DEPARTMENT OF COMPUTER SCIENCE

Challenging a Moment Retrieval model

An investigation into the functionality of 2D-TAN

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Thursday 4th May, 2023

Abstract

In this project I identify the weaknesses in a well known Temporal Grounding model, Two-dimensional temporal adjacency network (2D-TAN). Temporal grounding is the computer vision problem of matching video to natural language and can be thought of as 'video search'. This is an incredibly complex problem as it requires comprehension of natural language and visual semantics *across time*. In this work I show that 2D-TAN is not able to do this effectively and also exhibits a large amount of bias and over-fitting. This is shown through a series of tests, some of which are enabled by a data augmentation and testing pipeline, constructed for this project, that applies temporal augmentations to a sub-sample of the original data-set. I then put forward the case for the potential of temporal augmentations to both eliminate the bias shown by the model and significantly decrease its over-fitting.

Contents

List of Figures

List of Tables

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Michael Wray.

Supporting Technologies

- The language Python was used through-out the project as the models being tested were written in python. The testing pipeline I built was also written in python
- Anaconda was used to create a python environment in which the python code could be run
- I used the torch library for building and executing the deep models
- I used the MoviePy library to perform video augmentations https://zulko.github.io/moviepy/.
- All code was deployed and executed on Blue crystal phase 4, the university super-computer https://www.acrc.bris.ac.uk/acrc/phase4.htm.
- The FFmpeg program was used to extract frames from videos for feature extraction https://ffmpeg.org/
- I used json files for storing annotations for the original and augmented data-sets
- I used HDF5 files for storing the extracted video features for use in the 2D-TAN model
- I used the code supplied by the authors of 2D-TAN for building and testing the 2D-TAN model
- I used pre-trained C3D model checkpoints for extracting the visual features from the activity net videos as well as an accompanying script. https://github.com/yyuanad/Pytorch_C3D_Feature_Extractor
- I used Anaconda to create the python enviroment needed to execute my code on BC4
- I used CUDA to allow the deep models to be execute on GPUs https://github.com/yyuanad/Pytorch_C3D_Feature_Extractor
- I used the numpy library for matrix functions such as generating noise or full 0 matrices.

- I used yt-dlp to download YouTube videos for use in th sampled data-set `https://github.com/yt-dlp/yt-dlp`

Notation and Acronyms

0.0.1 Technical Terms

json stands for java-script object notation and is a 'self describing' data storage format

hdf5 stands for hierarchical data format 5. It is an open source file format for storing large amounts of heterogeneous data in a hierarchical manner

csv stands for comma separated values and is a rudimentary data storage file type

BC4 , Blue Crystal Phase 4 is the latest phase of the university of Bristol's super computer.

Slurm is a job scheduling system utilised by BC4 and is used for queuing jobs on computation nodes.

Bash is a command language used for interfacing with linux OS

0.0.2 Terminology

2D-TAN stands for 2 dimensional temporal adjacency network and is described in section 2.4

Convolution is described in detail in section 2.3.1

Kernels are matrices used in the Convolution operation (2.3.1)

Max Pool Function is described in detail in section 2.3.1

Visual Features . Central to computer vision is the concept of visual features. These are properties of an image or parts of an image. Visual features could be objects in the image e.g. Car, Tree, Ball or a broader concept such as 'nigh vision' or 'underground' (such as the environments described in Gregg and Woodford [8]). Visual features can be hand-made or generated autonomously.

Hadamard Product between two matricies of the same size produces a matrix of the same size where each element in the new matrix is the product of the two elements in the input matrices in the same position

L2 Normalisation is defined as the square root of the sums of the squares of every number in a sequence. It is often used to penalise large weights in deep learning models.

Sigmoid Functions maps anything to between 0 and 1. Typically used to re-frame an input as a probability.

Moment Candidate A proposed section of a video that semantically matches the given natural language query

Anno is often used instead of Annotation in space sensitive environments such as tables and diagrams

Over-Fitting is when a model learns the patterns in the data-set it's trained on too strongly and is then unable to generalise.

Temporal Augmentations Augmentations performed on a video/Annotations that alter their position or speed in time.

Chapter 1

Introduction

From dirverless cars to facial recognition, computer vision (2.1) is now embedded in our everyday lives. For instance, if you use an online photo library, like Google photos, you can search through your entire photo library for certain image features, 'cats' or 'trees' for example. Now imagine if you could do that with videos. This encompasses the concept of the problem of Temporal Grounding. Temporal grounding is a problem in computer vision where, given a video and a natural language query (a sentence), the solution system finds the moments in the video that semantically align with the query. So, given a video of a saxophone lesson and the prompt "a man plays a saxophone", the system will find all parts of the video where a man is playing a saxophone. Many researchers have proposed solutions to this problem (2.2) with ever increasing success. However, these solutions are trained and tested on data-sets (2.5) that may not reflect data found 'in the wild'. This is where the issue of generalisation comes in. Generalisation essentially asks if a model trained on a specific data-set can perform well on data it was not trained or validated on. This is a critical issue as during training a model is never going to be exposed to as much or as varied data as it will be when exposed to 'the wild'. This means that it may have flaws or biases that were not identified during its development that could cause critical issues during deployment. An example of this may be a driverless car trained to stop before hitting children, if it's only trained on a data-set of children wearing red shirts then it may run over children not wearing red shirts. The objective of this project is to investigate and expose the flaws of the '2D-TAN' (2.4) Computer Vision Temporal Grounding (2.2) model, particularly its lack of generalisation. This is an important model to investigate as 2D-TAN is an influential model in the field of Temporal Grounding (2.4) with over 230 citations as of writing. In order to identify its weaknesses, 2D-TAN is exposed to temporally augmented data which tests both its cross-time correlation capability and generalisation. By showing that it's unable to properly correlate vision and language across time I make the case for the use of temporal augmentations to better generalise temporal grounding models.

Chapter 2

Background

2.1 Computer Vision

The field of computer vision encompasses the broad set of problems wherein a computational system is tasked with processing visual data to produce a meaningful understanding and representation of that data.[25]. Included in this field are problems such as object classification (identifying an object in an image [1]), facial recognition (identifying and matching faces[29]), image reconstruction (removing noise from and or filing in missing parts of an incomplete image[27]), and many more. Contemporary solutions to computer vision problems overwhelmingly utilise machine learning, usually in the form of deep learning given its Superior performance over other current methodologies [16].

2.1.1 Statistical Tests

One of the most important statistical measures in computer vision is the Intersection over the Union (IoU)[22]. This is a measure of to what degree two regions overlap. Figure 2.1 demonstrates what each of these areas are.

$$IntersectionOverUnion = \frac{Intersection}{Union}$$

In Computer vision the 'ground truth' is what is known to be true and represents what you want the model to learn from. In image classification this is an area in part of an image or video where the feature the model is trying to classify is located. For example, Figure 2.2 shows the output of a no-entry sign detection system in green and the ground truth locations in red. In this example the predicted candidate may have quite a high IoU with one of the ground truths; however the system fails to identify and produce predictions for the other ground truth regions.

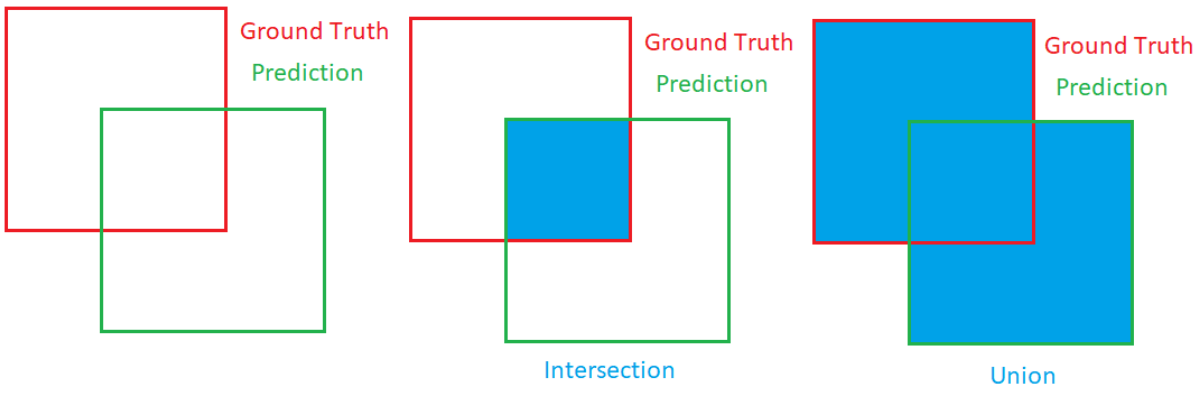

Figure 2.1: Visual Demonstration of the IoU reigons

Figure 2.2: Example output from an object classifier. The ground truths are the red boxes while the predictions are the green boxes

2.2 Temporal Grounding

The focus of this project is on a sub-problem in computer vision, that of Temporal Grounding, which Gao et al[6] defines as "-aims to predict a time interval of a video clip corresponding to a natural language query input", essentially, video search. For example, given a video of someone teaching the saxophone and the query "a man plays the saxophone" the model will return the start and end times for all the moments in the video where a man is playing the saxophone. This problem is also known as 'moment retrieval' because the model *retrieves moments* from the video. The key difficulty posed by this problem is that it requires comprehension of vision and language in the context of *time*. This is because if the text prompt was "a woman opens a door", by looking at a single frame of someone holding a door handle, it would be very difficult to know if they were opening or closing the door. And while current models are very good at classifying features of still images [1] and much work has been done to classify activities [3, 11, 9, 21] the problem of temporal grounding is, comparatively, still in it's infancy [15].

Temporal Grounding is an important problem as video media is now ubiquitous. There is so much of it produced every day (271,330 hours [30]) that it is all but impossible for humans to comprehend and manage it all. Having an automated system to interpret this data into a compressed but still meaningful format would be an invaluable asset for several fields such as research, academia, and content moderation, to name but a few. If a video could be compressed into it's semantic content it would also allow for more efficient data storage as video formats currently occupy a lot of space [18].

Another key issue in temporal grounding is the number of moments considered. The way current models are structured and trained means they only return a single moment for any prompt. They also expect there to be a moment in the video that semantically matches a given query and so cannot handle the case where the semantics of the query don't match any part of the video. Think "A cat playing with string" in a video of a dog playing fetch. The result of this is that models often struggle to correlate multiple moments across time. So, for instance, a model given a video and a prompt "A man plays a saxophone *again*" it will be unlikely to find that moment because it would have to identify the moment where the man plays the saxophone the first time and then look for a *second* moment. This also means that given a generic query of "Door opening" the model will only find a single moment where a door is opened, even if there are many more.

Temporal grounding also has a different relationship to the IoU (2.1.1) than conventional computer vision problems as it identifies points in time, not space. This means the IoU is calculated in a single dimension, time, instead of 2 dimensions, space in an image (as demonstrated in section 2.1.1). Figure 2.3 shows the intersection over union performed in the single dimension of time, while figure 2.4 shows this again but in the context of Temporal Grounding.

A further metric used in temporal grounding is the rank@IoU. This is the main test metric used through both the original 2D-TAN work[34] and this paper. The rank@IoU (n@m) is the percentage of queries having one correctly identified moment in the top n retrieved moments. A moment is correct if

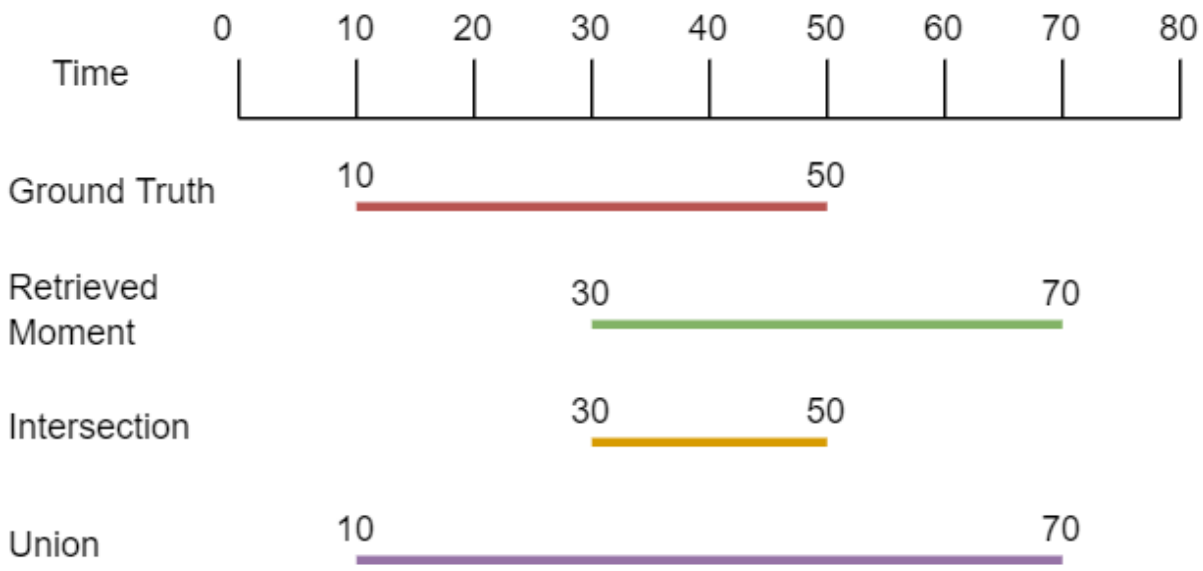

Figure 2.3: Visual example of the intersection and union in one dimensional space. Here the intersection over union would be 1
3 (20
60)

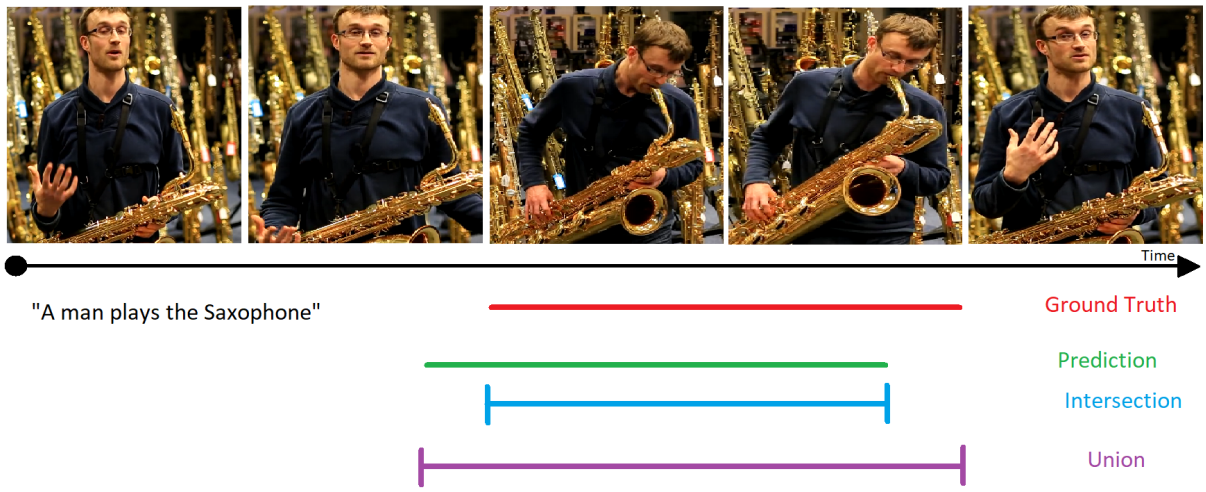

Figure 2.4: An Example of temporal grounding. Shown are the frames from a video of a man reviewing a saxophone with the annotation "A man Plays the Saxophone". Below them are shown example time markers.

its IoU (2.1.1) is greater than m. Figure 2.5 shows how the n@m is calculated for a single annotation.

2.3 Deep Learning

In the past ten years Deep Learning has taken over as the primary vehicle for machine learning given it's superior performance over other techniques [16]. The premise of deep learning is that it mimics the processes of the brain through the use of artificial neurons. Each artificial neuron, called a 'perceptron'[24], takes in several numerical inputs (either from other perceptrons in a lower layer or source data) and multiples each input by a weight specific to that perceptron and input. It then sums all the modified inputs and applies a function to the result. The product from this function is then outputted from that perceptron to perceptrons in the layer above or an output layer. These perceptrons are then stacked in layers so that each layer receives inputs from the perceptrons in the layer below it and outputs its

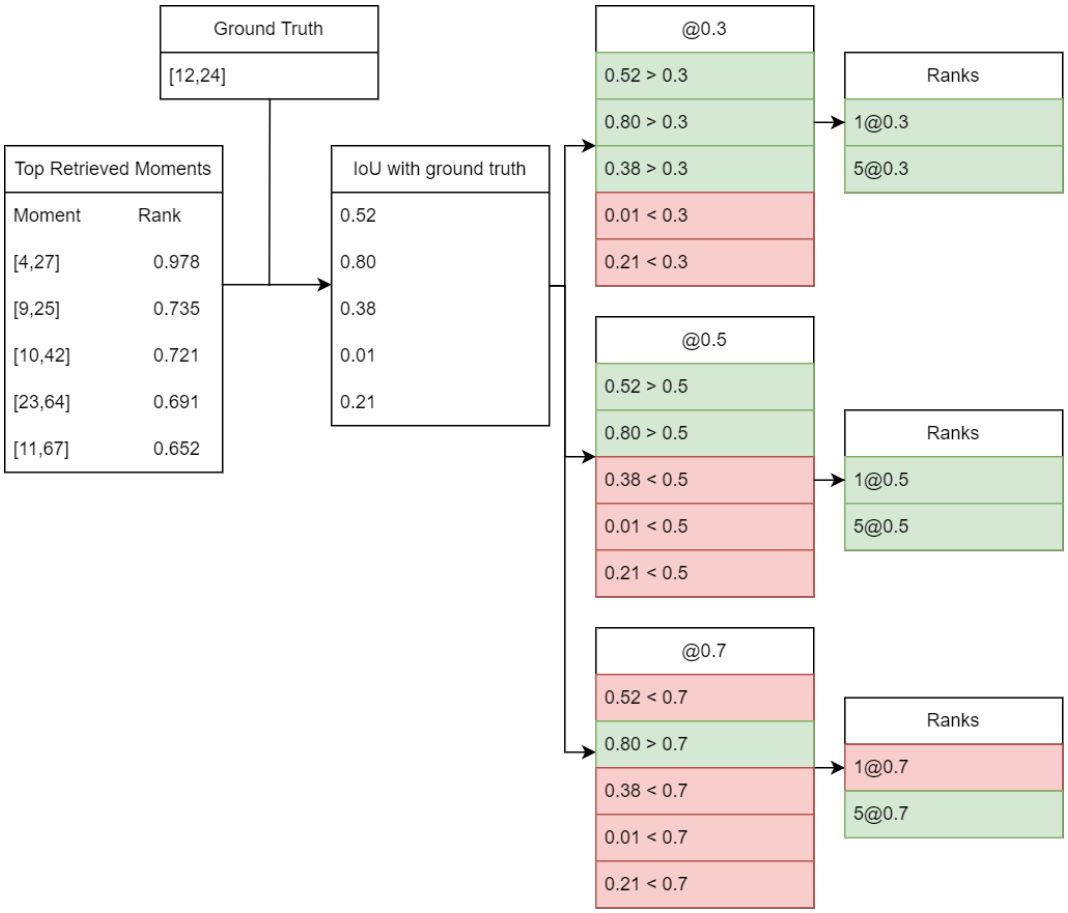

Figure 2.5: This figure shows what the rank @ IoU looks like. The initial retrieved moments are ranked on their weighting. This example is correct for all six n@m except the rank 1@0.7. This is because, while it does have an over 0.7 IoU correct moment this moment was not the number one ranked.

signal to perceptrons in the layer above it. In this way a network is created which, given an input, will propagate it through the network to produce a result. A fully connected layer is a layer where every perceptron in the layer below is connected to every perceptron in the layer above. Figure 2.6 shows a simple network with two fully connected 'hidden' layers (layers between the input and output). In order for these networks to learn, they need to change the weights each perceptron applies to it's inputs so as to modify it's output. The intent is that each perceptron will learn to identify a feature of the input data and recognise when that feature is present. The network is trained by taking in an input and calculating the difference between the networks actual output and what it should have outputted. This difference is then 'back propagated'[32] through the network to modify each weight according to it's influence on the networks output. By repeating the process of giving the network an input and then modifying its weights based on how incorrect the network was, it can be trained to identify features within the data-set it's trained on.

2.3.1 Convolutional Neural Networks

However, fully connected deep networks have a significant limitation as the deeper and broader you make a network, the more weights (parameters) you have to train. Contemporary models have upwards of several billion parameters which makes training them incredibly computationally expensive. This has led to the concept of sharing weights. Instead of each perceptron having it's own weights, it can share weights (but not inputs) with every other perceptron in it's layer. This led to the adoption of the convolution operation. The convolution operation is performed by sliding a smaller matrix (kernel) over the input matrix. At each step element-wise multiplication is performed between the kernel and the overlapping values in the input. The results are then summed and output to a corresponding cell in the output matrix. The Kernel is then shifted, the number of cells it's shifted by is called it's 'stride' and is usually just one. Convolution is trained in the same way as standard deep networks, via the back propagation algorithm

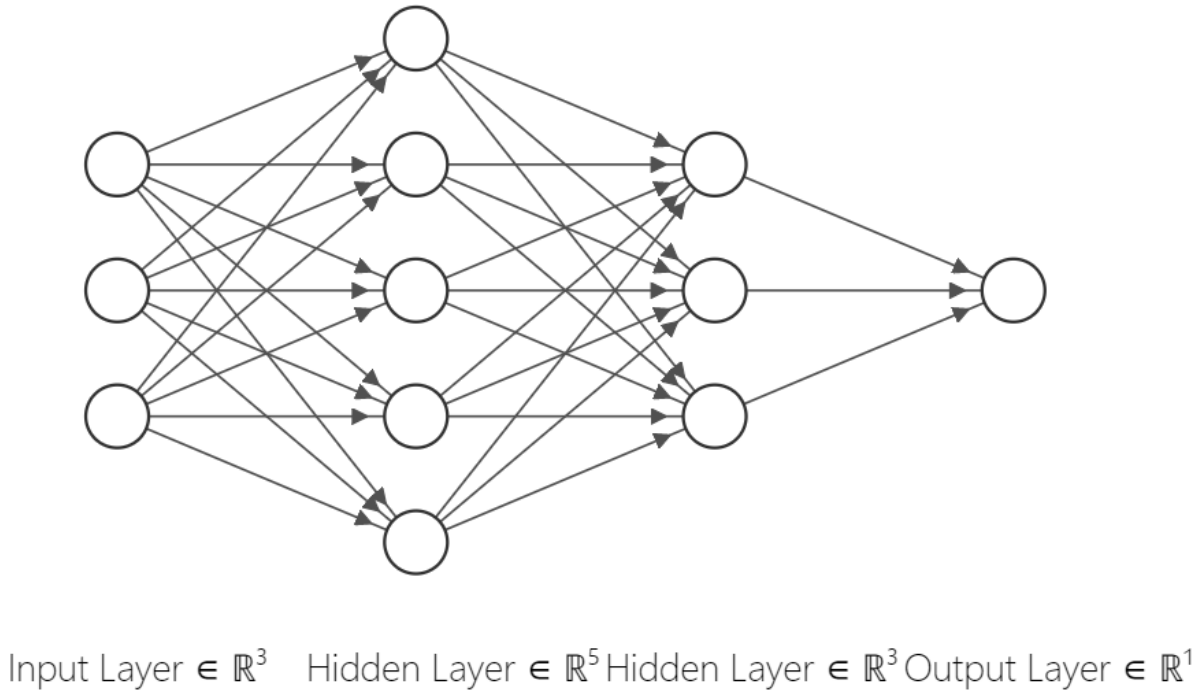

Figure 2.6: An Example of a simple fully connected Network with two different sized hidden layers and a single output node.

[32]. Convolution also has the advantage that several convolutions can be performed on the same input layer meaning several different features can be extracted at each layer. Figure 2.7 shows an example of convolution. In this example there are three outputs for the one input. This is because three different convolutions were performed with three different convolution Kernels. Convolutional neural networks were initially inspired by the structure of the visual pathway in the brain where neurons would focus on a particular part of the visual field. The deeper the layer, the larger the receptive field and the more complex the features a kernel identifies. For example, in the lower layers the Kernels learn to identify basic shapes, like lines in a particular part of the image. The Kernels above then learn to combine these basic shapes into compound shapes such as a square in broader parts of the image. The final layer then identifies the desired features, e.g. a car, or a person, in any part of the image. Each layer of convolution also broadens the receptive field from which a perceptron is influenced. This allows for upstream features in a single part of the input to influence larger and larger parts of the network.

Pooling is an operation that downsizes an input matrix while retaining key information. It separates the matrix into discrete regions and then summarizes these regions in some manner. For instance, max-pooling summarizes each region by taking the highest value in each region.

2.3.2 3D Convolution

Convolution can also be extended into 3D space for videos. 2D convolution does use 3D kernels and matrices as images have three colour channels, Red, Green, Blue. However, the kernel only moves in two dimensions, it doesn't change which colour channels it's looking at, the 3D kernel looks at all of them at once. 3D convolution does move the 3D kernel through 3D space. This 3D space is the two pixel dimensions and the time dimension (not the RGB channel dimension). 3D convolution has also been shown to have higher performance in certain tasks when compared to 2D convolution [26].

2.4 2D-TAN

2D-TAN is the name of a solution to the aforementioned Temporal Grounding problem 2.2, proposed by Songyang Zhang, Houwen Peng, Jianlong Fu and Jiebo Luo in their 2020 paper[34] "Learning 2D Temporal Adjacent Networks for Moment Localization with Natural Language". Their solution uses a

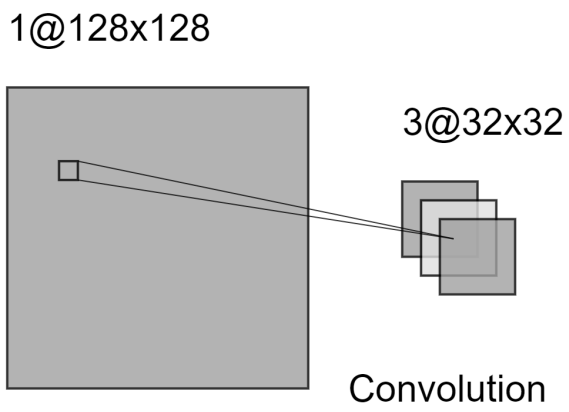

Figure 2.7: An Example of a very simple network with a single hidden layer.

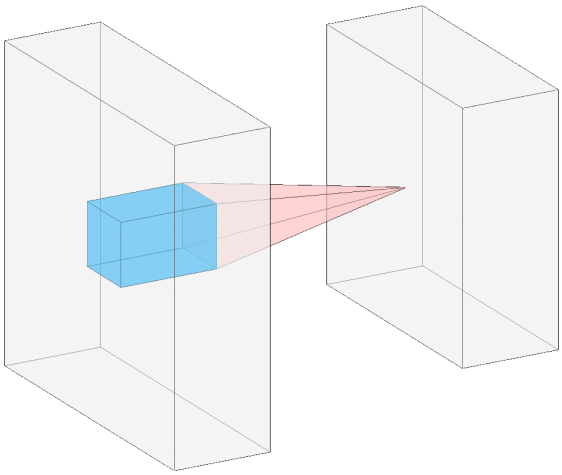

Figure 2.8: An Example of a very simple network with a single hidden layer.

set of fixed moment candidates which then have the visual and textual features layered on-top. They call this arrangement a 2-dimensional temporal adjacency map (2.9), hence 2D-TAN (2-dimensional temporal adjacency network). The core of my project is investigating the weaknesses of this model due to its influence in the field of temporal grounding, with over 230 citations as of writing. The code for 2D-TAN is publicly avialable at https://github.com/microsoft/VideoX/tree/master/2D-TAN.

2.4.1 Clips

2D-TAN first decomposes a video into *clips* of size T frames. These clips are then sampled over a fixed interval to produce N sampled clips. The features of each of these sampled clips are then collected from the features of their constituent frames. Each frame has its features extracted via a deep network specific to the data-set it came from. For instance the videos from Activity net have the features from their frames extracted by a model called C3D (2.6).

2.4.2 2D Feature map

The core of the 2D-TAN model is the 2D temporal feature map. This is an NxN table with the index of each of the N sampled clips arrayed along both axis (see fig 2.9). Each cell in this table represents a *moment candidate* starting at clip i (vertical axis) and finishing at clip j (horizontal axis). i must be equal to or larger than j as a moment can't start after it ends, this means half of the table is empty (the white cells in figure 2.9). The arrangement of moments means that moments that are near to each other temporally (i.e. they have similar start and end times) are placed near each other spatially. The features for each moment candidate in the 2D-temporal map are max-pooled (2.3.1) from the features of

it's composite clips.

sparse Sampling

In order to save memory the table also uses a sparse schema wherein longer moment candidates are sampled less frequently. If they were sampled as frequently as the shorter moments they would have a huge overlap and therefore very little difference between each moment. The sparse sampling can be seen in the top right of figure 2.9 where the blue cells denote the sampled moments and the grey cells denote the ignored moments.

2.4.3 Sentence Features

The natural language features are extracted from the query sentence by applying the GloVe model [19]. This produces a set of word embeddings which are fed through a three layer LSTM network [10] who's final hidden layer is used as the representation of the natural language query features. The natural language features are then fused with the temporal adjacency map by first projecting both of them into the same subspace via fully connected layers then performing the Hadamard product and L2 normalisation over them. At this point the natural language and visual features are fused in such a way that the moment candidates that match the semantics of the natural language query have a higher weighting in the 2D-Temporal Adjacency Map.

2.4.4 Convolution

The 2D temporal adjacency map, now with fused language features, is then passed through L convolutional layers with kernel size K. These variables, L and K are changed depending on the data-set. This stage allows moment candidates to share their weighting with adjacent moment candidates. Because moments that are near each other in time are arrayed near each other in space, temporally similar moments will share their weighting as they likely share semantic content due to their temporal adjacency. Finally it's run through a fully connected layer and a Sigmoid function before being projected back into the 2D-map space. The outcome of this is a heat-map with the same structure as the 2D-temporal adjacency map. The heat (weighting) of each cell represents the probability that the corresponding moment candidate matches the semantics of the natural language query. The authors claim the structure of this map allows the model to identify moments with the semantics of "a person does x *again*" as the first instance is also identified in the map and then shares its weight with the second instance. The Strongest moment candidates in the heat-map are then extracted as the proposed moment candidates for the model.

2.5 Data-sets

2D-TAN was trained separately on 3 different benchmark data-sets, Charades-STA [5], ActivityNet Captions [9] and TACoS [23]. However, for the purposes of my investigation I only used the Activity net data-set as this has the largest number of videos and annotations, therefore should produce the most robustly trained model as it will have the strongest signal. It was also the data-set the authors of 2D-TAN used for their ablation study meaning the model is better optimised for 2D-TAN. This shows in their results as activity net is the data-set the model performed best on.

2.5.1 Activity Net Captions

Activity Net Captions is a data-set of annotated untrimmed videos designed to teach models to recognise activities and actions [28]. The data-set is comprised of YouTube videos with multiple annotations per video and is based on the original Activity Net data-set [9]. The Activity net annotations are also rich and diverse; annotations over videos with similar content, e.g. playing the bagpipes, may have very different annotations which incorporate not just the action, playing bagpipes, but also the context such as descriptions of the person, environment and even in some cases the facial expressions of people in the video. This level of detail can be obtrusive to models learning simple actions but for more complicated models, such as those performing Temporal grounding, it may have the benefit of improving their generalisation by also making them focus on the context in which an action occurs and not just the action itself. By training these models on such data it makes them more prepared for the types of data they'll face deployed 'in the wild'. Activity Net Captions is split into three partitions, train, validate and test. The train and

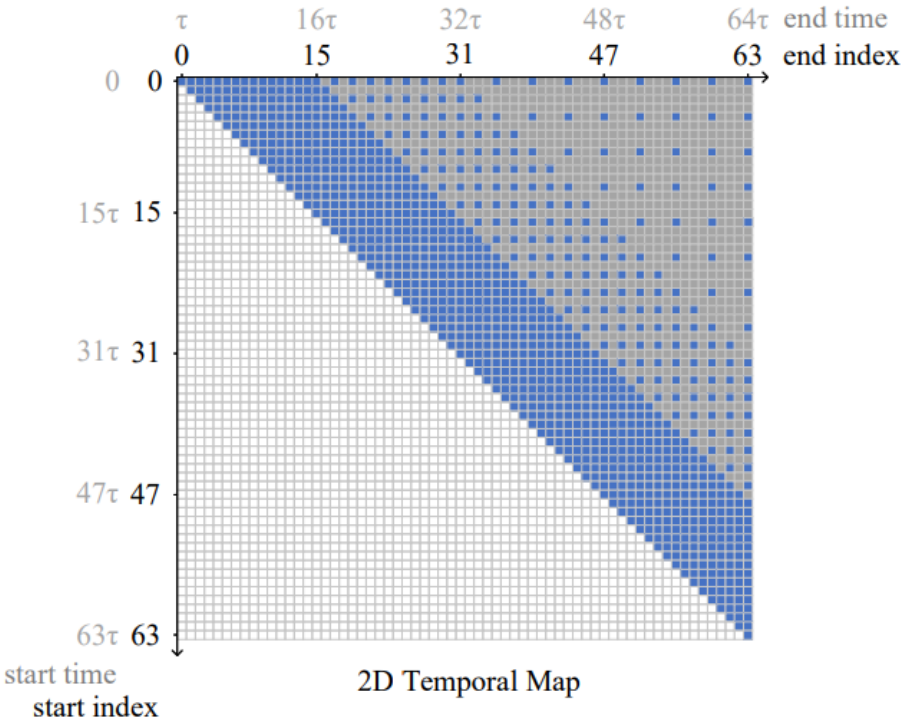

Figure 2.9: "The selection of moment candidates when there are N = 64 sampled clips in an untrimmed video. The upper triangular part of the 2D map enumerates all possible moment candidates starting from clip va to vb, while the lower triangular part is invalid. In our method, only the blue points are selected as moment candidates." Source 2D-TAN [34]

validate caption sets are for training the model while the test caption set is used for testing the model once it's trained. This means that the model has not seen the test set before it's trialed on it. This is supposed to show the models generalisation, that it can perform well on data it wasn't trained on.

2.5.2 Sport1M

Sport1M [12] is a data-set of over a million videos covering 487 different sports and is designed to train models in the task of action recognition. This data-set was used to train the C3D model checkpoints which were then used to generate the features for the augmented videos in section 3.7.

2.6 C3D

It is common practice for more complex deep learning models, such as temporal grounding models, to not extract the features themselves. Instead a different model, which is trained separately, is used to extract the visual features. The original features were extracted using a type of model called C3D [26]. This is a network that makes extensive use of 3D convolution (hence the name, Convolution 3 Dimensional, C3D) 2.3.2 to classify visual features across time. It does this by stacking several 2D frames into a 3D volume and then performing 3D convolution over this volume. Figure 2.10 demonstrates the advantages to 3D convolution compared to other forms of convolution, particularly that it allows for modeling *across time*. Using a separate network for feature extraction has the distinct advantage that it requires training fewer weights at a time. Instead the feature extraction model can be trained to extract the desired features and then the Temporal grounding model can be specifically trained to perform temporal grounding on those features.

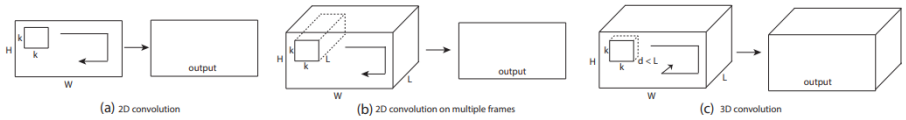

Figure 2.10: "2D and 3D convolution operations. a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal." Source C3D [26]

2.7 Other Works

In their 2022 work [17] Li et al explore how augmenting the composition of the natural language query given to a temporal grounding model can affect its ability to correlate vision and language. They find that many state of the art models (including 2D-TAN) fail at this task. In my work I analyse how well 2D-TAN is able to perform when the video input is augmented temporally.

Augmentations are transforms applied to training data to generate new training data [35, 13, 2]. This increases the amount of training data available while also preventing over-fitting and increasing generalisation by giving the model more varied data. Examples of common visual augmentations are flipping the image, increasing contrast and brightness, and cropping. However, the field of temporal augmentations is relatively under-explored. In their 2019 paper [20] Price and Dima investigate video transforms that alter annotations. One such augmentation they investigate is reversing a video which they found has a highly contextual effect on the video annotation. For instance, a vague annotation such as 'moving something' would be considered invariant under their frame-work as played backwards the video and language would be semantically the same, something would still be moved but in the opposite way. A more specific annotation like 'putting something in-front of something' would be considered Equivariant under their framework as when reversed the annotation would change to 'removing something from in-front of something'. The final category of augment in their framework are those that are irreversible, typically due to entropy such as 'a stack of cards knocked over'. Figure 2.11 showcases this. In my work I look at other forms of temporal augmentation such as speeding the video up and shifting annotations by putting another video in-front of the video the annotations correspond to. I even change the annotation semantics in one of the augmentations by adding the word 'again' to the end of the annotations.

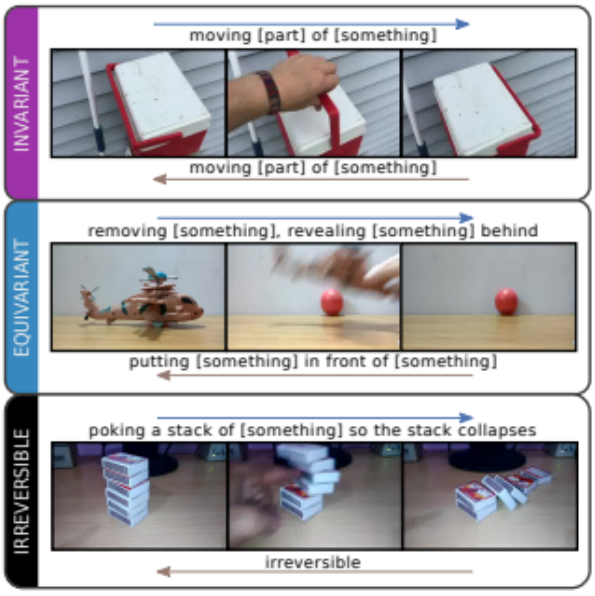

Figure 2.11: "When time-reversing a video, invariant actions (top) maintain their label, equivariant actions (middle) exchange labels, while some actions (bottom) are irreversible producing motions that defy laws of physics." Source: [20]

Another key consideration is *why* augmentations are necessary. A deep learning model can be thought of as identifying a 'signal' (pattern) in the data it's trained on. The problem is that models often over-fit to this signal meaning they fail to generalise well. This over-fitting can be even more problematic when the data has a bias in it. In [33] the authors identify a significant temporal bias in Activity Net Captions towards the start of the video. Figure 2.12 shows this visually. Many of the annotations in Activity Net Captions start at the beginning of the video meaning it has a large temporal bias towards 0.

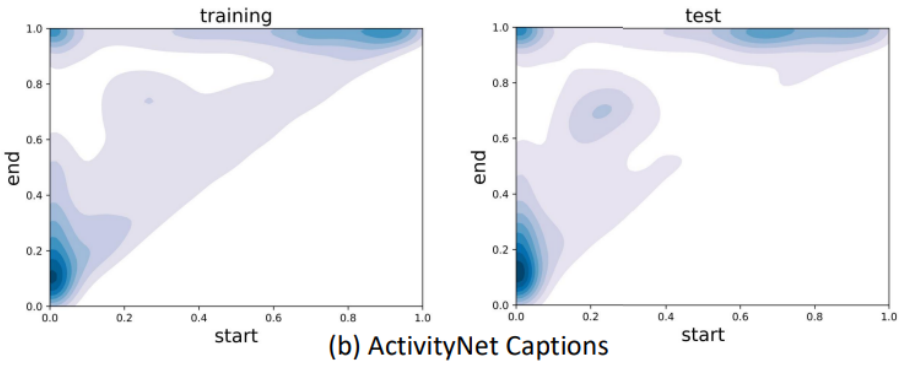

Figure 2.12: "The ground-truth moment annotation distributions of all query-moment pairs in Charades-STA and ActivityNet Captions. The deeper the color, the larger density in distributions." Source: [33]

2.8 Objective

The objective of this project is to investigate the weaknesses of 2D-TAN, particularly when it is subjected to visual data manipulated by novel temporal augmentations. By building on previous works [34, 20, 15] on temporal grounding and temporal augmentation (particularly Price and Dima) I hope to prove that 2D-TAN is incapable of generalising across all these augmentations, particularly those designed to test its capability to correlate language over time.

Chapter 3

Project Execution

3.1 Deploying 2D-TAN

The first stage of the execution of the project was to deploy 2D-TAN (2.4) to BC4 and run the author provided benchmark. After cloning the code repository; the model checkpoints, annotations, and visual-features also had to be downloaded and put in the correct folders. This was made more complicated by the fact that I chose to put many of these in the BC4-scratch space. Anaconda was used to construct a python environment in which 2D-TAN could be run as the model required many modules that weren't available directly through BC4 but which were available through python. I then wrote a bash script to be used by Slurm on BC4 which first loads CUDA and Anaconda before invoking the python environment and running the 2D-TAN python program. With 2D-TAN properly established I tested it's performance on the test set, the results of which are shown in 4.1.1.

3.2 Dissecting 2D-TAN

The next step was to understand how 2D-TAN functioned so that I could modify its functionality to test it on extreme data as well as compare it to the author reported functionality. This necessitated going line by line through its core modules annotating each line as the authors didn't annotate any of the code-base. The code was designed to work with multiple data-sets meaning the core elements were programmed in a modular manner, which, while good practice, made comprehending it more difficult. The central module is the engine, which, given a data loader (a torch wrapper for a data-set) and a network, will iterativly train the network on the data-set. The second two modules are the test and train modules. These act as a wrapper for the engine, supplying it with the data-loader and network as well as instructions on whether to train or test the network. They also contain function call-backs for the engine which are used to calculate the loss function as well as define the network. Each data-set also has its own class which is used to extract its features and annotations and is an extension of the torch class 'dataloader'. As an extension of 'dataloader' it must over-ride a function called 'getItem'. The dataloader object is passed to the Torch system which calls the getItem function whenever it needs a new sample from the data-set to train the given network. The most important section of code for the purposes of my project is the 'network' function in the test.py module. This function defines the network and is passed to the engine which trains it. The function takes in a variable called sample which is a dictionary that contains the visual and textual features as well as other data-points needed by the network. The function starts by extracting all of these data-points into local variables before feeding them into the actual network (which is defined elsewhere in the class). Between these two sets of code is where I inserted several lines of code used in the initial experiments (Section 3.3).

3.3 Initial Experiments on 2D-TAN

Initial testing was concerned with feeding the 2D-TAN model extreme data in an attempt to ascertain how it was calculating it's moment boundaries. The test metric is the n@m IoU (2.2). In all the experiments conducted, across this paper, six n@m (2.2) points were used 1@0.3, 1@0.5, 1@0.7, 5@0.3, 5@0.5, 5@0.7.

3.3.1 The worst performance case

The hypothetical worst performance would be if the model were to randomly guess the moments as any performance worse than this would have to be artificial. In order to ascertain a benchmark of the theoretical worst performance, the joint probabilities, that is the best predicted results, were replaced with a Gaussian noise of (mean 0 and standard deviation 1). The grounded moments from this are essentially random guessing. This experiment was performed 3 times with the mean average of the results taken. The results are shown in section 4.1.2.

3.3.2 Zeroing the input

The next set of experiments on the model were to give it a version of the data-set where the feature values were set to 0. The first of these saw the visual features set to 0 while the word features were kept the same. The word features were then zeroed while the visual features were kept the same. Lastly both the visual and word features were set to 0. The results of these experiments are shown in section 4.1.3.

3.3.3 Statistics

In order to gauge the effect of levels of noise on the performance of the model I ran some statistical tests on the original features to identify their overall mean and standard deviation. The results of which are shown in section 4.1.7. These results were used in later experiments to gauge the magnitude of noise to add to the different feature sets (section 3.3.5).

3.3.4 Fully Replacing Data with noise.

The next set of experiments involved fully replacing both the visual and word features with Gaussian noise with a mean of 0 and a standard deviation of 1. They were each replaced separately and then both replaced at the same time. Each experiment was repeated 3 times and the mean average taken. The results for each experiment are shown in Section 4.1.5.

3.3.5 Gradual addition of noise

In the third round of experiments I gradually added ever increasing degrees of noise to the original feature sets. Given the difference in influence on the prediction between the visual and word features (4.1.5) I added noise in different orders of magnitude for each feature set. All reported results are from the addition of noise from a normal distribution with mean 0 and a given standard deviation. Each experiment was conducted 3 times and the mean average taken. The results from these tests can be found in section 4.1.6. The choice of standard deviations for each model was influenced by the results from the statistical analysis in section 3.3.3.

3.4 Augment Planning

The main core of this project was the automatic generation of augmented data to feed into 2D-TAN to identify it's flaws in temporal reasoning. The data augmentations chosen were heavily influenced by the functions available in the MoviePy library as well as by the observations made during the initial testing of 2D-TAN. I was initially going to implement three types of augment: time acceleration (speeding the video up), concatenation (adding a random video to the front of the chosen video) and reverse (reversing the video). However, reversing the video would have been extremely costly in terms of time as each annotation would have had to have been manually reversed as reversing the semantics of a natural language query in time is an incredibly complicated task for a computer to do and would likely require its own research project (2.7). Instead I replaced it with an augment I called 'again' which simply concatenated the same video in-front of its self and adjusted the times of the annotations to all be in the second iteration of the video as well as appending the word 'again' to the end of each annotation. I chose this augment as one of the principle claims by the authors of 2D-TAN (2.4) is that their model should be able to correctly identify and correlate such phrases. Figure 3.1 demonstrates how these augmentations function.

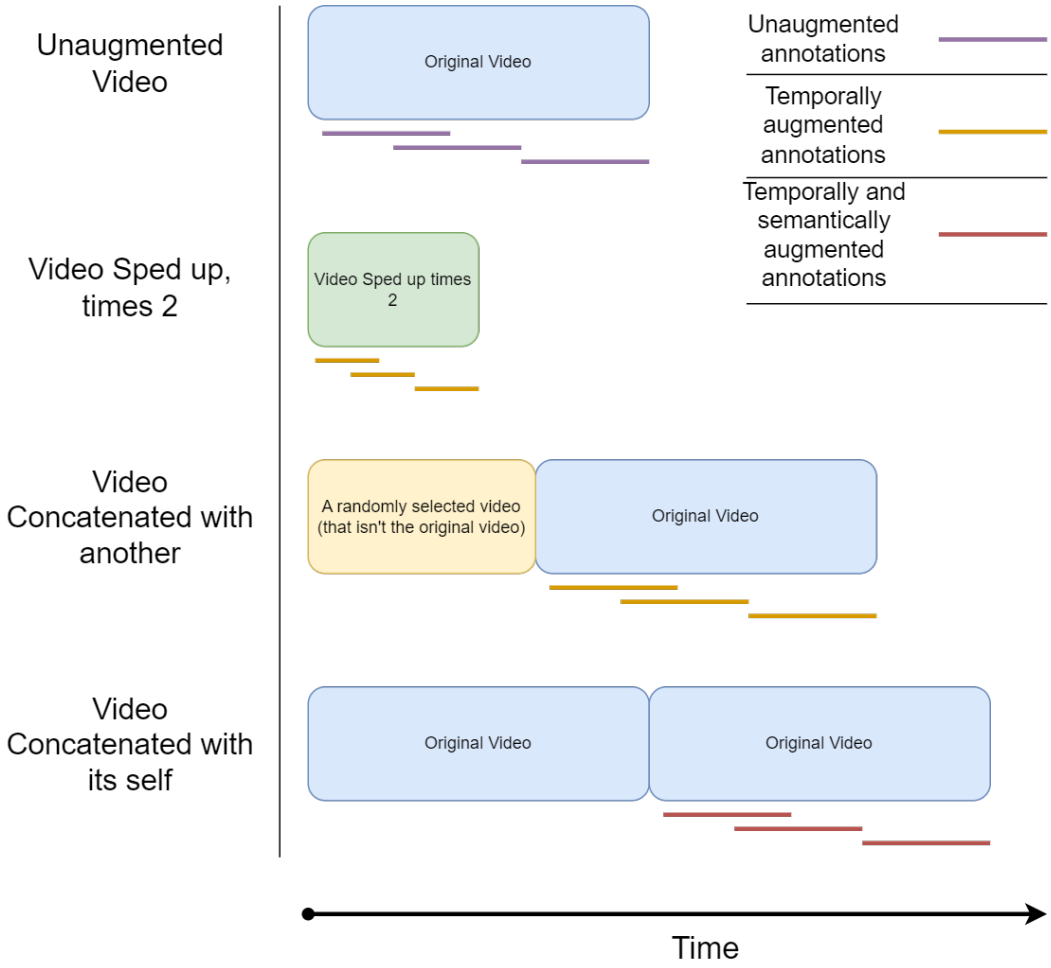

Figure 3.1: Figure shows a diagram of the effect of the augments on the videos and their annotations. The bars under each video indicate the position and duration of each annotation in time. The bottom most augment is the 'again' augment wherein each annotation has the word 'again' concatenated to it's end as well as being temporally shifted so as to be in the second copy of the original video

3.5 Programming the Augmenter

The Augmenter was written in python as this would allow for it to be deployed to BC4 in the same Anaconda environment as 2D-TAN. I took a modular approach keeping each functionality to it's own class. Listed below are the classes used in the augmenter:

- Augmenter core. Core module which utilises the others to augment the video and annotations.
- Annotation Editor. Module for editing the annotations.
- Clip Editor. Module for editing the actual video clips.
- Annotation Manager. Module for reading in base annotations and outputting the edited annotations

3.5.1 Augmenter Core

The core module of the augmenter operates around a subclass called Trial. Each trial represents an annotation, not a video, as there are multiple annotations per video. The augmenter takes a single parameter (as the file names were hard-coded), called flag which indicates the type of augment to perform. The Augmenter first reads in a list of videos and their annotations from a json file via the annotation manager module. It then goes through each annotation and creates a trail for it. The contents of the trial are based on the type of augment being performed (which is determined by the flag) however every trial has a path to it's video clip, as well as the video annotation its built around. From there it has additional

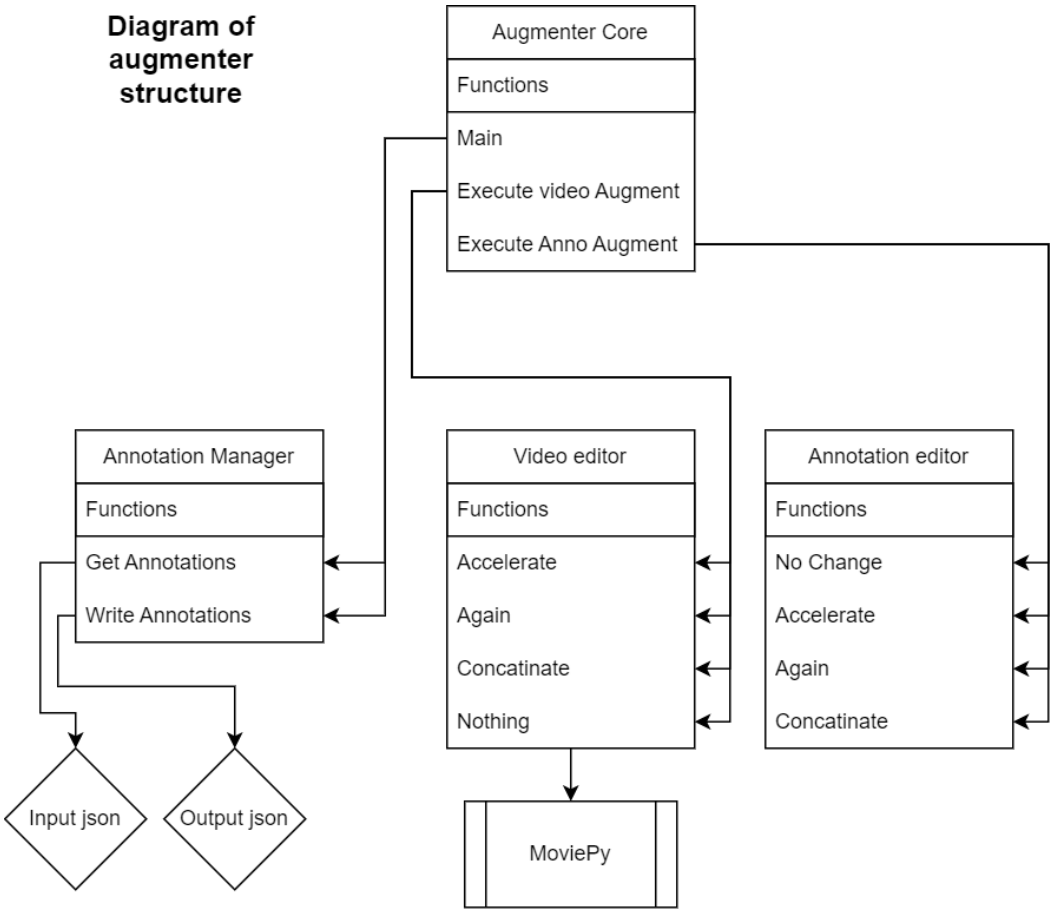

Figure 3.2: Figure shows a diagram of the class structure of the augmenter

information added based on the augment type. For instance the name of the output video is changed to reflect the augment, in the case of the speed up augment this involves adding an 'X' {speed} to the end of the video name. The speed up augment also adds a value to the delta section in the trial, this is used to control the speed of the video when it's augmented. The most complicated trial to establish is the Concatenate trial as this first needs to find a random video that isn't the current annotation's video. It then adds the directory for this video to the trial object so that, come the augmentation, it can add it to the start of the annotations' video.

After the trials are established, they are then each executed. Only the first annotation for each video calls the 'augment video' function so as to not re-augment the same video for each annotation. The augment video function takes in a trial and uses the 'transform' variable in the 'trial' instance to identify the augment to be performed. The actual video augmentation is performed by the video-editor class, which behaves as a wrapper for moviePy's functionality. The augment annotation function is all but identical to the augment video function, except it calls the annotation-editor module. It also receives an augmented annotation from the annotation-editor which is added to the list of augmented annotations. Once every annotation and video has been augmented, the augmented annotations are output to a new json file by the annotation manager. An example of such a file can be found in the appendix A.

3.6 Data Sampling

Due to the immense size of Activity Net (2.5.1) it was not practical to trial the augment system on the entire data-set, primarily due to storage constraints on BC4. Instead I handpicked a small but diverse subset of the data to trial the augments. I was still considering using a reverse augmentation (2.7, 3.4) when selecting the data and so sampled more videos which were equivariant or invariant (2.7) across time. However, such videos are rare, they only make up 50% of the hand selected data-set, and can only be qualitatively assessed. With the sub-set of videos selected I retrieved them from YouTube using yt-dlp. I then found their annotations, from the train and test annotation files provided by the 2D-TAN

authors, and moved them into their own json file. The sampled data-set consists of 43 videos with 143 annotations over 15 categories of activity. These categories refer only to the main activity in the scene, the associated annotations for each video capture more behaviour than just the labeled activity.

3.7 Generating C3D features for 2D-TAN

In section 2.6 it is stated that the 2D-TAN code is not capable of generating the visual features from a video for Activity Net. I needed the ability to do this so I could generate the visual features from the augmented videos. In order to get the C3D checkpoints used to generate the visual features the authors published, I contacted them. They didn't have the checkpoints either as they acquired the visual features as part of a competition organised by the people behind Activity Net Captions (2.5.1). Instead the authors suggested that I use a different set of checkpoints, which they sent me the link too, and then re-train 2D-TAN using features generated from these checkpoints over the entirety of Activity Net. However, as stated in section 3.6, doing any operation on the entirety of Activity Net was practically impossible due to time and space constraints. I tried it anyway but after leaving the feature extractor to run for 48 hours it had only managed to process 99 videos out of 7,800. At that rate it would have taken over 5 months to extract them all. I could have tried to parrallelise it; but again, doing so would have taken more space and time than I had available. The pre-trained C3D checkpoints also came with a python script that would run a given list of videos through the network. I took apart this script in order to make it function to my needs. Instead of having it take in the video names from a text file I had it take in the augmented annotations json file. This new script goes through each input video and extracts all of its frames using the FFmpeg program. It then pre-processes each frame before putting them through the C3D model and outputting the feature set to a HDF5 file. The final number of visual features produced by this model is 487 while 2D-TAN operates on a feature set of size 500. This necessitated padding the last 13 features with zeros. The use of FFmpeg on the command line by the extraction script also meant I had to add FFmpeg to the slurm bash script for the feature extractor as FFmpeg wasn't run in the python environment.

3.8 Assembling the Pipeline

With the Augmenter and C3D extractor finished, all that was left to do was to build a data pipeline from the source data to retrieved moments. Figure 3.3 shows a diagrammatic representation of the pipeline. Each module was connected by mounting shortcuts to folders in the scratch space, which is shown in figure 3.4. This meant that data was immediately available to each module without having to manually move it; while also taking advantage of the additional storage provided by the scratch space. The C3D extractor module also copies over the augmented annotations for use by 2D-TAN. During the use of the pipeline in section 3.9 the C3D feature extractor was found to be very slow so I copied it to create a crude parallel set-up; although I still had to manually go in and link the second C3D extractor with its own directories in the scratch space so as to prevent the extractors over-writing each other's output. Each part of the pipeline was in its own self contained directory and had its own Slurm bash script which meant that they had to be manually activated once the last part of the pipeline had finished. This was very useful as it meant that I could examine the output from each module individually before the next module started operating on it meaning bugs were caught far sooner in the augmentation and testing process.

3.9 Exposing 2D-TAN to augmented Data

With the pipeline constructed all that was left to do was to test it. During testing many bugs were discovered and fixed, however, it did eventually work as intended. But it was still quite slow, in section 3.8 I mention that I doubled the C3D extractor to try mitigate this. With the pipeline working I was able to easily apply each of the 3 augments to the sampled data-set and then test 2D-TAN on each of these augmented data-sets along side the un-augmented sampled data-set. In order to get the best predicted moment for each video out of 2D-TAN I needed to modify it slightly. This was done by inserting a couple of lines of code into the network function to record the best predicted moments and then output them to a CSV file at the end of the main function. An example of such an output CSV can be found in the appendix A. I also changed the parameters of the pool-test so that only a single annotation would be tested at a time. With this done, 2D-TAN now created a CSV with each annotation sentence, ground

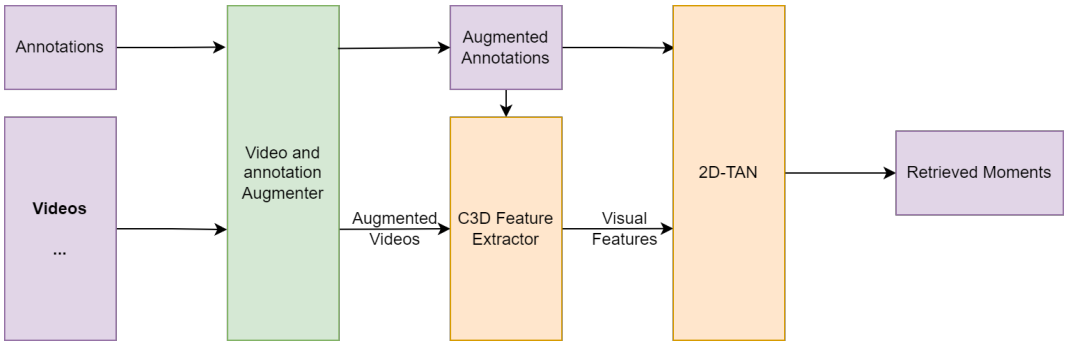

Figure 3.3: Figure shows a diagram of the constructed pipeline

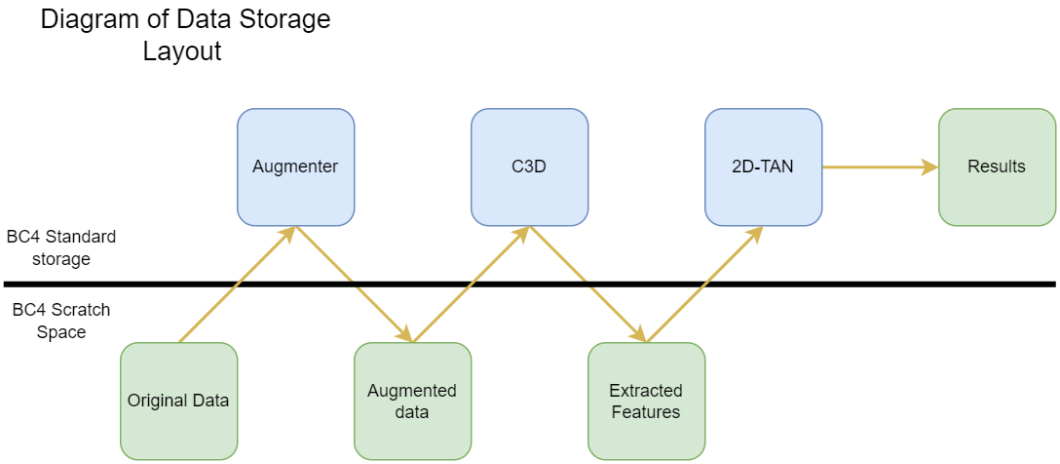

Figure 3.4: Figure shows a diagram of the positioning of modules and folders within the two main BC4 storage spaces.

truth moment and best predicted moment for every data-set it was trialed on. Section 4.2 shows the results of this testing.

3.10 Analysis of Activity Net Captions

During section 3.9 I noticed that allot of the retrieved moments started at 0, i.e. the start of the video. This led me to believe that the model may be biased towards the start of videos due to a bias within the data-set. With this in mind I created a simple python script to count the number of annotations in each of the sets that started at zero. The results of this analysis are shown in section 4.1.7.

Chapter 4

Evaluation

4.1 2D-TAN original data-set experimentation

The following sections present the results from the experiments conducted in Chapter 3 along-side detailed analysis of what these results show. Section 4.3 summarises the findings from the analysis as well as postulating other weaknesses inherent to 2D-TAN.

4.1.1 Initial Results

Table 4.1: Table of author reported results and initial test results

Result Set	1@IoU0.3	1@IoU0.5	1@IoU0.7	5@IoU0.3	5@IoU0.5	5@IoU0.7
Author Reported	59 .45	44.51	26.54	85.53	77.13	61.96
My findings	59 .45	44.51	26.54	85.53	77.13	61.96

Shown in table 4.1 are the results from running 2D-TAN on the original data-set as described in section 3.1. My findings matched the authors exactly corroborating their performance claims.

4.1.2 Worst case Results

Table 4.2: Table of results when the joint probability was set to be random

1@IoU0.3	1@IoU0.5	1@IoU0.7	5@IoU0.3	5@IoU0.5	5@IoU0.7
27.37	13.9433	4.8766	66.3833	45.1133	18.0633

In table 4.2 are shown the results from the test described in section 3.3.1. These results show a significant drop in performance across all categories and a particularly high drop in the more stringent categories 1@0.7 and 5@0.7. This shows that the model is gaining performance over just randomly guessing.

4.1.3 Zeroed Results

Table 4.3: Table of results from experiments where features were set to 0

Test	1@0.3	1@0.5	1@0.7	5@0.3	5@0.5	5@0.7
Visual Features Zeroed	51.07	27.16	13.92	85.00	74.41	56.93
word Features Zeroed	44.24	29.08	14.96	66.75	56.49	40.85
Both Feature sets Zeroed	1.45	0.80	0.33	28.04	11.80	3.39

Table 4.3 shows the results from the experiments conducted in section 3.3.2 where the two feature-sets were set to 0, both separately then together. The results for each feature set being zeroed separately, particularly the visual features, are incredibly surprising as the model is still performing very well when

compared to just guessing (4.2). It's only when both feature sets are zeroed that the model shows the expected drop in performance. This shows that the model needs at least one of these feature sets to maintain performance, but the fact that the model can still perform better than guessing with the loss of one feature set suggests the model may be over-fitting to the data-set by finding an alternative mechanism to identify the data instead of correctly correlating vision and language.

4.1.4 Original Data-set distribution

Table 4.4: Table showing the distribution of the visual and word features in the base data

Visual Features Median of Means	0.000138249
Visual Features Median of Standard Deviations	0.04421956
Word Features Median of Means	-0.000802777
Word Features Median of Standard Deviations	0.22021659

Table 4.4 shows the outcomes from section 3.3.3. These results show that both feature sets are roughly distributed about the mean but that the word features have an order of magnitude higher standard deviation than the visual features. This suggests that the visual features will be more sensitive to the addition of noise in the magnitude of two decimal places than the word features. It also shows that the visual features are far less varied than the word features.

4.1.5 Noise results

Table 4.5: Table of results from experiments where features were set to random noise from a Gaussian distribution of mean 0 and standard deviation 1

Features replaced with noise	1@0.3	1@0.5	1@0.7	5@0.3	5@0.5	5@0.7
Visual Features	0.58	0.3	0.15	0.95	0.48	0.22
word Features	38.75	23.80	12.49	67.09	53.62	36.54
Both Word and Visual Features	0.19	0.09	0.03	0.42	0.20	0.05

Shown in table 4.5 are the results from the experiments detailed in section 3.3. These results are interesting due to the disparity between replacing the visiual and word features with noise. Replacing the word features didn't cause the results to drop too much but replacing the visual features with the same degree of noise to the visual features caused the models performance to plunge to nearly 0. This seems to indicate that the model bases the majority of its decision making on the visual features to the almost total exclusion of the word features. Figure 4.1 shows this more clearly. The small decline in performance with the replacement of the word features compared to the visual features corroborates evidence from 4.1.3 that the model is over-fitting to the data-set. The fact that the results from both experiments where the visual features are replaced fall far below the results reported in section 4.1.2 indicates that the model is performing artificially poorly as its results are worse than if it were to just randomly guess. The seeming reliance by the model on the visual features shown here also contradicts the findings in section 4.1.3 which suggested the model could function without the visual features. The fact that the model performs better than if it were to just guess when the visual features are replaced with noise again suggests that it doesn't factor them into its decision making nearly as much as it should be.

4.1.6 Adding noise to the feature sets

Tables 4.6 and 4.7 list the results from the experiments described in section 3.3.5. These results paint a clearer picture of the phenomena observed in section 4.1.5. Namely, that the model utilises the visual features far more than the word features. Figures 4.2 and 4.3 show this visually. If the model were using the visual features in the way it should then figure 4.3 should look like fig 4.2, i.e. performance would massively decline with an increase in noise. Instead the addition of allot of noise to the word features can be seen to have little overall effect on the model performance while the addition of even a little noise to the visual features causes a massive decline in performance. This shows that the model is not just over-fitting to the data (due to it maintianing performance when the word features have noise added) but that it is also neglecting the word features in its retrieval process.

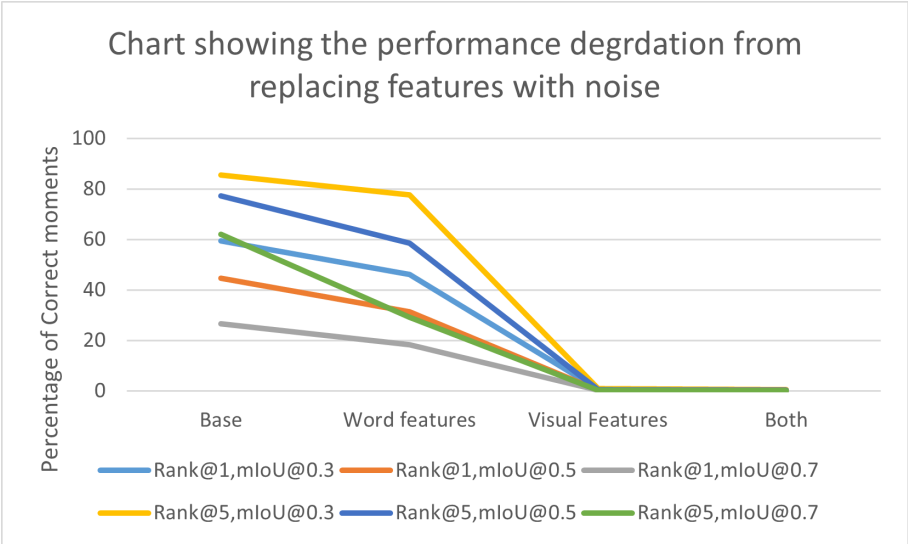

Figure 4.1: Chart to show the drop in model performance as the feature sets are replaced with noise

Table 4.6: Table showing the results of adding Gaussian noise to the visual features with a mean of 0 and a variable standard deviation

Standard Deviation	1@IoU0.3	1@IoU0.5	1@IoU0.7	5@IoU0.3	5@IoU0.5	5@IoU0.7
0.0005	59.4500	44.5000	26.5567	85.5433	77.1367	61.9700
0.005	59.4767	44.6567	26.7433	85.5333	77.0233	61.9333
0.05	45.9867	28.9133	13.8700	68.8333	54.2167	35.0433
0.07	23.6833	10.9700	4.3967	39.8833	24.7667	14.2333
0.1	5.7700	2.0233	0.7400	12.1767	5.3600	2.6367
0.5	0.4900	0.2200	0.0900	0.8933	0.4200	0.1400
1	0.6600	0.3167	0.1467	1.0333	0.4900	0.1967

Table 4.7: Table showing the results of adding Gaussian noise to the word features with a mean of 0 and a variable standard deviation

Standard Deviation	1@IoU0.3	1@IoU0.5	1@IoU0.7	5@IoU0.3	5@IoU0.5	5@IoU0.7
0.1	59.2367	44.2633	26.5433	85.3667	76.7100	61.6967
0.5	54.4000	39.0767	22.8300	82.4033	72.9300	57.2533
0.7	50.9733	35.7600	20.6500	79.8400	69.6567	53.5033
1	47.7767	32.4400	18.5800	76.4567	65.6233	49.4667

4.1.7 ActivityNet Captions Statistics

Table 4.8: Table showing the number of annotations in ActivityNet Captions that start at 0

	No. Annos	No. of Zeros	%
Train	37417	8156	27.49
Validate	17505	4081	21.80
Test	17031	4681	23.31
Total	71953	16918	23.51

Table 4.8 shows the results of analysis performed on the Activity Net Captions annotations described in section 3.10. It shows an alarming temporal bias towards the start of the video in the data-set captions with almost a quarter of all annotations starting at the beginning of the video.

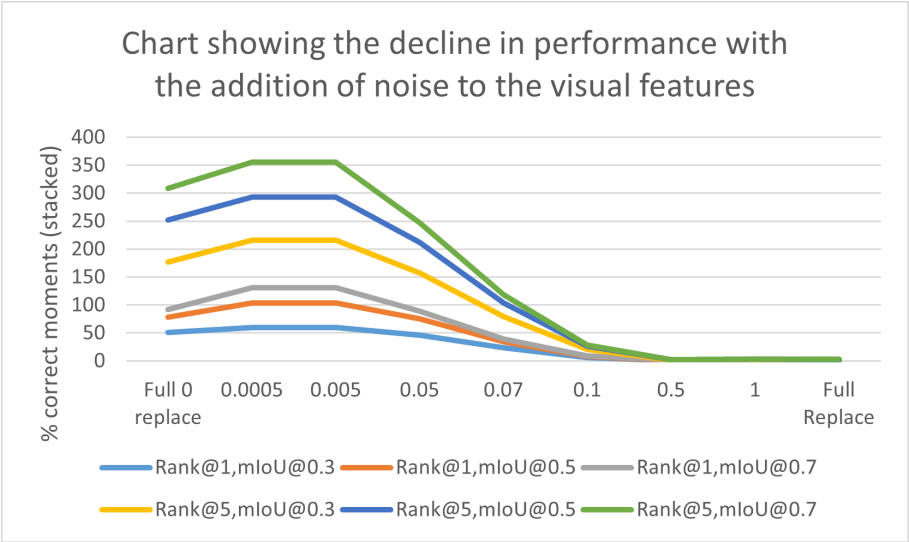

Figure 4.2: This chart shows the decline in model performance as the visual features are changed. Full zero replace is the case where the visual features were replaced with 0s (3.3.2) and Full replace is the case where the visual features were replaced with Gaussian noise (3.3.4). It is a stacked chart as this makes the difference in performance between categories of n@m clearer.

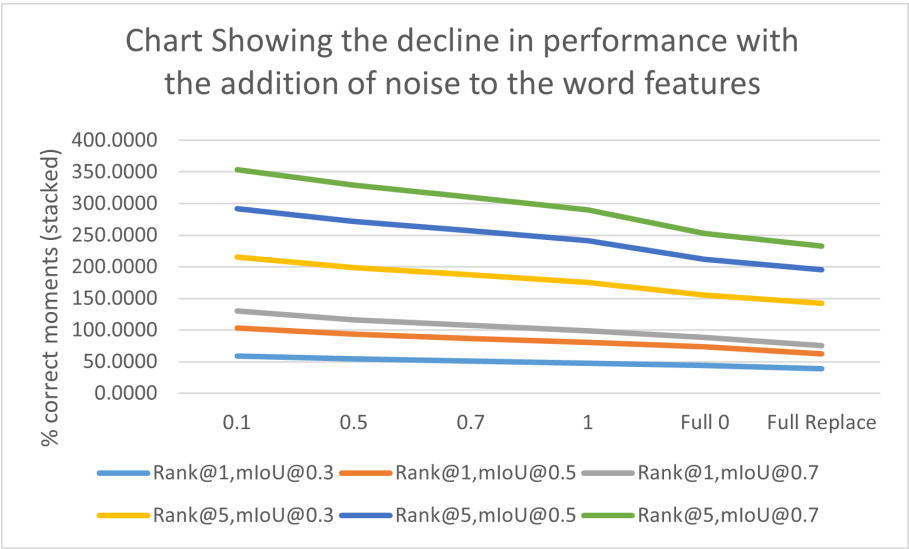

Figure 4.3: his chart shows the decline in model performance as the word features are changed. Full zero replace is the case where the word features were replaced with 0s (3.3.2) and Full replace is the case where the word features were replaced with Gaussian noise (3.3.4). It is a stacked chart as this makes the difference in performance between categories of n@m clearer.

Table 4.9: Table of results for the augmented data tests

Augmentation type	1@0.3	1@0.5	1@0.7	5@0.3	5@0.5	5@0.7
Un-augmented	50.69	39.58	24.31	76.39	65.28	51.39
Double speed	50.69	38.99	27.08	75.00	66.67	54.17
Concatenate	30.56	12.50	5.56	45.83	30.56	23.61
Again	26.39	6.94	1.39	47.22	26.39	15.97

4.2 Augmented results

Shown in table 4.9 are the results from the augmentation tests. These results are also shown in graph form in figure 4.4. Starting with the un-augmented data, we see a slight drop in performance compared

to the author reported results (4.1.1). This could be due to the size of the sampled data-set (3.6) and there not being enough data for the signal to come through. It could also be that the examples chosen for the sampled data-set happened to be difficult moments that the model struggles to retrieve. However, these results are not significantly different from those reported by the authors, even given that a different set of C3D checkpoints were used to generate them (3.7). The results from the double speed test are interesting as, while they are almost identical to the un-augmented results, in some of the n@m categories the model actually outperforms against the un-augmented data-set. It even does so on the most stringent category, 1@0.7, gaining roughly 3% over the un-augmented results. Rank 5@0.7 similarly gained 3% over its un-augmented counter-part. The expected outcome was either similar performance to the the un-augmented data-set or a drop in performance. This outcome could be due to the fact that, from the model's perspective, the effect that doubling the speed has is just an increase in sampling. Bearing in mind that the 2D-TAN model samples N clips of T frames (2.4) so an increase in video speed of times 2 effectively doubles this sampling rate. The fact that this increases performance may indicate that the model is more performant with either a higher T value or a lower N value or both. The results from the Concatenate test show a significant drop in performance to even below that of if it were randomly guessing (4.1.2) in most n@m categories (1@o.5, 5@0.3, 5@0.5,5@0.7). The reason for this drop in performance is, again, likely over-fitting. The model has learnt an optimised strategy for performing well in the n@m tests, not to actually correlate language and video. The last augmentation test, the again test, shows a drop in performance when compared to the concatenate test. As mentioned above this is likely due to the model over-fitting to the data-set.

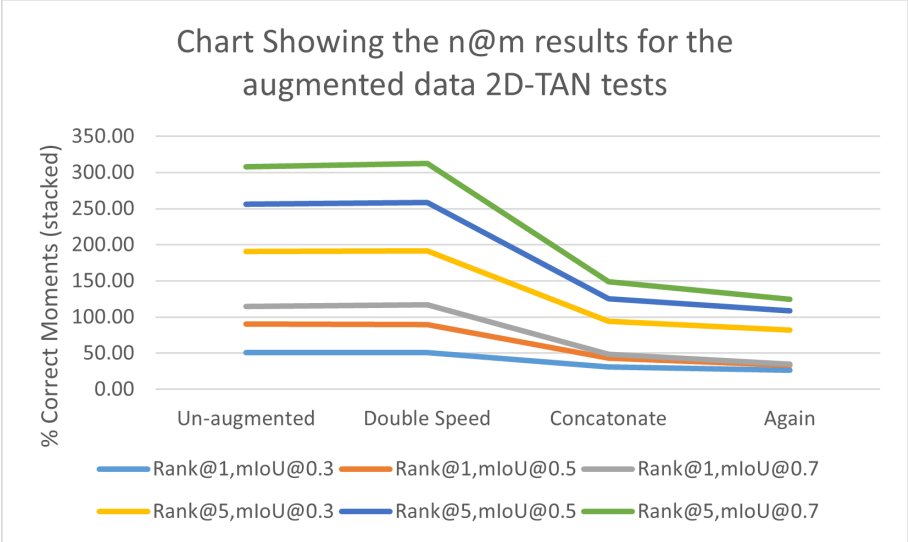

Figure 4.4: Chart shows the results from the augmented data tests (stacked)

Table 4.10: Table showing statistics of the retrieved moments compared to the actual annotations

	No. of Zeros	%	Actual No. of Zeros	%	No. Repeats	%
Un-augmented	61	43.36	36	25.17	43	30.07
Double Speed	56	39.16	36	25.17	33	23.08
Concatenate	45	31.47	0	0	45	31.47
Again	81	56.64	0	0	68	47.55

Table 4.10 shows statistics of the moments retrieved by the model for each of the augmentation tests. It shows the number of moments retrieved that start at 0 as well as the actual number of annotations that start at 0 for each test-set. It also shows the number of identical moments that were retrieved for the same video but under different annotations. The percentages for each of these metrics are also shown. The actual number of annotations starting at 0 in the sampled data-set is slightly higher than the original data-set, as reported in section 4.1.7 meaning the sampled data-set is slightly more temporally biased than the original. The number of retrieved moments starting at 0 is significantly higher than the actual number of annotations starting at 0 across all tests. This is the clearest evidence yet for the model's over-fitting to the bias present in the data-set (4.1.7). The number of retrieved moments starting at 0

significantly drops between the un-augmented and Concatenate tests. This suggests that the model is somewhat correlating vision and language as it was able to realize, in more cases, that the annotations didn't start at 0. However, it still believed that roughly 31% of annotations started at 0 when they did not. Over half of the moments retrieved by the again test started at 0. This is a mixed result as it suggests that the model is correlating the vision and word features, its just not able to fully comprehend the entirety of the natural language semantics, namely the meaning of 'again'. The model is likely comprehending the broad semantics of the natural language query and associating it with the first time it occurs: in the first iteration of the video. But even if this is the case, its number of retrieved moments starting at 0 is over double the number it retrieved on the un-augmented set, again demonstrating its bias. I discuss the fairness of this test in section 4.5.4 as there are some edge cases where this test unfairly penalises the model. The number of identical moments retrieved for different language queries in the same video also corroborates the evidence in sections 4.1.6 and 4.1.5 which suggests that the model doesn't use the word features to retrieve moments as much as it uses the visual features. These results show that in the un-augmented test, the model is essentially ignoring the language query 30% of the time. This drops to 23% for the double speed test. This, with the decreased number of 0 start annotations retrieved, shows that the model increases in performance with an increase in sampling. The again test is shown to ignore the language query in almost 50% of cases, contradicting the inference made above that it may be correctly correlating parts of the natural language query. Further analysis showed that, for the again test, 37.06% of total retrieved moments both ignored the natural language query and started at 0. This again demonstrates a high level of over-fitting and temporal bias.

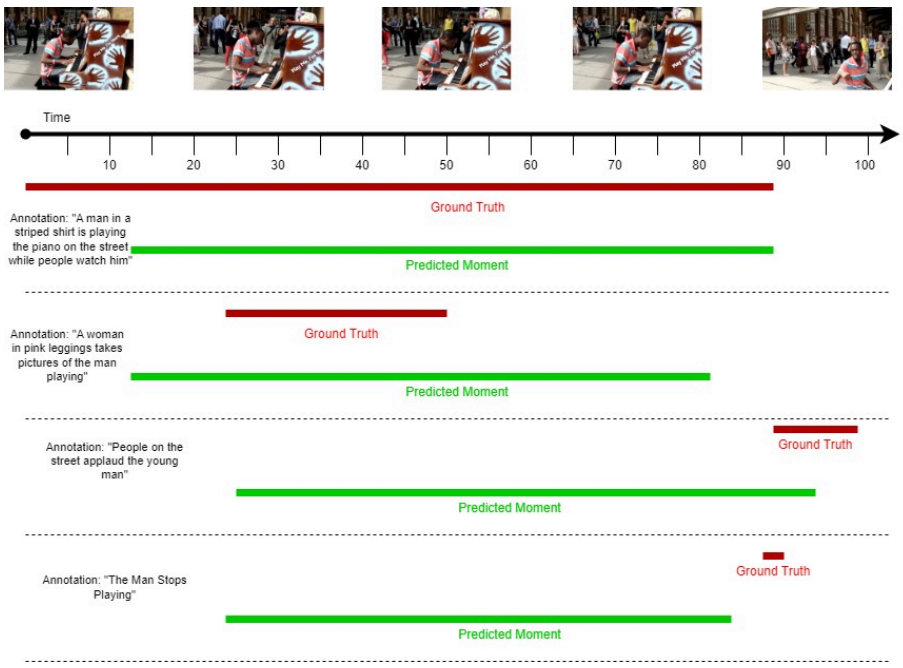

Figure 4.5: This diagram shows the ground truth vs retrieved moments for the video 'Pi3' from the un-augmented test.

4.3 Findings

The thorough analysis of the experiment results in sections 4.2 and 4.1 have laid bare many of the inadequacies of 2D-TAN. Specifically its bias, over-fitting, neglect of the word features and inability to correlate across time. Further more, the fundamental structure of the 2D-temporal adjacency map (2.4) means that 2D-TAN may hold biases that can't be unlearnt (4.3.5).

4.3.1 Over-fitting

The strongest conclusion that can be drawn from my investigation is that 2D-TAN is over-fitting to Activity Net Captions. This is evidenced in almost every-test that was conducted. In the initial zero (4.1.3) test the model was still able to function better than if it were just guessing with the loss of one of

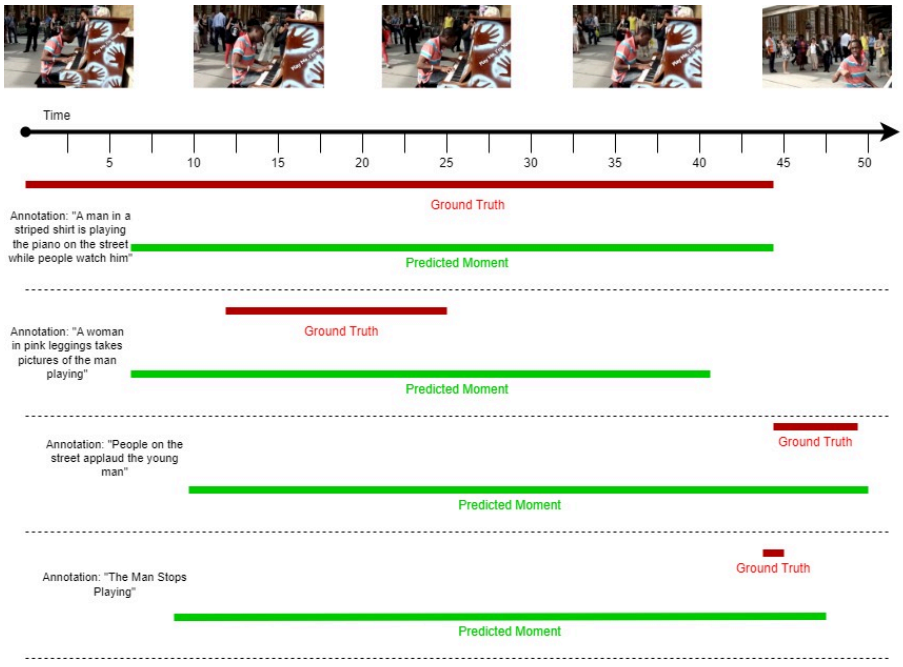

Figure 4.6: This diagram shows the ground truth vs retrieved moments for the augmented video 'Pi3X2' from the augmented double speed test. The differences are incredibly subtle with the model predicting almost exactly the same moments. However, for the latter two annotations it predicted slightly broader moments

the feature sets. In the noise tests it could also still function well when the word features were replaced with noise (4.1.6). This shows that the model has learnt the trends in the data-set so strongly that it can still perform well even with the loss of certain features because it has learnt other ways to identify the moments. In the augment tests (4.2) the model was shown to be disproportionately retrieving moments that started at the beginning of the video. This is due to it over-fitting to the bias present in the data-set.

4.3.2 Temporal Bias

The second most prominent finding is 2D-TAN's temporal bias towards retrieving moments that start at the beginning of the video (index 0). This is most clearly shown in section 4.2 where the percentage of retrieved moments that start at zero was shown to be at 30% for the un-augmented sample-set. This is higher, even, than the temporal bias within the data-set its self which was shown to be 23% in section 4.1.7. Granted, the sampled-set was slightly more biased than the original set (25% in 4.2) but still significantly below the temporal bias exhibited by the model. This temporal bias is further exhibited in the concatenation test's poor performance. The model should have exhibited performance on par with the un-augmented test during this experiment but instead it significantly dropped in performance. This is again, likely due to the model's temporal bias towards moments earlier in the video.

4.3.3 Use of natural language

It was also shown, across multiple tests, that 2D-TAN is not using the features from the natural language query as much as it should be, likely due to over-fitting. In the noise replacement test 4.1.5 it was observed that replacing the word-features with noise had a far lesser effect on performance than doing the same for the visual-features. This comparison was made clearer by the results in section 4.1.6 which showed that adding noise to the word-features only caused a linear decrease in performance whereas doing the same for the visual-features caused an exponential decline in performance. It would be expected that if the model was properly correlating vision and language that a similar change in either of them would impact performance in the same way. Finally, analysis of the best retrieved moments during the augmentation tests showed that the model was often retrieving the same moment for different queries in the same video, meaning it's ignoring the query and only using the moment candidates and visual features to retrieve the moments. This is likely a by-product of the model's over-fitting to the data-set, it has likely found a way

to optimise the intersection over union against the data-set ground truth without the need for the word features.

4.3.4 Cross time correlation

In their paper on 2D-TAN [34] the authors claim that 2D-TAN is able to correlate moments across time, such as "a man plays a saxophone, *again*". The poor performance during the again experiment (4.2), specifically designed to test this, shows that this is not the case. It also further highlights the model's inability to correctly correlate vision and language.

4.3.5 Bias against Longer Moments

None of the issues discussed thus far are specific to 2D-TAN, over-fitting, data-set bias and a failure to use all given data are problems that could afflict any machine learning model, yet alone temproal grounding models specifically. However, during the investigation a particular flaw inherent to 2D-TAN was identified but not tested. By looking at the 2D-Temporal adjacency map (2.4) and more specifically the sparse sampling technique employed (2.4.2) it should be clear that 2D-TAN has a bias against longer moments (longer relative to the size of the video as the sparse sampling is conducted in proportion to the duration of the video). The reasons for this are two fold. First, due to the sparse sampling technique (2.4.2) employed, as the moment candidates get longer they're sampled exponentially less frequently, figure 4.8 shows this. The sparse sampling schema also means that longer moment candidates become isolated. In figure 4.7 it can be seen that the longer moment candidates are surrounded by grey unsampled candidates. These unsampled candidates then act as a valley in the weight space during the convolution phase as they have no initial weighting as their features aren't sampled. This means the longer a moment is, the more isolated it is so the less it shares it's weights with other moments or has other weights shared with it. Longer moments are also sampled far less frequently than shorter moments, meaning that shorter moments will be retrieved far more frequently than longer ones as there is more of them. While demonstrating this long moment bias via tests is beyond the scope of this project it is highlighted here as potential future work.

4.4 The case for Temporal augmentation

In the computer vision task of classification existing training data is augmented to produce new training data (section 2.7). This improves model performance as it prevents the trained model from over-fitting, increasing its generalisation. Therefore the over-fitting exhibited by 2D-TAN (4.3.1) could be alleviated by performing augmentations to the training data-set; Augmentations such as those discussed in 3.4. The Concatonate augmentation could also over-come the temporal bias in the Activity Net Captions data-set, identified in [33] and corroborated in by my own investigation (4.3.2), by autonomously producing many test samples with annotations that don't start at the beginning of the video. What's more, the problems of bias and over-fitting shown by 2D-TAN in the performed experiments are not due to its structure, meaning it could be retrained with augmented data to over-come them. This also means that other Temporal Grounding models trained on Activity Net Captions likely exhibit the same temporal bias. The concatenate test has the potential to increase the number of training examples exponentially by repeatedly joining videos together to create new longer videos. It is also likely that Temporal Grounding models other than 2D-TAN may be over-fitting to their data-sets if temporal augmentations have not been used in the training process; as 2D-TAN was demonstrated to be doing through-out this work. These augmentations could be used as common practice by future temporal grounding models to overcome temporal bias and over-fitting.

4.5 Justification of methodology

As mentioned in many of the previous sections, specific choices were made in regards to the use of certain programs and libraries over others which may have impacted the quality of my final results. These as well as time and space saving measures may have impacted the viability of my results. In this section I look the largest of these decisions and analyse their possible effect on the experiment results.

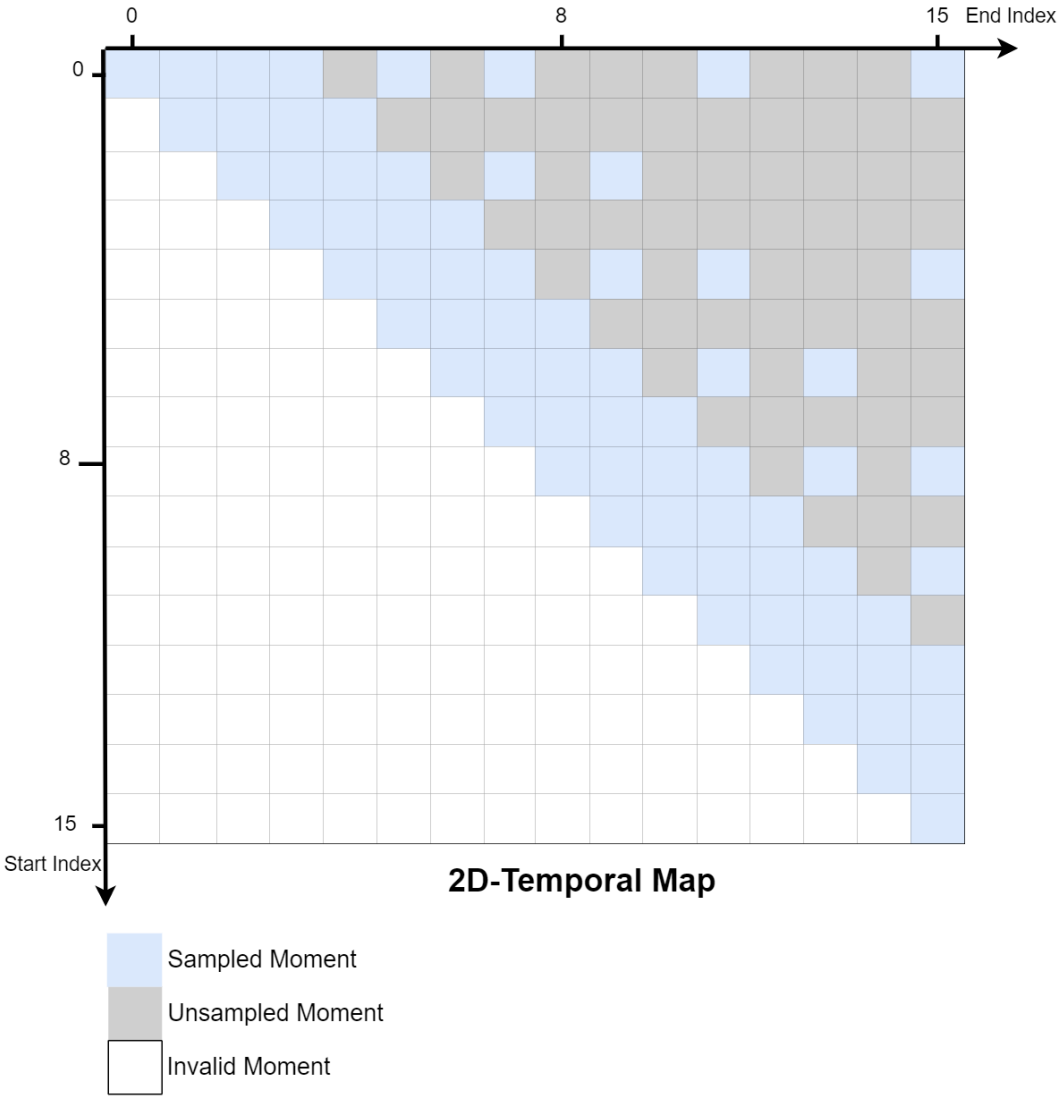

Figure 4.7: Figure shows a smaller version of the same 2D-Temporal adjacency map from Fig 2.9. In this version the N value (number of sampled clips) is just 16.

4.5.1 Choice of model and data-set

The first decision made was that to use 2D-TAN. 2D-TAN was chosen as the model to test due to its influence in the field of temporal grounding with over 230 citations as of writing. Its use of a 2-dimensional temporal adjacency map (2.4) also makes it easier to reason about how it's functioning, instead of just being a complete black box.

Activity Net Captions (2.5.1) was chosen as the data-set to trial the augmentation system as it is the largest data-set 2D-TAN was trained on meaning it should have the strongest signal. It was also the data-set that the authors of 2D-TAN used for their ablation study meaning 2D-TAN was better optimised to operate on Activity Net Captions. These factors combined mean that 2D-TAN should perform best on Activity Net Captions making any drop in performance due to augmentation more significant.

4.5.2 Data Sampling

The choice to hand pick data likely introduced some bias into the system. In section 4.2 it is shown that the sampled data-set has a 2% higher bias towards annotations starting at the beginning of the video than the original data-set. In future, to prevent human bias I would randomly sample the data-set. This could be done, while also minimising space usage, by first integrating the augmenter into the feature

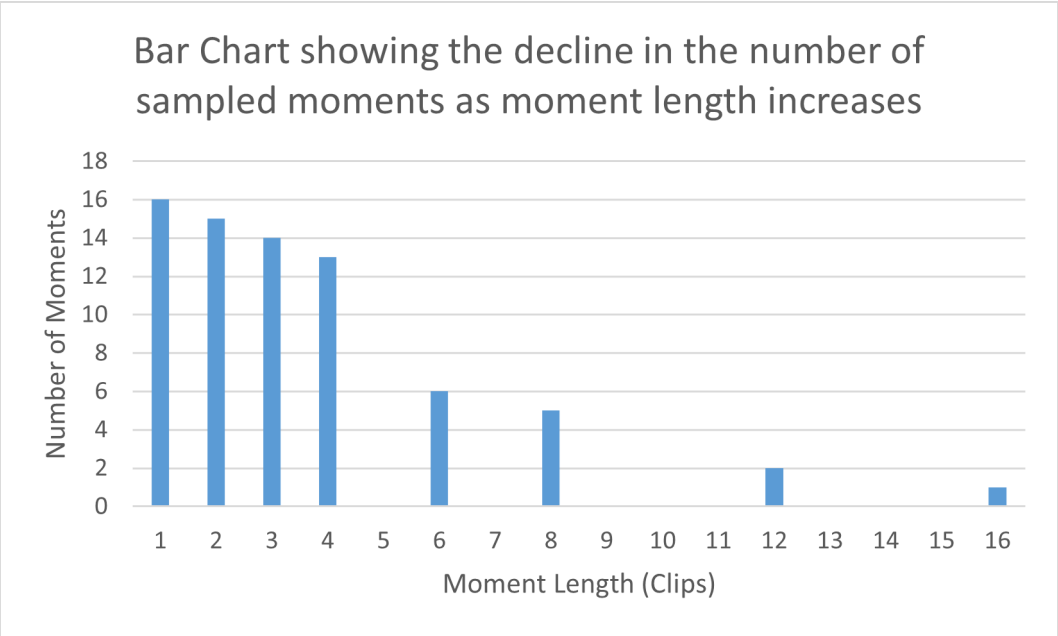

Figure 4.8: Figure shows the number of sampled moments against moment length for the example in figure 4.7

extractor. This could then be linked to yt-dlp to download videos directly from Youtube, extract their frames, and then delete them. The videos could be indexed via the original annotations json file in much the same way that the list of sampled annotations was used to index the sampled videos (3.5).

4.5.3 Use of C3D

As previously mentioned in section 3.7, retraining 2D-TAN on features generated from the entire Activity Net Captions data-set using new C3D checkpoints wasn't viable given the time and space constraints of the project. The use of C3D checkpoints that weren't the ones originally used to generate the features 2D-TAN was trained on to produce the features for the augmented videos adds some unfairness to the results. Comparing the performance of 2D-TAN on the unaugmented sampled data-set videos' features (Table 4.9) with the authors results (Table 4.1) shows a dip in performance but this could also be due to the fact that only a small subset of the data was used and therefore the signal in the data didn't come through as clearly. It could also be because the samples chosen happen to be more difficult for the model to understand. This could of course be remedied, given more space and time, by retraining 2D-TAN with features from the new C3D network checkpoints. The disparity between the number of features out-put by the new feature extractor and the original features size (487 against 500) meant the last 13 features had to be padded with 0s. It is is possible this affected the performance of the model but in a limited capacity as it was shown in section 4.1.3 that even when the visual features were set to zero the model still performed moderately well.

4.5.4 Annotation fairness

There is also an argument to be made as to the fairness of the concatenation and again tests (3.9) as the data is very artificial and unlikely to be found in the wild. However, artificial augments are used in object classification and have been shown to improve the performance of classifiers ([14],[4],[31]). The 'again' test (3.9) may also be unfair as the original video may have a point described by one of the augmented annotations and so the model will, correctly, associate the query with a moment in the first iteration of the original video where there's no ground truth to validate it. Figure 4.9 shows this more clearly. It also demonstrates another edge case in annotation 2a where the word 'again' is added to the end of a sentence that already ends with the word again. However, the annotations in Activity Net Captions are so unique and contextual it is practically impossible for this kind of scenario to occur. But if this framework were to be extended to other data-sets with simpler captions such as 'Something something' [7] where this edge case may be more likely, the annotation editor would have to be modified to prevent this.

4.5.5 Stochastic tests

The initial experiments on 2D-TAN that utilised noise were performed 3 times and the average over the results taken to improve validity. However, three is still a lower number of trials. In order to improve confidence these tests should, in future, be performed several more times. They weren't in this project due to time constraints. It is also due to both time, and space constraints on BC4, that the stochastic 'Concatenate' augment was only performed once. It is therefore possible that the results for the Concatenate augment (4.2) are an outlier. However, the trends shown in the Concatenate test results are consistent with the non-stochastic test results. This uncertainty could of course be alleviated by performing the Concatenate test several more times or by removing its stochastic nature entirely by performing the augment between every possible video pair. However, this would use an immense amount of space as, even on a limited data-set of 43 videos, this would generate 1806 augmented videos. But it is also possible that the model could learn and over-fit to this as well.

Figure 4.9: Diagram highlights a possible unfair edge case in the again test. Here the augmented annotation 1a is a copy of the original annotation 2. The model then may correctly correlate the annotation for 1a to the moment for 2 but this wouldn't be counted as correct because the ground truth for 2 is not included as part of the test.

Chapter 5

Conclusion

5.1 Summary of Achievements

Over the course of this project I was able to successfully identify several weaknesses in 2D-TAN. Namely its bias, inherent from the data-set Activity Net Captions which it was trained on (4.3.2), its over-fitting (4.3.1) to Activity Net Captions, and its inability to correctly correlate moments across time (4.3.4). In order to gather evidence, a series of experiments were carried out. The experiments conducted with the original data-set, where the features were replaced and noise added, showed that the model was over-fitting to the data-set (4.3.1) A data-augmentation and testing pipeline was then constructed to test the model's performance on data that had been temporally augmented. With the use of this pipeline it was further shown that the model was incredibly temporally biased towards moments that start at the beginning of the video (4.3.2) and that the model could not properly correlate moments across time (4.3.4) as the authors had claimed it could in [34]. A bias against longer moment candidates due to the sparse sampling schema used by 2D-TAN was identified (4.3.5) but not tested due to it being out of scope of this project as doing so would have likely required building a new data-set of long annotations and retraining 2D-TAN on it. Finally the case was made for the use of temporal augmentations (4.4) as a means to eliminate temporal bias as well as improve the generalisation of temporal grounding models.

5.2 Status

The project achieved its core goal of identifying and proving weaknesses within 2D-TAN (4.3), however not all weaknesses were tested. One such possible weakness that wasn't thoroughly investigated is the bias that the sparse sampling technique (2.4.2) may have against longer moments (4.3.5). The code quality could also be improved as while the code-base for the pipeline (3.5) is functional it is not written to a professional standard. There are also many more avenues of investigation stemming from this work detailed in section 5.3.

5.3 Future Work

This project has identified an immense amount of potential future work in not only investigating 2D-TAN but also the borader field of Temporal augmentation. Within 2D-TAN there is the suspected 'long moment bias' identified in 4.3.5 that needs to be verified. This is a bias against longer moments inherent to 2D-TANs sparse sampling schema. The investigation also focused solely on the Activity Net Captions data-set when 2D-TAN was trained on two other data-sets, TACoS [23] and Charades STA [5]. Charades STA is shown in [33] to also have a significant temporal bias meaning that the 2D-TAN checkpoints trained on Charades STA are likely also biased. 2D-TAN may also be over-fitting to both TACoS and Charades STA as no augmentation was used by 2D-TAN during its training on any of the data-sets. The investigation also only focused on 2D-TAN when there are many other temporal grounding models in the field, [15] which, if they're trained on Activity Net Captions and Charades STA, are also likely very temporally biased [33]. These models may also be over-fitting to their data-sets if they are not utilising augmentations during training. A further investigation into the use of temporal augmentations for training temporal grounding models may also show that their use eliminates temporal bias in trained

models as well as making models trained on them generalise better (4.4). This could provide a significant performance boost to the entire field of Temporal Grounding.

Bibliography

[1] al. Yolov5. URL: https://github.com/ultralytics/yolov5.

[2] François Bérard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *IEEE workshop on Projector-Camera Systems*, 2003.

[3] Shyamal Buch, Victor Escorcia, Bernard Ghanem, Li Fei-Fei, and Juan Carlos Niebles. End-to-end, single-stream temporal action detection in untrimmed videos. *British Machine Vision Association and Society for Pattern Recognition*, 2019.

[4] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020.

[5] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *Proceedings of the IEEE international conference on computer vision*, pages 5267–5275, 2017.

[6] Yanjun Gao1, Jason Wang Lulu Liu, Huayan Wang Xin Chen, and Rui Zhang. Evoquer: Enhancing temporal grounding with video-pivoted backquery generation. *arXiv*, 2021.

[7] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[8] Ashley Gregg and Zac Woodford. Cave notes: Co.clare, ireland. In *Proceedings of the University of Bristol SpelÆological society, Volume 28, Number 3*, pages 341–346, 2021.

[9] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *2015 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 961–970. IEEE, 2015.

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

[11] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, page 5, 2014.

[12] Andrej Karpathy, Sanketh Shetty George Toderici, Rahul Sukthankar Thomas Leung, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014*, page N/A, 2014.

[13] Cherry Khosla and Baljit Singh Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 79–85. IEEE, 2020.

[14] Cherry Khosla and Baljit Singh Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 79–85. IEEE, 2020.

[15] Xiaohan Lan, Yitian Yuan, Xin Wang, Zhi Wang, and Wenwu Zhu. A survey on temporal sentence grounding in videos. *none*, 19(2), feb 2023. doi:10.1145/3532626.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[17] Juncheng Li, Junlin Xie, Long Qian, Linchao Zhu, Siliang Tang, Fei Wu, Yi Yang, Yueting Zhuang, and Xin Eric Wang. Compositional temporal grounding with structured variational cross-graph correspondence learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3032–3041, 2022.

[18] Alex Mackin, Fan Zhang, and David R Bull. A study of high frame rate video formats. *IEEE Transactions on Multimedia*, 21(6):1499–1512, 2018.

[19] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[20] Will Price and Dima Damen. Retro-actions: Learning'close'by time-reversing'open'videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[21] Abhisek Ray, Maheshkumar H Kolekar, R Balasubramanian, and Adel Hafiane. Transfer learning enhanced vision-based human activity recognition: A decade-long analysis. *International Journal of Information Management Data Insights*, 3(1):100142, 2023.

[22] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[23] Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. Script data for attribute-based recognition of composite activities. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part I 12*, pages 144–157. Springer, 2012.

[24] F Rosenblatt. Cognitive systems research program. *Scientific and Technical Information*, 1963.

[25] George Stockman and Linda G. Sahpiro. *Computer Vision*. Prentice Hall PTR, 2001.

[26] 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.

[27] Rohit Verma and Jahid Ali. A comparative study of various types of image noise and efficient noise removal techniques. *International Journal of advanced research in computer science and software engineering*, 3(10), 2013.

[28] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. Bidirectional attentive fusion with context gating for dense video captioning. *CoRR*, abs/1804.00100, 2018. URL: http://arxiv.org/abs/1804.00100, arXiv:1804.00100.

[29] Mei Wang and Weihong Deng. Deep face recognition: A survey. *Neurocomputing*, 429:215–244, 2021.

[30] wyzowl. Youtube stats: Everything you need to know in 2023! URL: https://www.wyzowl.com/youtube-stats/#:~:text=Around%203.7m%20new%20videos,average%20length%20of%204.4%20minutes.

[31] Mingle Xu, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, page 109347, 2023.

[32] G Hinton Y LeCun, D Touresky. A theoretical framework for back-propagation. *Proceedings of the 1988 Connectionist Models Summer School*, 1988.

[33] Yitian Yuan, Xiaohan Lan, Xin Wang, Long Chen, Zhi Wang, and Wenwu Zhu. A closer look at temporal sentence grounding in videos: Dataset and metric. In *Proceedings of the 2nd International Workshop on Human-centric Multimedia Analysis*, pages 13–21, 2021.

[34] Songyang Zhang, Jianlong Fu Houwen Peng, and Jiebo Luo. Learning 2d temporal adjacent networks for moment localization with natural language. *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[35] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 566–583. Springer, 2020.

Appendix A

Appendix

A.1 Example of a json annotation


```
{
    "A1": {
        "duration": 196.09,
        "timestamps": [
            [
                3.92,
                15.69
            ],
            [
                16.67,
                176.48
            ],
            [
                98.05,
                112.75
            ],
            [
                177.46,
                180.41
            ]
        ],
        "sentences": [
            "A man talks to the audience while seated on stage before a performance.",
            " A man plays the keys of an accordion while opening and closing the air box.",
            " The man sings a few verses loudly with excitement.",
            " The man finishes the song and gathers up his notes from the podium."
        ]
    },
```


Figure A.1: Figure shows an example of a video entry in the json annotations file

A.2 Example of a csv output file

```
vid,sentence,bestGuess,actual
['A1'],['A man talks to the audience while seated on stage before a performance.'],"[0.0, 196.08999633789062]","[[3.92, 15.69]]"
['A1'],[' A man plays the keys of an accordion while opening and closing the air box.'],"[0.0, 196.08999633789062]","[[16.67, 176.48]]"
['A1'],[' The man sings a few verses loudly with excitement.'],"[0.0, 196.08999633789062]","[[98.05, 112.75]]"
['A1'],[' The man finishes the song and gathers up his notes from the podium.'],"[134.81187438964844, 196.08999633789062]","[[177.46, 180.41]]"
['A2'],['A man in a movie mask is seated by a flight of stairs.'],"[0.0, 59.75]","[[0, 10.75]]"
['A2'],[' He is playing an accordian for the people passing by.'],"[7.46875, 52.28125]","[[11.05, 48.39]]"
['A2'],"[' A few pause to listen, then continue on their way.']","[35.4765625, 59.75]","[[49.59, 59.75]]"
['A3'],['A man wearing glasses is seen stepping into frame and playing an accordion.'],"[0.0, 33.592498779296875]","[[0, 38.52]]"
['A3'],[' The man continues to play while looking off into the distance and stops by smiling to the camera.'],"[27.993751525878906, 89.58000183105469]","[[25.98, 89.58]]"
['B1'],['A lady stands in a dance studio.'],"[0.0, 170.33999633789062]","[[0, 13.63]]"
['B1'],[' the lady performs a ballet dance.'],"[21.292499542236328, 149.04750061035156]","[[13.63, 170.34]]"
['B1'],[' The lady lifts her leg up behind her and moves it to the front and rests it on her leg.'],"[85.16999816894531, 165.01687622070312]","[[53.66, 68.14]]"
['B1'],[' The lady stands in front of the camera facing the mirror.'],"[31.938749313354492, 170.33999633789062]","[[103.91, 114.98]]"
['B1'],[' We see a person sitting on the left side of the room.'],"[0.0, 170.33999633789062]","[[168.64, 170.34]]"
['B2'],['There are a lot of dressed up people sitting in a large dining area at a reception.'],"[0.0, 35.029685974121094]","[[0, 8.97]]"
```

Figure A.2: Figure shows an example of a results file produced by the modified 2D-TAN