	Proposal
A man is outside in his front yard.	the man in his wearing a end and a driveway up balancing a the side on the window and the of the
He is using a push mower to mow the grass.	the man pushes the around and turns to a yard while is the with the edge and the water on the water on the water on the
He lifts the mower and turns it at every corner.	a person pushes his all around turns and the turns around the it around the, a turns, the, with the with the

General Qualitative Result Observations

While the captions may vary there are some commonalities to observe. The most common issue is sentence structure, particularly all of our models struggle to end the sentences correctly, the most common endings are represented by articles (”the”) or conjunctions (”and”), this is attributed to how common these words occur in the dataset. There is a large class imbalance and while we make use of a weighted loss function during training it did not completely resolve this problem. A second commonality between all of our generated captions is that the subject is the easiest to identify but the object and scenes are not represented in all cases, if some object is identified it may not be referenced by the correct name, in a few cases the ”mower” was referred to as ”equipment” which is not strictly incorrect but it does group the mower under a generic term.

5.6 Conclusion

In this chapter, we found that action classifiers with a larger temporal depth resulted in better classification results. We found that the attention estimator module which we adapted for our work did result in better classification performance on the C3D-8 and C3D-16 architectures but the 3D MobileNetV3 architecture did not. This is a result of the manner in which the linear bottleneck layers expand the number of channels internally while producing a minimal subset as an output. This resulted in less information being available to the attention estimator module and therefore the performance was hindered by its use.

We found that features obtained from networks pre-trained on the Kinetics-368 dataset did not result in better temporal and caption scores than the features obtained from the C3D network trained on the Sports1M dataset. This is a result of the Kinetics-368 dataset having fewer classes and fewer videos per class. We found that features from better performing action classifiers do not necessarily result in better temporal action

proposals or better captions, there are a variety of factors involved such as the information lost when applying dimensionality reduction to the feature vectors, the temporal depth of the action classifiers and the manner of extracting the feature vectors from the action classifiers. We found that better temporal action proposals are not indicative of better captions in our case features with better temporal action proposals generally had a loss of information from dimensionality reduction or were obtained from classifiers that did not achieve the best classification results on our dataset. Lastly, we found that the manner in which we extracted information from our attention augmented architectures resulted in poor temporal action proposal and captioning performance, we consider that the features themselves should not be directly extracted from the concatenation of 3 separate attention vectors but rather the concatenation itself should be projected into a new subspace using an additional fully connected layer.

Chapter 6

Conclusion

6.1 Discussion

In this dissertation, we set out to identify the impact that features from action classifiers had on temporal action proposals and caption generation for dense video captioning. To this end, we sought to identify if we could improve the temporal action proposal and caption scores generated by Wang *et al.* (2018). The features used by Wang *et al.* (2018) were generated by the C3D network [Tran *et al.* 2015] which was pre-trained on the Sports1M dataset [Karpathy *et al.* 2014]. To identify the impact these features had on the temporal proposal and caption scores we constructed a new dataset which is formed from a subset of the Kinetics-600 dataset [Carreira *et al.* 2018]. Due to the limited amount of computational resources available to us at the onset of this project we elected to construct a subset of this dataset using the 368 common classes between the Kinetics-600 and Kinetics-400 [Kay *et al.* 2017] datasets. We did this instead of using the Kinetics-400 dataset to maximize the amount of data available to each class as each of our 368 classes has approximately 600 videos per action class, whereas the Kinetics-400 dataset has only 400 videos per action class. We construct this dataset as the Kinetics dataset family is comprised of not only sporting activities but also more general tasks, whereas the Sports1M datatset only contains sporting action classes.

We reconstructed the C3D architecture, furthermore, we constructed a 3D version of the MobileNetV3 architecture [Howard *et al.* 2019] as a means to identify if a resource efficient architecture with careful consideration to its construction would result in better classification accuracies. As a final addition to our contribution we constructed a 3D version of the attention estimator module proposed by Jetley *et al.* (2018), we augmented the C3D and 3D MobileNetV3 architectures with the attention estimator modules and evaluated its effect on the classification performance on the Kinetics-368 dataset. Furthermore, we evaluated the C3D classification performance when trained using different temporal depths specifically 8 and 16. The MobileNetV3 architectures were only designed to be trained using a temporal depth of 16.

We found that the C3D architecture trained using a temporal depth of 16 resulted in better classification performance, furthermore, we found that the 3D MobileNetV3

architecture achieved the best classification results compared to all classifiers implemented by us. We found that with the inclusion of the attention estimator modules that the C3D architectures achieved an improvement to their classification scores but the MobileNetV3 architecture did not. Specifically, due to the design of the 3D attention module, the number of input channels is directly related to how much information the attention estimator will provide. One of the design characteristics introduced in the MobileNetV2 architecture [Sandler *et al.* 2018] was constraining the number of channels that occur outside the linear attention modules. This resulted in a loss in performance when augmenting the MobileNetV3 architecture.

While evaluating the temporal action proposal and caption results from features obtained using our action classifiers on the ActivityNet Captions Dataset [Krishna *et al.* 2017], we found that while classification performance is indicative of temporal action proposal and caption performance, it needs to be coupled with an understanding of the manner in which the features are generated. We found that when applying PCA to our data, the features obtained using the MobileNetV3 architecture had a large reconstruction error and thus they had a large loss of information. When extracting features using our attention augmented architectures we found that simply using the concatenation of 3 attention vectors resulted in poor results. Furthermore, we found that temporal action proposals achieved better results on the C3D architecture and its attention augmented version when using a temporal depth of 8 rather than 16 even with the architectures trained on a temporal depth of 16 having better classification results. Lastly, we found that due to the scale of the Kinetics-368 dataset which had fewer action classes and fewer videos per class than that of the Sports1M dataset, our temporal action proposal scores and caption results were all lower than those achieved by Wang *et al.* (2018).

6.2 Future Work

In this dissertation, we found that the size of our dataset had a large influence on the final result as such future work can involve training our classifiers on the full Kinetics-600 or even the new Kinetics-700 [Carreira *et al.* 2019] datasets to identify how much of an impact the number of classes and videos per class had on the final result. We also found that the features obtained from the MobileNetV3 architecture did not form a linear subspace after applying PCA to the features, as such PCA was not the ideal dimensionality reduction technique for those features. A test of non-linear dimensionality reduction techniques can be used on the features to further identify how much temporal action proposal and captioning performance was hindered by the loss of information. We found better temporal consistency between features obtained using classifiers with a temporal depth of 8 frames. In our data preparation step on the ActivityNet Captions dataset, we generated features using an action classifier with temporal depth δ and traversed the video with a stride of δ until we have a representation of all the frames in a video. To try and enforce temporal consistency one could try and generate features using an action classifier with temporal depth δ but with a stride of $\frac{\delta}{2}$ to enforce a temporal link between successively generated features.

Appendix A

Appendix

This chapter will contain additional resources to expand and aid in understanding our work.

A.1 Classifier Architectures

	Input	Layer Type	#out	NL	s
	3 x 8 x 112 x 112	Conv3D	64	RL	1
	64 x 8 x 112 x 112	MaxPool3D	64	-	2
	64 x 4 x 56 x 56	Conv3D	128	RL	1
	128 x 4 x 56 x 56	MaxPool3D	128	-	2
	128 x 2 x 28 x 28	Conv3D	256	RL	1
	256 x 2 x 28 x 28	Conv3D	256	RL	1
	256 x 2 x 28 x 28	MaxPool3D	256	-	2
	256 x 1 x 14 x 14	Conv3D	512	RL	1
	512 x 1 x 14 x 14	Conv3D	512	RL	1
	512 x 1 x 14 x 14	MaxPool3D	512	-	1,2,2
	512 x 1 x 7 x 7	Conv3D	512	RL	1
	512 x 1 x 7 x 7	Conv3D	512	RL	1
	512 x 1 x 7 x 7	MaxPool3D	512	-	1,2,2
	8192 x 1	Linear	4096	RL	1
FV:	4096 x 1	Linear	4096	RL	1
	4096 x 1	Linear	368	-	1

Table A.1: C3D_8 Architecture

	Input	Layer Type	#out	NL	s
	3 x 16 x 112 x 112	Conv3D	64	RL	1
	64 x 16 x 112 x 112	MaxPool3D	64	-	2
	64 x 8 x 56 x 56	Conv3D	128	RL	1
	128 x 8 x 56 x 56	MaxPool3D	128	-	2
	128 x 4 x 28 x 28	Conv3D	256	RL	1
	256 x 4 x 28 x 28	Conv3D	256	RL	1
	256 x 4 x 28 x 28	MaxPool3D	256	-	2
	256 x 2 x 14 x 14	Conv3D	512	RL	1
	512 x 2 x 14 x 14	Conv3D	512	RL	1
	512 x 2 x 14 x 14	MaxPool3D	512	-	1,2,2
	512 x 2 x 7 x 7	Conv3D	512	RL	1
	512 x 2 x 7 x 7	Conv3D	512	RL	1
	512 x 2 x 7 x 7	MaxPool3D	512	-	2
	8192 x 1	Linear	4096	RL	1
FV:	4096 x 1	Linear	4096	RL	1
	4096 x 1	Linear	368	-	1

Table A.2: C3D_16 Architecture

	Input	Layer Type	#out	NL	s
	3 x 8 x 112 x 112	Conv3D	64	RL	1
	64 x 8 x 112 x 112	MaxPool3D	64	-	2
	64 x 4 x 56 x 56	Conv3D	128	RL	1
	128 x 4 x 56 x 56	MaxPool3D	128	-	2
	128 x 2 x 28 x 28	Conv3D	256	RL	1
L1	256 x 2 x 28 x 28	Conv3D	256	RL	1
	256 x 2 x 28 x 28	MaxPool3D	256	-	2
	256 x 1 x 14 x 14	Conv3D	512	RL	1
L2	512 x 1 x 14 x 14	Conv3D	512	RL	1
	512 x 1 x 14 x 14	MaxPool3D	512	-	1,2,2
	512 x 1 x 7 x 7	Conv3D	512	RL	1
L3	512 x 1 x 7 x 7	Conv3D	512	RL	1
g1	256 x 2 x 28 x 28	AttentionEstimator3D	256	-	-
g2	512 x 1 x 14 x 14	AttentionEstimator3D	512	-	-
g3	512 x 1 x 7 x 7	AttentionEstimator3D	512	-	-
FV:	256 x 1, 512 x 1, 512 x 1	Concat	1280	-	1
	1280 x 1	Linear	900	-	1
	900 x 1	Linear	600	-	1
Classifier:	600 x 1	Linear	368	-	1

Table A.3: C3D_8 Attn Architecture

	Input	Layer Type	#out	NL	s
	3 x 16 x 112 x 112	Conv3D	64	RL	1
	64 x 16 x 112 x 112	MaxPool3D	64	-	2
	64 x 8 x 56 x 56	Conv3D	128	RL	1
	128 x 8 x 56 x 56	MaxPool3D	128	-	2
	128 x 4 x 28 x 28	Conv3D	256	RL	1
L1	256 x 4 x 28 x 28	Conv3D	256	RL	1
	256 x 4 x 28 x 28	MaxPool3D	256	-	2
	256 x 2 x 14 x 14	Conv3D	512	RL	1
L2	512 x 2 x 14 x 14	Conv3D	512	RL	1
	512 x 2 x 14 x 14	MaxPool3D	512	-	1,2,2
	512 x 2 x 7 x 7	Conv3D	512	RL	1
L3	512 x 2 x 7 x 7	Conv3D	512	RL	1
g1	256 x 4 x 28 x 28	AttentionEstimator3D	256	-	-
g2	512 x 2 x 14 x 14	AttentionEstimator3D	512	-	-
g3	512 x 2 x 7 x 7	AttentionEstimator3D	512	-	-
FV:	256 x 1, 512 x 1, 512 x 1	Concat	1280	-	1
	1280 x 1	Linear	900	-	1
	900 x 1	Linear	600	-	1
Classifier:	600 x 1	Linear	368	-	1

Table A.4: C3D_16 Attn Architecture

	Input	Layer Type	Exp Size	#out	SE	NL	s
	3 x 16 x 112 x 112	Conv3D	-	16	-	HS	1,2,2
	16 x 16 x 56 x 56	LinearBottleneck3D	16	16	-	RE	1
	16 x 16 x 56 x 56	LinearBottleneck3D	64	24	-	RE	2
	24 x 8 x 28 x 28	LinearBottleneck3D	72	24	-	RE	1
	24 x 8 x 28 x 28	LinearBottleneck3D	72	40	*	RE	2
	40 x 4 x 14 x 14	LinearBottleneck3D	120	40	*	RE	1
	40 x 4 x 14 x 14	LinearBottleneck3D	120	40	*	RE	1
	40 x 4 x 14 x 14	LinearBottleneck3D	240	80	-	HS	2
	80 x 2 x 7 x 7	LinearBottleneck3D	200	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	184	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	184	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	480	112	*	HS	1
	112 x 2 x 7 x 7	LinearBottleneck3D	672	112	*	HS	1
	112 x 2 x 7 x 7	LinearBottleneck3D	672	160	*	HS	2
	160 x 1 x 4 x 4	LinearBottleneck3D	960	160	*	HS	1
	160 x 1 x 4 x 4	LinearBottleneck3D	960	160	*	HS	1
	160 x 1 x 4 x 4	Conv3D	-	960	-	HS	1
	960 x 1 x 4 x 4	MaxPool3D	-	-	-	-	1
FV:	960 x 1	Conv3D	-	1280	-	HS	1
	1280 x 1	Linear	-	368	-	-	1

Table A.5: 3D MobileNetV3 Architetcure

	Input	Layer Type	Exp Size	#out	SE	NL	s
	3 x 16 x 112 x 112	Conv3D	-	16	-	HS	1,2,2
	16 x 16 x 56 x 56	LinearBottleneck3D	16	16	-	RE	1
	16 x 16 x 56 x 56	LinearBottleneck3D	64	24	-	RE	2
	24 x 8 x 28 x 28	LinearBottleneck3D	72	24	-	RE	1
	24 x 8 x 28 x 28	LinearBottleneck3D	72	40	*	RE	2
	40 x 4 x 14 x 14	LinearBottleneck3D	120	40	*	RE	1
L1	40 x 4 x 14 x 14	LinearBottleneck3D	120	40	*	RE	1
	40 x 4 x 14 x 14	LinearBottleneck3D	240	80	-	HS	2
	80 x 2 x 7 x 7	LinearBottleneck3D	200	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	184	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	184	80	-	HS	1
	80 x 2 x 7 x 7	LinearBottleneck3D	480	112	*	HS	1
	112 x 2 x 7 x 7	LinearBottleneck3D	672	112	*	HS	1
L2	112 x 2 x 7 x 7	LinearBottleneck3D	672	160	*	HS	2
	160 x 1 x 4 x 4	LinearBottleneck3D	960	160	*	HS	1
	160 x 1 x 4 x 4	LinearBottleneck3D	960	160	*	HS	1
L3	160 x 1 x 4 x 4	Conv3D	-	960	-	HS	1
g1	40 x 4 x 14 x 14	AttentionEstimator3D	-	40	-	-	-
g2	112 x 2 x 7 x 7	AttentionEstimator3D	-	112	-	-	-
g3	960 x 1 x 4 x 4	AttentionEstimator3D	-	960	-	-	-
FV:	40 x 1, 112 x 1, 960 x 1	Concat	-	1112	-	-	-
Classifier:	1112 x 1	Linear	-	368	-	-	1

Table A.6: 3D MobileNetV3 Attn Architetcure

References

[Aafaq *et al.* 2018] Nayyer Aafaq, Syed Zulqarnain Gilani, Wei Liu, and Ajmal Mian. Video description: a survey of methods, datasets and evaluation metrics. *arXiv preprint arXiv:1806.00186*, 2018.

[Aafaq *et al.* 2019] Nayyer Aafaq, Naveed Akhtar, Wei Liu, Syed Zulqarnain Gilani, and Ajmal Mian. Spatio-temporal dynamics and semantic attribute enriched visual encoding for video captioning. *arXiv preprint arXiv:1902.10322*, 2019.

[Banerjee and Lavie 2005] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[Buch *et al.* 2017] Shyamal Buch, Victor Escorcia, Chuanqi Shen, Bernard Ghanem, and Juan Carlos Niebles. Sst: Single-stream temporal action proposals. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2911–2920, 2017.

[Carreira and Zisserman 2017] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017.

[Carreira *et al.* 2018] Joao Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. A short note about kinetics-600. *arXiv preprint arXiv:1808.01340*, 2018.

[Carreira *et al.* 2019] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019.

[Cho *et al.* 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Elfwing *et al.* 2018] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[Escorcia *et al.* 2016] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784. Springer, 2016.

[Ghanem *et al.* 2018] Bernard Ghanem, Juan Carlos Niebles, Cees Snoek, Fabian Caba Heilbron, Humam Alwassel, Victor Escorcia, Ranjay Khrisna, Shyamal Buch, and Cuong Duc Dao. The activitynet large-scale activity recognition challenge 2018 summary. *arXiv preprint arXiv:1808.03766*, 2018.

[Hara *et al.* 2017] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 3154–3160, 2017.

[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.

[Hendrycks and Gimpel 2016] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. 2016.

[Hochreiter and Schmidhuber 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Horn and Schunck 1981] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[Howard *et al.* 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Howard *et al.* 2019] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.

[Hu *et al.* 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[Jetley *et al.* 2018] Saumya Jetley, Nicholas A Lord, Namhoon Lee, and Philip HS Torr. Learn to pay attention. *arXiv preprint arXiv:1804.02391*, 2018.

[Karpathy *et al.* 2014] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[Kay *et al.* 2017] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[Kingma and Ba 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kopuklu *et al.* 2019] Okan Kopuklu, Neslihan Kose, Ahmet Gunduz, and Gerhard Rigoll. Resource efficient 3d convolutional neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[Krishna *et al.* 2017] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 706–715, 2017.

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kuehne *et al.* 2011] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[Levy and Lindenbaum 1998] Avraham Levy and Michael Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. In *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, volume 2, pages 456–460. IEEE, 1998.

[Li *et al.* 2018] Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei. Jointly localizing and describing events for dense video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7492–7500, 2018.

[Lin *et al.* 2017] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[Lin 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[Ma *et al.* 2018] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[MacQueen *et al.* 1967] James MacQueen *et al.* Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[Papineni *et al.* 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[Ramachandran *et al.* 2017] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[Ross *et al.* 2008] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1-3):125–141, 2008.

[Russakovsky *et al.* 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Sandler *et al.* 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[Shen *et al.* 2017] Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue. Weakly supervised dense video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1916–1924, 2017.

[Shou *et al.* 2016] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016.

[Sifre and Mallat 2014] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*, 2014.

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

[Soomro *et al.* 2012] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[Szegedy *et al.* 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[Tan *et al.* 2019] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[Tran *et al.* 2015] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[Vedantam *et al.* 2015] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[Vinyals *et al.* 2015] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[Wang *et al.* 2018] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. Bidirectional attentive fusion with context gating for dense video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7190–7198, 2018.

[Wold *et al.* 1987] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[Xie *et al.* 2018] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[Yang *et al.* 2018] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 285–300, 2018.

[Yao and Li 2018] Ting Yao and Xue Li. Yh technologies at activitynet challenge 2018. *arXiv preprint arXiv:1807.00686*, 2018.

[Yosinski *et al.* 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[Yu *et al.* 2016] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4584–4593, 2016.

[Zoph and Le 2016] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.