UNIVERSITY OF EXETER

Computer Science Department

B.Sc. Computer Science
Computer Science Department

What happens next? LSTM and Transformer based human action anticipation methods

Candidate

Lucas Proudhon-Smith

Student ID 700074221

Supervisor

Dr. Sareh Rowlands

University of Exeter

Academic Year
2022/2023

Abstract

Over the past decade, the adoption of computer vision has surged across diverse sectors, from manufacturing to surveillance, this technology has slowly crept into almost every aspect of our lives. However, most current models focus on analysing and finding patterns in past and current data, so a logical next step is for them to attempt to learn from past patterns to predict what might happen in the future, by emulating a human's sense of anticipation.

This task is called Action anticipation, and models come in many forms - Recurrent Neural Networks(RNNs), Long short-term memory (LSTMs), Gated Recurrent Units (GRUs) and, most recently, Transformer based models. It is widely accepted that Transformer-based models perform the best for action anticipation, whereas LSTM models can produce results that are almost as good while requiring less computational power and time.

This paper aims to analyse the strengths and weaknesses of LSTM and Transformer models in the context of action anticipation in videos. We also analyse two existing popular models for action anticipation using egocentric and third-person video datasets; then attempt to improve upon these methods.

	Yes	No
I certify that all material in this dissertation which is not my own work has been identified.	☑	☐
I give the permission to the Department of Computer Science of the University of Exeter to include this manuscript in the institutional repository, exclusively for academic purposes.	☑	☐

Contents

1

Introduction and Project Specification

Advances in computer vision have made it possible for a machine to understand the objects in an image or video. This information can then be used to infer contexts such as human actions like exercising or sitting in a chair or environmental context such as the current weather or where the scene is taking place.

Recognising the context of our surroundings and anticipating what might happen next and how best to act on that information is something that humans unconsciously do, with many recent experiments aiming to emulate these human abilities through computer vision, action recognition, and anticipation.

Computer vision is a technology that aims to make computational models that simulate the human visual system that became prevalent in research in the late 1960s when researchers aimed to create the tools needed to produce intelligent robot behaviour. [1] More recently, the field has progressed to a point where deep learning techniques have become the dominant approach for almost all recognition and detection tasks. [2]

Human Action Anticipation uses general context learned using computer vision, and observed behaviours of motion, objects and other details in a video to anticipate what might happen next. This can be crucial for developing intelligent systems that have the ability to efficiently and effectively interact with their task and environment by recognising actions before they are completed, or even before they are started. The many potential applications of Action Anticipation include:

- Human-Robot Interaction: A robot capable of Action Anticipation may use its predictions of what might happen next to enable them to make proactive decisions about how they can help a human without being explicitly told what to do. For example, if a robot predicts that a human is about to leave their home, it might want to tell them the weather and their commute time. [3]
- Surveillance Systems: Anticipation can improve the chances of catching a crime on video by adjusting camera angles to where a crime might happen; it can also reduce the risk caused by criminal actions by giving police information on potentially suspicious behaviour [4]
- Autonomous Vehicles: Anticipation allows a vehicle to efficiently and accurately predict what other road users' intentions might be so that it can proactively prepare for likely scenarios. [5]
- Gaming and virtual reality: In these applications, anticipation allows for a more interactive and realistic experience by adapting environments or experiences based on predicted user actions. [6]

Modalities in the context of action anticipation refer to the different types of data that can be input into a model to represent information, with each modality containing different perspectives or representations of what is happening in a video. The most commonly used modalities are RGB and Optical Flow. The RGB modality captures the standard visual appearance of scenes, objects, and actions. Optical Flow can be extracted from RGB videos to capture information about the movement of objects within a scene which can result from the motion of objects or the camera angle [7]. Another

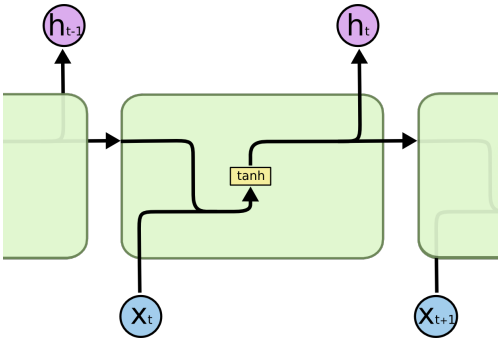

Figure 1.1: RNN architecture. RNNs are made up of a chain of repeating modules of a simple neural network, in this example, containing a single Hyperbolic Tangent Function (Tanh) layer - used as an activation function that squashes an input number into a value between -1 and 1. Diagram from [11]

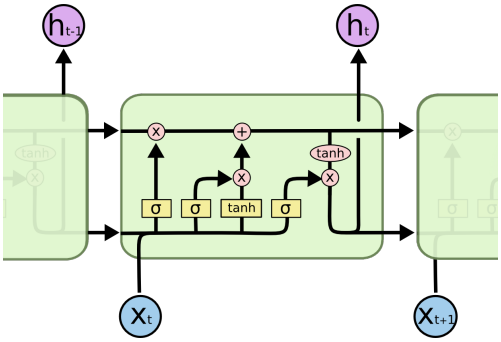

Figure 1.2: The LSTM architecture features a cell state running along the top of each module, enabling data addition or removal, providing LSTMs with their memory. Each layer (represented by yellow boxes for sigmoid and tanh) determines the data amount passed to the cell state. Diagram from [11].

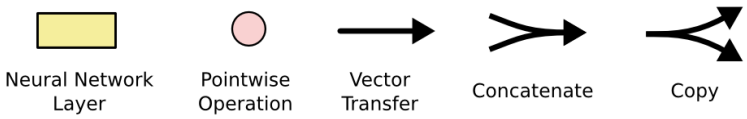

Figure 1.3: Legend for the notation used in the above diagrams. Diagram from [11]

commonly used modality is Semantic Annotations which can include descriptions of actions and objects within a scene. For example, object detectors can be used to generate object labels for items within a scene. These have been proven to improve accuracy in action anticipation and recognition tasks [8]. Other less used modalities include Depth - distance of objects from the camera, Gaze - where a person is looking, Audio, and Skeletal Data.

There are two leading technologies behind almost all recent action anticipation model implementations - RNN-based approaches including LSTM [8] and GRU [9] models, and Transformer [10] based approaches. It is widely accepted that RNN-based approaches are more efficient to train and use, whereas Transformer approaches produce more accurate Action Anticipation results.

Recurrent Neural Networks (RNNs) are Neural Networks capable of modelling temporal dependencies in sequential data. Long Short-Term Memory (LSTM) Networks build upon RNNs specialising in the ability to capture long-term dependencies in data sequences. They utilise gates including input, forget and output to regulate information flow through the network. This allows them to retain or forget information based on its relevance. Simplified diagrams of RNN and LSTM architectures can be seen in figure 1.1 and 1.2.

Transformers are a more recent model than LSTMs, introduced in the paper "Attention Is All You Need" by Vaswani et al. in 2017 [12]. They were designed to follow on from recurrent and convolutional neural networks by basing their architecture entirely on attention mechanisms, which enable them to weigh the importance of different parts of input data. Since their introduction, they have become popular in tasks such as Natural Language Processing due to their superior performance.

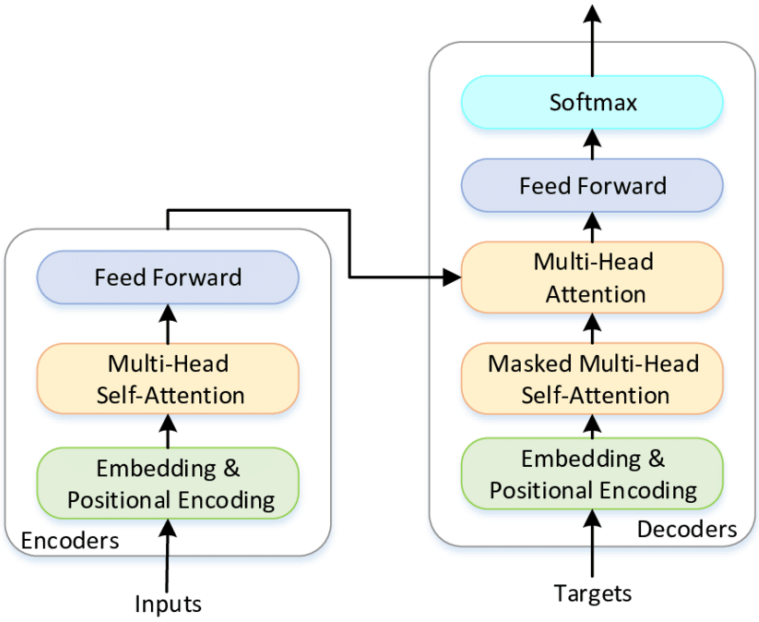

Figure 1.4: A simplified diagram of the transformer model, transformers vary in specific structure, however they all consist of an Encoder and a Decoder which use attention mechanisms. Diagram from [13].

Transformers consist of an encoder and a decoder, each composed of a stack of layers. The encoder maps an input sequence to a sequence of continuous representations, and the decoder generates an output sequence of symbols. In the context of action anticipation, this could be an action label.

Attention mechanisms allow a model to focus only on the most important pasts of input data, which in the context of action anticipation could be specific frames or segments of a video . These allow the model to focus more on data that is useful, leading to more accurate predictions [12]. A simplified diagram of the Transformer model architecture can be seen in figure 1.4.

This project focuses on researching the strengths of LSTM-based and Transformer-based models, and how the underlying technology within the architecture of implementations affects performance and accuracy in the task of Action Anticipation.

1.1 Project Specification

The motivation is to research what exactly makes LSTM approaches more efficient, what exactly makes Transformer approaches more accurate. Previous implementations also haven't attempted to benchmark how egocentric action anticipation models perform while using third-person datasets.

The aim of this project is to understand commonly used technologies in popular Action Anticipation implementations, and to attempt to make use of the best technologies from both model types to learn which aspects are most important when creating an action anticipation model.

2

Existing models and datasets

2.1 LSTM approach - RULSTM

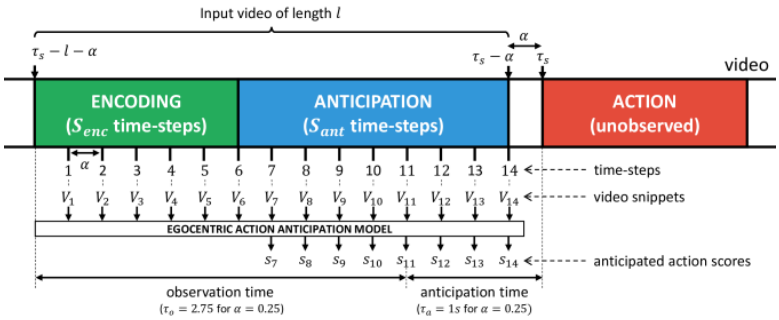

Figure 2.1: RULSTM's Video processing strategy using $S_{enc} = 6$ and $S_{ant} = 8$. [8]

Rolling Unrolling LSTM (RULSTM) is an egocentric action anticipation model that uses two LSTMs. The first LSTM - 'Rolling LSTM' is used to encode observed context from the past, and then an 'Unrolling LSTM' decodes this to predict what may happen next in a video.

RULSTM was chosen as a model to focus on for this project because even though it is a model that is a few years old now (released in 2019), it still performs almost as well as more recent models according to its "Top 5 Accuracy" scores (see chapter 4.1) on the EPIC-Kitchen dataset [14]. RULSTM is also much better documented. The model's functioning is outlined below, based on the original paper on RULSTM by Furnari et al. [8].

This model processes RGB, optical flow, and object-based features as its modalities. RGB and flow modalities are processed in separate branches of the architecture, while the object-based features are fused with the output of the flow branch. Temporal Segment Networks (TSN) [15] are used to represent features from the RGB and Flow modalities. TSN allows for the modelling of long-term temporal structures within videos by leveraging motion information using dense optical flow fields as the source of motion representation. TSN has been proven to provide a powerful and efficient way to model the temporal dynamics of actions in videos [15]. To extract optical flows before using TSN, the TV-L1 algorithm [7] is used on the RGB videos. Object features are extracted using a Faster R-CNN object detector, trained to recognise object classes from Epic-Kitchens-100.

The processing strategy of this model seen in Figure 3.1 involves a short video snippet being consumed every V_t consumed every α seconds where t refers to the current time step. An action occurring at time Ts is anticipated by processing a video segment of length l starting at time $T_s - l - \alpha$ and ending at $T_s - \alpha$. The encoding stage is carried out for S_{enc} time steps and anticipation is carried out for S_{ant} time steps.

The Rolling LSTM (R-LSTM) used during the encoding stage takes an input in the form of feature

vectors which can be extracted from the different modalities. It encodes the video by summarising video content within S_{enc} input video snippets, one snippet at a time. At each time step, modality-specific representation functions $\phi_1, .., \phi_M$ are used to obtain modality-specific feature vectors $f_{1,t} = \phi_1(V_t), .., f_{M,t} = \phi_M(V_t)$ with M referring to the number of modalities, $f_{M,t}$ referring to the calculated feature vector at time step t for modality m. Multiple identical branches are used to analyse the video according to its different modalities. Feature vectors $f_{M,t}$ are fed to the m^{th} branch of the architecture. All this allows the R-LSTM to capture temporal dependencies within an input sequence to provide a context for making predictions. In other words, it "rolls" through the input sequence to generate a sequence of hidden and cell states. Feature vectors are passed to the R-LSTM as follows:

$$(h^R_{m,t}, c^R_{m,t}) = LSTM_{\theta^R_m}(f_{m,t}, h^R_{m,t-1}, c^R_{m,t-1}), \tag{2.1}$$

where $LSTM_{\theta^R_m}$ is the R-LSTM of branch m, the learnable parameters are θ^R_m , and the hidden and cell states are $h^R_{m,t}$ and $c^R_{m,t}$

The Unrolling LSTM (U-LSTM) used in the decoding (anticipation) stage decodes the encoded hidden and cell states from the R-LSTM, and "unrolls" them to output S_{ant} action scores S_t that are used to perform action anticipation. $c^R_{m,t}$. It takes over the hidden and cell vectors from the R-LSTM at the current time step and iterates over the current video snippet's representation $f_{m,t}$ for the number of times needed to equal the number of time steps to reach the beginning of the action. The hidden and cell states of the U-LSTM are calculated as follows:

$$(h^U_{m,t}, c^U_{m,t}) = LSTM_{\theta^U_m}(f_{m,t}, h^U_{m,j-1}, c^U_{m,j-1}), \tag{2.2}$$

where j is the iteration $LSTM_{\theta^U_m}$ is branch m of the U-LSTM, θ^U_m is the learnable parameters, $h^U_{m,t}$, $c^U_{m,t}$ are the hidden and cell states calculated.

Sequence Completion Pre-training (SCP) is used in RULSTM to encourage each LSTM (R-LSTM, U-LSTM) to specialize in the two different sub-tasks of encoding past observations and anticipating future actions. In SCP, connections of the decoder LSTM (U-LSTM) are modified to allow it to process future representations, rather than iterating on the current one. SCP involves training the RULSTM architecture on a sequence completion task, where it is required to predict missing frames in a video sequence. As a result of pre-training the model in this manner, the model learns to encode past observations and anticipate future actions more effectively, improving its overall performance.

Modality Attention (MATT), is a mechanism introduced in the RULSTM paper, used to adaptively fuse multi-modal predictions. MATT estimates attention Scores are calculated by processing the concatenation of hidden and cell vectors from modality-specific R-LSTM branches using a deep neural network. A SoftMax function is used to compute the attention weights for each modality, which are then used to combine predictions from all modalities. This mechanism allows the model to effectively take advantage of the complementary nature of different modalities and improve its overall action anticipation performance. RULSTM's overall architecture including MATT can be seen in Figure 2.2.

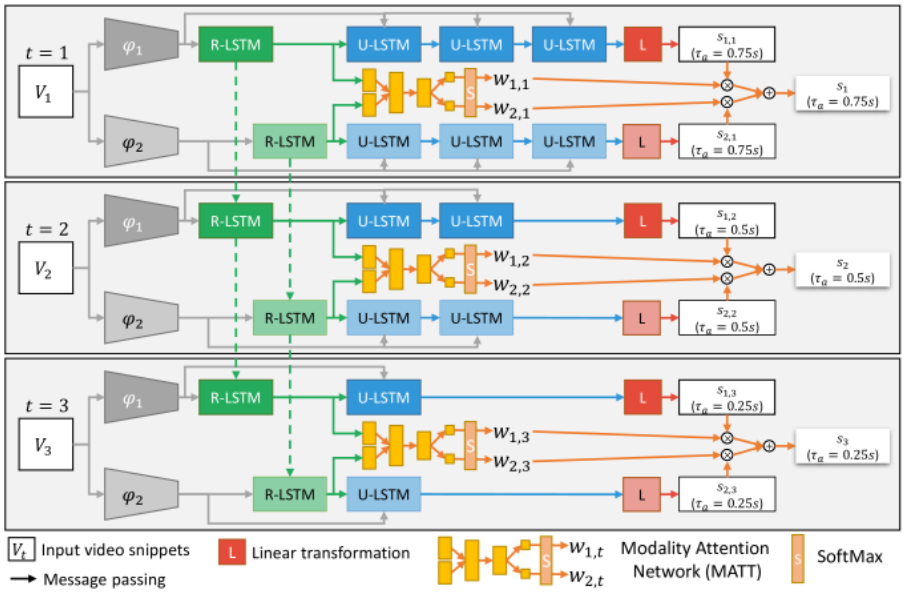

Figure 2.2: RULSTM's complete architecture using including Modality Attention. [8]

2.2 Transformer approach - AFFT

AFFT stands for Anticipative Feature Fusion Transformer. It is a model that performs the task of action anticipation, by predicting the future action labels given a sequence of multi-modal data. The model, proposed by Zhong et al. [16], consists of three main components: a feature extractor, a multi-modal feature fusion module, and a feature predictor. AFFT was tested by its creators using the same kitchen-based action anticipation/recognition datasets used for RULSTM - Epic-Kitchens-100 and EGTEA Gaze+. AFFT improves upon Anticipative Video Transformer (AVT) [17], by using it's decoder alongside AFTT's custom model. AVT is another transformer-based model for action anticipation in videos by Facebook AI Research. The anticipation model predicts $\hat{z}_{i+1} = a_\Omega(z_i), i \in 1, ..., T$, where $\hat{z}_{i+1}$ represents the predicted feature vector for the next time step, a_Ω represents the feature anticipation module, z_i represents the feature vector at time step i, i represents the current time step and T is the total number of steps in the input sequence. AFTT's overall simplified architecture can be seen in Figure 2.3.

We had initially planned to use AFFT for this experimentation section of this project, however, due to the relatively new nature of this model, and the lack of documentation, we struggled to build upon or even recreate the published results. We contacted the creator of AFFT on their official GitHub repository - [1], but did not receive any response as of writing this report. Instead, we used the FUTR model, which had its own issues such as not implementing noun and verb predictions, and also not having extensive documentation, however, it was easier to work with nevertheless.

The AFFT model makes use of four modalities: RGB, optical flow, depth, and audio. The same TSN features as RULSTM are used in this project for RGB and Optical Flow. This model also uses

[1]https://github.com/zeyun-zhong/AFFT/issues/5

the same R-CNN object extraction method as RULSTM. The audio modality for this model uses features extracted from 1.28 seconds of past audio per timestamp, converted to a log spectrogram representation, which is then fed to a TSN network for the final representation.

This model also implements visual features extracted from RGB frames using a Swin-Transformer model [18]. Swin Transformer features are hierarchical feature maps generated from computing self-attention locally within non-overlapping windows within non-overlapping windows that partition an image (frame of a video in this context), theoretically allowing for faster and more accurate than other similar feature representation methods[18]. Swin features are extracted in this model by feeding 32 consecutive past frames to the model for each timestamp.

The multi-modal feature fusion module combines the features from each modal using an attention-based fusion method that allows the model to selectively attend to different modalities at different times, similarly to how Modality Attention (MATT) from RULSTM works.

The feature predictor of this model takes the fused features and predicts a future action label using AVT's decoder, or as it is referred to in AVT's paper, the 'Head Network' - AVT-h. which predicts future features, $\hat{z}_t$, using D, a Casual Transformer Decoder on past features z_t. 'Casual' in this context refers to a transformer that only attends to frames that come before the current frame in the input sequence.

$$\hat{z}_1, , \hat{z}_t = D(z_1, , z_t). \tag{2.3}$$

The predicted features can then be decoded into action classes using a linear classifier. D uses a masked transformer approach, inspired by generative language models such as GPT-2. Firstly, a temporal position encoding is implemented as a learned embedding of the frame's position within the clip. The embedded features are passed through decoder layers which consist of masked multi-head attention, LayerNorm and a Multi-Layer Perceptron. The final output is passed through another LayerNorm to produce future frame embeddings. This architecture can be seen in Figure 2.4.

AFTT's overall architecture consists of three different fusion modules: SA-Fuser, T-SA-Fuser, and CA-Fuser. The SA-Fuser, applies Transformer Encoder blocks at individual time steps. The T-SA-Fuser combines temporal and modality attention and performs fusion on the whole temporal sequence at once. The CA-Fuser is a transformer decoder-based fusion module that introduces a new modality with each consecutive block, splitting the process of attention into separate smaller problems instead of presenting all temporal and modality tokens at once.

2.3 Transformer approach - FUTR

FUTR is a transformer-based model that makes use of self-attention mechanisms to capture long-range temporal relations between past and future egocentric and third-person video features so that it can perform action anticipation. FUTR's main modality input comes in the form of i3d RGB extracted features - spatio-temporal features learned by the Two-Stream Inflated 3D ConvNet (i3d) model [19].

This model was chosen to focus on for this project as it is a relatively recent transformer-based model that performs comparably to more complex state-of-the-art models while also benefiting from

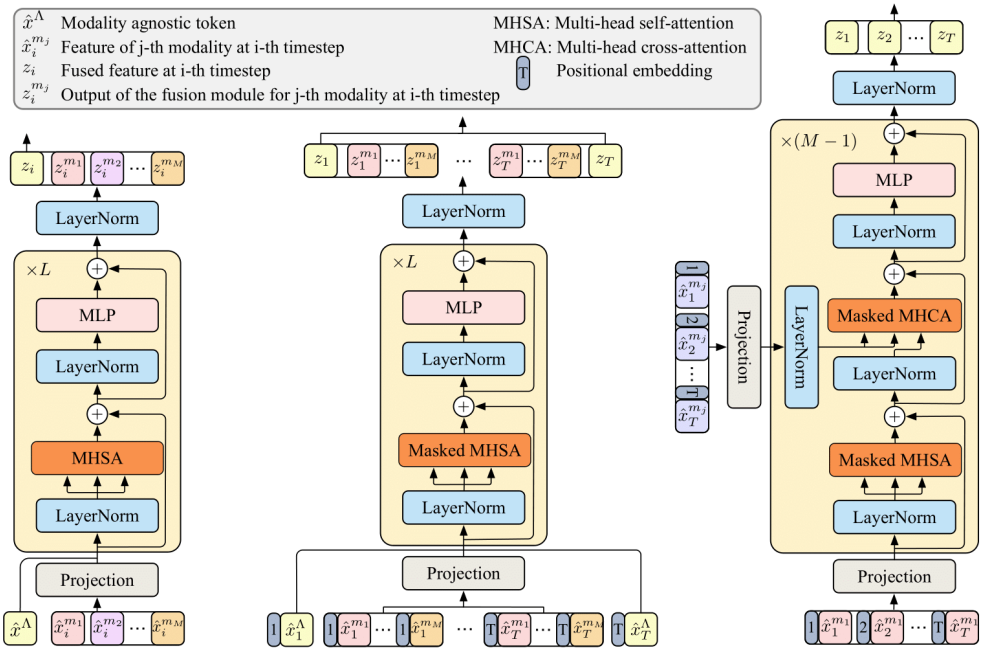

Figure 2.3: AFTT's overall architecture. The SA-Fuser, shown on the left, T-SA-Fuser, shown in the middle, CA-Fuser, shown on the right. [16]

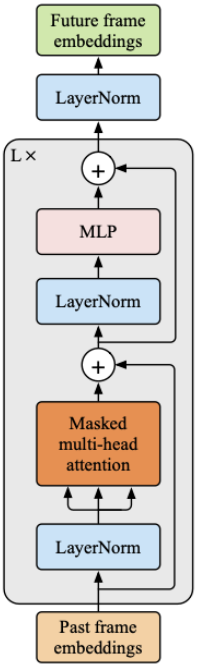

Figure 2.4: AVT's decoder overall structure. [17]

its ability to perform parallel decoding. Recent data shows that the best-performing Transformer model for Action Anticipation is Facebook Research's AVT+. In FUTR's paper, in some experiments, it is demonstrated that FUTR can perform up to 173 times faster than AVT+ (with a β value of 0.01), while still producing comparable action anticipation results. The following summary explains the functionality of the FUTR model, based on FUTR's original paper by Gong et al. [10].

FUTR's encoder uses visual features as input and segments actions from past frames so that it can learn feature representations using self-attention. Each encoder layer is composed of a multi-head self-attention layer, layer normalisation and a feed-forward network layer.

Input tokens are obtained using the equation $X_0 = ReLU(EW^F)$, where X_0 is the input tokens, $ReLU$ is a non-linear activation function, E is a matrix defined by $E \in \mathbb{R}^{T^O \times C}$ that contains visual features from the past (T^O is the number of observed frames, C is the dimensionality of the visual features). W^F is a weight matrix of size $C \times D$ that maps visual features to a lower-dimensional space, and F is a hyperparameter that determines the dimensionality of the output tokens.

The encoder is made up of L^E identical layers, which each consist of a Multi-Head-Self-Attention mechanism (MHSA), Layer Normalisation (LN), and Feed-Forward-Networks (FFN) with residual connections to allow the model to learn and prevent errors from building up over time. MHSA takes in a sequence of vectors and computes a weighted sum of them, where the weights are determined by a learned attention mechanism. This attention mechanism assigns higher weights to vectors that the model considers to be more relevant for the prediction of future actions, allowing the overall model to focus more on parts of the input sequence that are more useful.

Multi-head attention (MHA) is used based on scaled dot-product attention [20] with input tokens

X and Y. The MHA output is defined as:

$$MHA(X, Y) = [Z_1, ..., Z_h]W^O, \tag{2.4}$$

$$\text{where } Z_i = ATTN_i(X, Y) \text{ and} \tag{2.5}$$

$$ATTN_i(X, Y) = \sigma(\frac{(XW_i^Q)(YW_i^K)^\top}{\sqrt{D/h}})YW_i^V \tag{2.6}$$

With $W_i^Q, W_i^K, W_i^V \in R^{D\times D/h}$ being the query, key, and value projection layers at the i-th head, respectively, K refering to the number of action classes to be predicted, H, V, O and K referring to learnable projection matrices used to transform the input variables X and Y into query, key, and value representations, h being the number of heads, and σ being the softmax function. $W^O \in R^{D\times D}$ is an output projection layer. MHSA is based on an MHA module with two of the same input.

$$MHSA(X) = MHA(X, X) \tag{2.7}$$

An output token X_{l+1} can be obtained from the l-th encoder layer using the Linear Normalisation and Feed-Forward-Networks. In the below, P is a learned positional encoding matrix that encodes temporal position information:

$$X_{l+1} = LN(FFN(X'_l) + X'_l), \text{ where } X'_l = LN(MHSA(X_l + P) + X_l) \tag{2.8}$$

FUTR's Decoder takes in a series of generated output tokens (referred to as action queries) as input and decodes them to anticipate potential future action labels. Action queries are represented by learnable tokens $Q \in \mathbb{R}^{M\times D}$, with M being the number of learnable tokens in each action query Q. Temporal orders of queries correspond with future actions, ie., the i^{th} query represents the i^{th} future action.

The decoder has L^D decoder layers, with each layer including a multi-head self-attention (MHSA) layer, a multi-head cross-attention (MHCA) layer, layer normalization (LN), and a feed-forward network (FFN). An output query Q_{l+1} from the l-th decoder layer can be obtained as follows:

$$Q_{l+1} = LN(FFN(Q''_l) + Q''_l), \tag{2.9}$$

$$Q''_l = LN(MHA(Q'_l + Q, X_{LE} + P) + Q'_l), \tag{2.10}$$

$$Q'_l = LN(MHSA(Q_l + Q) + Q_l), \tag{2.11}$$

$$Q_0 = [0, .., 0]^\top \in \mathbb{R}^{M\times D} \tag{2.12}$$

Where X_{LE} is the final output of the encoder. The first decoder layer Q_0 is initialised with zero vectors.

The output of the last decoder layer Q_{LD} is used to generate future action logits (vectors to represent the output) - $\hat{A} \in \mathbb{R}^{M\times(K+1)}$ by applying a fully connected (FC) layer $W_A \in \mathbb{R}^{D\times(K+1)}$, followed by a softmax, and duration vectors $\hat{d} \in \mathbb{R}^M$ by applying another FC layer $W_D \in \mathbb{R}^D$:

$$\hat{A} = \sigma(Q_{LD}W_A), \ \hat{d} = Q_{LD}W_D. \tag{2.13}$$

For clarification, in the above, $\hat{d}$ refers to the predicted duration vector for future actions, K refers to the number of action classes to be predicted, M refers to the number of learnable tokens in each query, σ is the softmax function, W_A is a weight matrix, and D is the dimensionality of each token. If none of the future actions are predicted, a dummy class "NONE" is predicted by the queries, resulting in a total of $K + 1$ classes.

Multi Head Cross Attention is not explicitly explained in the decoder section of FUTR's paper, however, its role in the model is to capture cross-modal dependencies and generate accurate predictions of future actions.

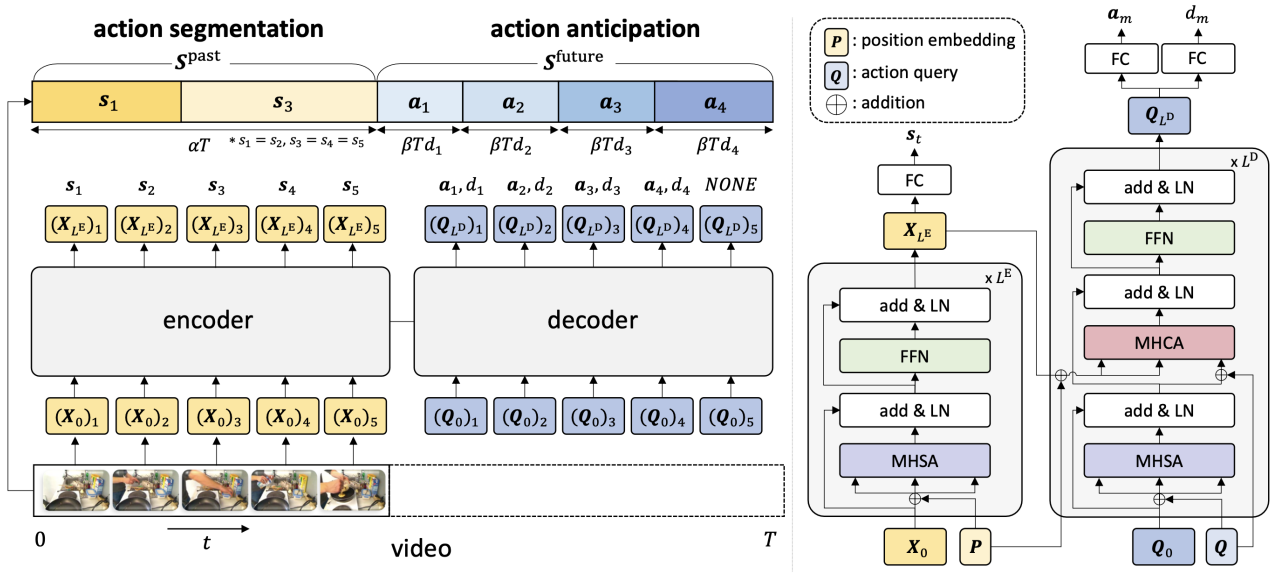

Figure 2.5: FUTR's overall architecture. [10]

2.4 Datasets

We will evaluate the models and experiments using three widely used action recognition and anticipation datasets. These datasets - Breakfast, 50-Salads and Epic-Kitchens - are all kitchen-based datasets, which are especially useful for action anticipation for several reasons, including:

- Fine-grained Actions: Some datasets may label an entire video with a single action, however, the following datasets have fine-grained actions such as "cracking an egg" or "take pan". Action labels of this nature are crucial for action anticipation, as the goal is to predict the next specific action, not the vague general action.
- Temporal Annotations: These datasets provide annotations which indicate when each action starts and ends, allowing models to learn the sequences and durations of actions.
- Large Volume of Data: Training an Action Anticipation model requires large amounts of data. These datasets provide a substantial number of videos and annotations, providing multiple examples of each action annotation, enabling the models to produce robust and accurate results.

The **"Breakfast"** [21] is a popular dataset for video-based action recognition and anticipation tasks. This dataset consists of 77 hours of video with an average length of average 2.3 minutes, from 52 actors

preparing breakfast dishes in various kitchens. Each of these videos can be categorised into one of 10 activities related to the preparation of breakfast, with 48 specific action labels used. These videos are recorded from several different camera angles in a third-person format. Example actions from this dataset include "cut_cheese" and "place_lettuce_into_bowl".

The **"50-salads"** [22] dataset consists of annotated accelerometer and RGB video data from 25 people preparing two mixed salads each. The dataset contains over 4 hours of videos with 17 fine-grained action labels for 3 high-level activities. The videos are, on average, longer than Breakfast's videos with 20 actions on average.

"Epic-Kitchens-55" [23] is a large egocentric action anticipation dataset, consisting of 432 egocentric videos recorded by 32 participants wearing head-mounted cameras while cooking. The dataset includes 125 verb and 331 noun classes such as "open" and "fridge". The creators of this dataset also made a larger version - EPIC-Kitchens-100, however, this dataset is larger than this project needs or can cope with considering storage limitations.

Figure 2.6: Example egocentric frames from the Breakfast Actions Dataset, demonstrates how one scene can be viewed from multiple camera angles. Taken from [21]

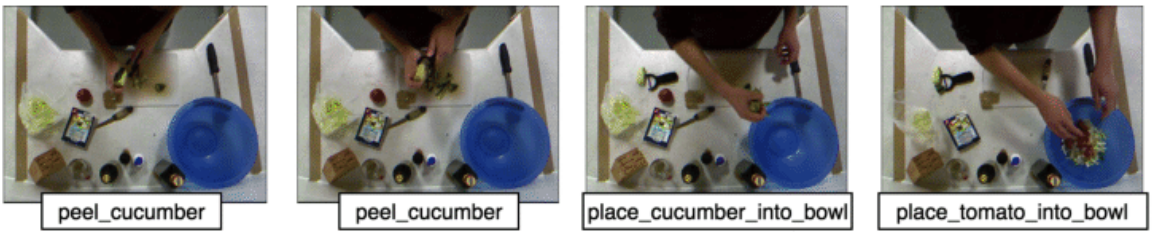

Figure 2.7: Example third person frames from the 50-Salads Dataset from [24].

Figure 2.8: Example egocentric frames from the EPIC-Kitchens Dataset from [25].

3

Experiments and evaluation

3.1 Proposed experiments

Compare performance between LSTM (RULSTM) approach and Transformer based approach (FUTR): The aim of this experiment is to evaluate the strengths and weaknesses of each approach and determine which one is more effective in predicting human actions. We had initially planned to use AFFT for this experimentation section of this project, however, due to the relatively new nature of this model, and the lack of documentation, we struggled to build upon or even recreate the published results. We contacted the creator of AFFT on their official GitHub repository [1], but did not receive any response as of writing this report. Instead, we used the FUTR model, which had its own issues such as not implementing noun and verb predictions, and also not having extensive documentation, however, it was easier to work with nevertheless.

Test both models using egocentric and third-person datasets: In RULSTM's original paper, the model is only tested with egocentric datasets. This experiment will attempt to see if the RULSTM model can be used for action anticipation in third-person videos to allow for a fair comparison with FUTR using the same datasets (both egocentric and third-person).

Implement TSN modality into FUTR: FUTR currently uses i3d features as its RGB modality, which process 64-frame video snippets, roughly over 2 seconds of past frames per feature on a video with 25 frames per second, whereas TSN features aim to model temporal dynamics of actions in videos by using short snippets over a long video sequence with a sparse sampling scheme, where one snippet is randomly sampled from each segment. This allows TSN to model long-range temporal structures over the whole video [15]. This experiment will attempt to see if TSN features can improve FUTR's performance.

3.2 Evaluation metrics

Top-1 Accuracy is the traditional accuracy metric used for classification. This metric checks if the highest predicted class (with the highest probability score) matches the true class.

$$\text{Top-1 Accuracy} = \frac{\text{Number of correct top-1 predictions}}{\text{Total number of predictions}} \tag{3.1}$$

Top-5 Accuracy is an accuracy metric popular in tasks with large numbers of classes. The metric

[1] https://github.com/zeyun-zhong/AFFT/issues/5

checks whether the true class is within the top 5 predicted classes.

$$\text{Top-5 Accuracy} = \frac{\text{Number of samples where true class is in top-5 predictions}}{\text{Total number of predictions}} \tag{3.2}$$

Avg Class Precision. Precision measures how many of the predicted classes are true positives. The average class precision is the mean of the precision for each class.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3.3}$$

Avg Class Recall. Recall measures "out of how many of the actual positives for the class, how many were predicted as positive". The average is the mean of the recall values for each class.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3.4}$$

3.3 Preparing datasets

To complete the experiments, we need the TSN RGB features for each dataset. RU-LSTM's GitHub repository provides example code for extracting features from RGB, Flow and Object modalities. These can be modified and adapted to be used on other datasets. The pre-extracted features for Epic-Kitchens-55 are also provided in this repository, so only Breakfast and 50-Salads need to be pre-processed for this project.

Firstly, we listed each .avi file and 50-salads datasets, then, for RGB features, we split each video into jpeg image frames resized to 456 × 256 pixels (same as how the model's creators processed Epic-Kitchens-100 frames). Then we adapted the provided code to generate an LMDB database of TSN-extracted features that the model has been designed to read. To make the comparisons between the two models fair, we only use RULSTM's RGB modality, as FUTR does not use flow/obj modalities. My original plan was to implement this into FUTR however I did not have enough time to do that in the time frame of this project. The code for the generation of video frame jpg files and RGB features can be found in extract_rgb.py

Then we adapted the dataset's action labels into the necessary files. This includes an actions.csv file containing a map of numerical action IDs, full action labels (eg. Open_fridge), and numerical verb and noun field, with the first number being the verb and the second being the noun (eg. 7_1 = Close_fridge with Close being assigned the numerical verb value of 7 and fridge being assigned the numerical noun value of 1). The other files include training.csv, and validation csv, which contain the action labels for each video in the training and validation splits, consisting of a numerical ID field, the name of the video file, the start frame, the end frame, the action label ID, the noun label ID and the verb label ID. This allows for benchmarking of noun and verb prediction accuracy in addition to evaluating the accuracy of the full action label, which has yet to be attempted for the 50-Salads and Breakfast datasets as the action labels provided only state the full action label.

The Breakfast and 50-Salads datasets dataset did not come with pre-defined splits, so we used a

random split for both, with the first 70% of the videos being used for training and the remaining 30% used for validation and testing.

The code for converting action labels can be found in rulstm/FEATEXT/convert_action_labels.py.

The same TSN features are used for testing FUTR; however, the action labels are in a different format. The files needed include 'mapping.txt': This file maps action labels to integer IDs. For example, a line in mapping.txt follows could look like: "0 take_pan". Another file is 'train.splitX.bundle' and 'test.splitX.bundle', where X corresponds to the number of splits needed. These files contain a list of video file names for each test/train split. groundTruth/video.txt' was the final file required. Each video in the dataset has a corresponding ground file for action labels. The ground truth files contain actions for each frame, with each line in the file being an action and each line corresponding to a frame.

FUTR/datasets/generate_labels.py contains the code that reads Epic Kitchen's original action label files and converts them into the format that FUTR expects. For Breakfast and 50-Salads, we use the same action labels FUTR uses for their i3d features.

3.4 Comparison of performance between LSTM (RULSTM) approach and Transformer based approach (FUTR)

The main task with regards to working with RULSTMs code was implementing the ability for the model to work with the Breakfast and 50-Salads datasets. The extraction of TSN features and generation of compatible action labels is discussed in section 3.3. RULSTM was designed to work with two datasets, Epic-Kitchens and EGTEA; however, EGTEA is not officially supported, and the code on the RULSTM Github repository [8] needs a fair amount of modification to load features and action labels from datasets other than Epic-Kitchens.

Since the feature storage method was not documented, we looked through the provided files for Epic-Kitchens and EGTEA to figure out how to format the input data. Firstly, we created read_lmdb.py, a basic python program that reads through LMDB files and lists every entry. From this, we learned that features are stored with keys in the format "VideoFileName_frame_0000000001.jpg" where the number corresponds to the frame number that the feature belongs to, with zero padding to ensure the number is ten numbers long. We then analysed the files provided for EGTEA's action labels, and converted Breakfast and 50-Salad's labels as discussed in section 3.3.

When trying to test the models, several files were required that were not found in the original GitHub repo. We tried creating them; however, with no documentation and no example of what they were supposed to look like, we could not recreate them. We opened an issue on the Github repo, asking for clarification or if they could provide the needed files; however, as of writing this, the issue has not yet been responded to[2].

We made a workaround by modifying the validation method and using the original action label, verb and noun files which ended up working well, with the only issue being that the accuracy scores

[2]https://github.com/fpv-iplab/rulstm/issues/30

for Epic Kitchens were very low compared to the scores achieved for Breakfast and 50-Salads; however, this can likely be explained by the relatively specific nature of the action labels in this dataset, the added complexity of it being an egocentric dataset, and the fact that we only used the RGB modality, the results seemed reasonable.

3.5 Implementation of TSN modality into FUTR

As we needed the comparison between FUTR and RULSTM to be fair, we needed to use the same data for each. This came with several challenges as FUTR is designed to use i3d features loaded from a .npy file, very different to the TSN features loaded from an LMDB file. The first task was to modify main.py to accept the Epic-Kitchens dataset as an option. This was relatively straightforward. Next, we modified the basedataset.py class. The BaseDataset class is a subclass of PyTorch's Dataset class used to load and pre-process data.

We added a load_lmdb_features method to load each video's TSN features from an LMDB database. This method reads every possible frame for a video in a specified LMDB database. It did this by reading the features as utf-8 text, then created a numpy array from the string, with each element being read as a float32 value". Individual decoded frame features are then returns as a numpy array of features for the whole video.

There were some issues with loading features from the Breakfast database, where video names in FUTR's provided GroundTruth files did not match the actual names from the dataset. These files needed to be renamed to match. Another minor change was specifying the input_dim hyperparameter as 1024. This refers to TSN's dimensionality of 1024, contrasting to i3d's dimensionality of 2048.

While training the Breakfast dataset model, we had issues where the training program would crash without producing an error code. This was frustrating as it was challenging to debug. Eventually, through extensive testing, we realised it was likely an issue with GPU memory running out due to the initial batch size of 16. We reduced the batch size to 3, which fixed this issue.

Unfortunately, we were not able to train Epic-Kitchens using FUTR. We are unsure why; however, after rigorous testing and debugging, we could not fix this error. It could potentially be an issue with the frame extraction method; whereas the other two datasets used the .AVI format, Epic-Kichens used .MP4. It is possible that the features generated from these frames are incompatible with the model. Another potential issue could be a hardware issue, same as before with the Breakfast model, where the Graphics Card used simply did not have enough VRAM to deal with the large amount of data that makes up Epic-Kitchens videos. We attempted increasing the max_pos_len parameter and reducing the batch size to its minimum value, 1; however, we could not get the code to run without errors.

3.6 Implementing evaluation metrics

RULSTM naively supports Top-1 Accuracy, Top-5 Accuracy and Mean Top-5 Recall; however, as briefly mentioned before, the code provided on GitHub for validation and testing did not initially work, so to get around this, we modified the 'validate' function to validate a current model against unseen labels (the test set). While training, the model still used a separate validation set. Once

this workaround was implemented, we added a couple of methods to utils.py to calculate Top-1 Precision. These methods were topk_precision_multiple_timesteps and topk_precision. They are heavily based on the existing code, making working out how data is stored much more straightforward. A significant challenge in implementing evaluation metrics was the exploratory analysis required to understand what each variable stores and how, as they weren't documented. The topk_precision_multiple_timesteps function calculates Top-k precision for each required time step using the topk_precision function. The topk_precision function, which is heavily based on the provided topk_recall function, calculates the number of true positives and false positives for each class, then returns calculated precisions.

FUTR natively supports MoC (Mean over Classes) accuracy, so implementing other accuracy methods was slightly more challenging as there was less to base them off. We implemented Top-1 Accuracy, Average Class Precision and Average Class Recall. We could not implement Top-5 Accuracy as we could not figure out how to get multiple predictions for a single time frame. Same as RULSTM, the variables used were not well documented, and a significant amount of time was spent performing exploratory analysis on them to attempt to learn what each variable represented. We needed the number of True Positives, False Positives, and False Negatives to calculate Precision and Recall. We calculated these by using a modified version of the provided "eval_file" method, which previously returned True Recognitions and False Recognitions. We created a modified version of this method - "eval_file_more" in utils.py. This function instead returned True Positives, False Positives, False Negatives and Top-1 accuracies. These values could then be used in the main code in predict.py to produce the evaluation metrics.

3.7 Results

This section contains results from the models trained as previously described.

Using FUTR, training times varied by a few minutes each time; however, each split for 50-Salads and Breakfast using TSN took roughly 15 minutes to train. Each split for 50-Salads using i3d features took roughly 4 minutes to train, and each split for Breakfast using i3d features took roughly 15 minutes. Using RULSTM, training for 50-Salads using TSN features took 7 minutes, Breakfast took 20 minutes, and Epic-Kitchens took about 50 minutes.

As previously mentioned, training times varied by a few minutes each time, and attributes such as batch size, other parameters, and programs running in the background simultaneously could potentially affect training times. These results are purely anecdotal; however, it is worth noting that training 50-Salads and Breakfast using the same TSN features took roughly 4x longer with the transformer model - FUTR, than with the LSTM-based model - RULSTM.

3.7.1 Anticipation results using TSN features with RULSTM

Results from RULSTM are split by Time to Anticipation (2.0 - 0.25), which refers to the amount of time in advance that the model anticipates an action in the video.

		2.0	1.75	1.5	1.25	1.0	0.75	0.5	0.25
Verb	Top-1 Accuracy	09.58	10.27	09.96	09.96	09.89	10.14	10.46	10.40
	Top-5 Accuracy	24.45	24.70	24.70	24.76	24.95	24.76	24.39	24.07
	Mean Top-5 Recall	30.50	31.08	30.96	30.80	30.72	30.68	30.54	29.54
	Mean Top-1 Precision	13.79	13.89	13.94	13.94	13.38	13.20	13.65	13.12
Noun	Top-1 Accuracy	11.09	11.85	11.47	11.34	11.85	11.97	12.35	12.54
	Top-5 Accuracy	30.88	30.50	30.25	30.37	31.00	31.88	31.19	31.88
	Mean Top-5 Recall	21.33	20.95	20.79	21.04	21.67	22.05	20.71	21.02
	Mean Top-1 Precision	05.38	05.64	05.50	05.46	05.69	05.74	05.95	06.02
Action	Top-1 Accuracy	60.81	63.07	62.76	63.01	63.14	63.01	62.26	60.49
	Top-5 Accuracy	89.98	89.48	89.73	89.73	90.11	89.60	89.35	89.22
	Mean Top-5 Recall	87.07	87.41	87.18	87.32	87.39	86.81	85.67	84.65
	Mean Top-1 Precision	55.73	57.52	60.53	60.80	62.86	59.54	57.48	52.49

Table 3.1: Breakfast dataset anticipation results using TSN features with RULSTM

		2.0	1.75	1.5	1.25	1.0	0.75	0.5	0.25
Verb	Top-1 Accuracy	07.41	05.93	07.41	07.41	08.15	06.67	05.93	07.41
	Top-5 Accuracy	33.33	33.33	33.33	33.33	33.33	33.33	33.33	33.33
	Mean Top-5 Recall	83.33	83.33	83.33	83.33	83.33	83.33	83.33	83.33
	Mean Top-1 Precision	20.56	17.22	20.40	20.56	22.07	18.70	16.11	21.67
Noun	Top-1 Accuracy	06.67	09.63	08.89	11.11	11.11	11.11	09.63	08.15
	Top-5 Accuracy	43.70	46.67	43.70	45.19	45.19	45.93	45.19	42.22
	Mean Top-5 Recall	41.20	43.59	41.10	42.13	43.46	42.59	43.40	40.05
	Mean Top-1 Precision	11.17	14.41	12.50	16.67	15.34	15.34	14.35	13.43
Action	Top-1 Accuracy	22.96	23.70	20.74	21.48	24.44	22.96	26.67	25.19
	Top-5 Accuracy	77.04	77.78	74.07	80.74	81.48	78.52	81.48	80.00
	Mean Top-5 Recall	77.55	78.51	73.49	80.27	80.55	78.55	80.89	79.57
	Mean Top-1 Precision	21.47	22.51	18.71	20.29	24.37	23.65	24.20	22.73

Table 3.2: 50-Salads dataset anticipation results using TSN features with RULSTM

		2.0	1.75	1.5	1.25	1.0	0.75	0.5	0.25
Verb	Top-1 Accuracy	06.41	06.61	06.55	06.85	07.01	06.69	06.97	07.21
	Top-5 Accuracy	20.33	20.37	20.47	20.39	20.49	20.57	20.41	20.47
	Mean Top-5 Recall	04.30	04.39	04.39	04.52	04.64	04.90	04.37	04.37
	Mean Top-1 Precision	01.21	01.24	01.39	01.29	01.33	01.26	01.32	01.36
Noun	Top-1 Accuracy	00.38	00.28	00.28	00.28	00.36	00.28	00.34	00.26
	Top-5 Accuracy	01.75	01.79	01.83	01.93	01.85	01.85	01.77	01.69
	Mean Top-5 Recall	01.63	01.73	01.80	01.88	01.89	01.89	01.80	01.68
	Mean Top-1 Precision	00.31	00.19	00.21	00.20	00.31	00.30	00.41	00.29
Action	Top-1 Accuracy	09.94	10.44	11.51	12.39	13.20	14.18	14.96	16.01
	Top-5 Accuracy	25.25	26.49	27.84	29.10	30.73	31.71	33.22	34.06
	Mean Top-5 Recall	09.48	10.32	11.20	11.27	12.22	12.68	13.09	13.17
	Mean Top-1 Precision	03.56	04.09	04.40	04.58	04.95	04.96	05.15	05.28

Table 3.3: Epic-Kitchens-55 dataset anticipation results using TSN features with RULSTM

The gap between Top-1 and Top-5 accuracy in 50-Salads indicates that the correct action is unlikely to be the top action, however, the model still often recognises the correct action as a possibility. This is likely due to it being a much smaller dataset, leading to the model not being fully exposed to the full variability in action classes.

Overall, the third-person datasets (Breakfast and 50-salads) performed significantly better than Epic-Kitchens, despite this model being designed for Epic-Kitchen's egocentric data. This is likely because egocentric data is inherently more complicated and challenging to predict due to the dynamic viewpoint of the camera, leading to significant variability and unpredictability in data. This contrasts with third-person data, which is captured from a fixed perspective, making the data relatively more predictable and straightforward.

Notably, Top-5 recall is generally much higher than Top-1 precision, indicating that the model is better at capturing a broader set of potentially relevant actions rather than making a precise top prediction.

3.7.2 TSN and i3d feature anticipation results with FUTR

FUTR is designed to predict actions at a longer range; the values 10, 20, 30, and 50 in the table are the prediction rate, β, representing the percentage of the future action sequence the model is trying to anticipate. The value that produces results most similar to RULSTM's is 10 or 20, as most videos in the datasets have roughly 5-10 actions.

	10	20	30	50
Top-1 Accuracy	0.4758	0.4451	0.4398	0.4032
Mean over Classes	8.8404	8.6658	8.6718	8.8491
Avg. Class Precision	0.1803	0.188	0.1843	0.1915
Avg. Class Recall	0.1841	0.1805	0.1807	0.1843

Table 3.4: Breakfast dataset anticipation results using i3d features with FUTR

	10	20	30	50
Top-1 Accuracy	0.3066	0.2703	0.2188	0.1658
Mean over Classes	2.7224	2.4734	2.1884	2.32108
Avg. Class Precision	0.13192	0.1218	0.1002	0.1163
Avg. Class Recall	0.14328	0.1302	0.1152	0.0975

Table 3.5: 50-Salads dataset anticipation results using i3d features with FUTR

	10	20	30	50
Top-1 Accuracy	0.1914	0.1936	0.1960	0.1848
Mean over Classes	3.7368	3.8542	4.0175	4.1071
Avg. Class Precision	0.0953	0.0868	0.0865	0.0891
Avg. Class Recall	0.0779	0.0803	0.0837	0.0856

Table 3.6: Breakfast dataset anticipation results using TSN features with FUTR

	10	20	30	50
Top-1 Accuracy	0.1226	0.1000	0.1096	0.0998
Mean over Classes	0.9376	0.8350	0.9527	1.2076
Avg. Class Precision	0.0542	0.0547	0.0643	0.0718
Avg. Class Recall	0.0493	0.0426	0.0497	0.0636

Table 3.7: 50-Salads dataset anticipation results using TSN features with FUTR

Comparing RULSTM and FUTR, when testing using the Breakfast dataset, RULSTM outperforms FUTR, whereas FUTR (using i3d features) outperforms RULSTM when testing using 50-Salads. This indicates that FUTR may be less prone to overfitting on smaller datasets such as 50-salads.

It is clear that i3d features consistently outperform the TSN features across both datasets when using the FUTR model across all metrics. There are several reasons why this may be the case.

Firstly, FUTR was initially designed and optimised for i3d features, meaning that the architecture, hyperparameters, and other attributes might not be optimal for learning with TSN features. It's also possible that TSN might introduce more complexity that the model is not capable of effectively regularising, which could potentially lead to some overfitting.

While TSN may have certain advantages over i3d in some contexts, it does not necessarily mean that it is a superior method of representing features. i3d can capture both spatial and temporal information in videos. TSN is effective in capturing long-range temporal structures, however, it might miss some fine-grained temporal patterns that i3d captures.

With both 50-Salads and Breakfast, RULSTM with TSN features generally has higher precision than FUTR with i3d features. This implies that predicted actions from RULSTM are more likely to be correct. Recall values are varied, however, RULSTM still generally outperforms FUTR, suggesting it can produce a broader range of correct predictions.

3.8 Critical Assessment

In this section, we critically assess the methodologies, findings, experiments and research undertaken.

The biggest issue we had while completing this project was underestimating the complexity of understanding and working with action anticipation models. As someone who previously only had a very basic knowledge of AI and data science, the learning curve was steep. The original project planned to create an action anticipation model from scratch; however, when this proved too great of a task, we pivoted the project's focus to a more research-based approach. This was more manageable but still challenging as the action anticipation models used were difficult to understand and modify, as their documentation was relatively vague. This lead to changing the model to be used multiple times throughout the project, and not completing all the originally planned experiments due to time constraints.

We also did not expect the LSTM model to outperform the transformer model; this was surprising. However, in retrospect, we should have anticipated that the model would perform much better on third-person data than on egocentric data. This contradicted the original hypothesis that Transformer based models produce better anticipation results than LSTM-based models; however, it is also important to note that FUTR is not a perfect example of a transformer. We initially planned to use AFFT, as mentioned earlier. This model is based on another state-of-the-art transformer, AVT, and should be one of the best currently available. According to paperswithcode.com, it is the best publically available action anticipation model [26]. When we could not get AFFT to work, we switched to FUTR as it was relatively simpler and still produced decent anticipation results.

While the project faced numerous challenges, it also emphasised the importance of adaptability, the need for improved forward planning so that setbacks are not so severe, and the value of clear documentation in open-source code. These skills will undoubtedly help me better structure and shape future research and development and serve as a valuable foundation for approaching complex projects.

4 Conclusion

4.1 Future Research

If we were to continue this project, we would attempt to test the AVT model, [17] as it is a reputable and likely well documented Transformer based model that could provide a fairer comparison to RULSTM. We would also produce i3d features for the Epic-Kitchens dataset, then test them on RULSTM and AVT to more accurately compare how they perform compared to TSN features.

4.2 Conclusion

With applications ranging from video analysis to real-time robotics, action anticipation remains a crucial topic of research within AI and data science. The findings of this project offer insightful information on how various models and feature representations perform in the setting of action anticipation.

The models used were FUTR, a transformer-based model, and RULSTM, an LSTM-based model. As initially predicted, FUTR had a significantly longer training time than RULSTM. This emphasises how computationally intensive transformer models are. The results demonstrated superior performance when anticipating third-person datasets such as Breakfast and 50-Salads. Egocentric datasets such as Epic-Kitchens are inherently much more challenging to model due to the dynamic viewpoint of the camera, leading to significant variability and unpredictability in data.

Contrary to the initial hypothesis, overall, the LSTM-based model mostly outperformed the transformer-based model; however, this could partially be due to a sub-optimal implementation of TSN features into FUTR. This also shows that although transformer models have shown promise, LSTM models remain competitive, and the trade-off between accuracy and computational power requirement should be considered.

While completing this project, we experienced several setbacks and challenges, notably the steep learning curve of understanding anticipation models and the difficult task of deciphering and adding to code that has not been thoroughly documented. However, These challenges emphasised the importance of adaptability in research and the need for thorough planning for complex projects.

In conclusion, while transformer-based models have proven to be an effective and powerful tool in deep learning and action anticipation, as evidenced by the results of this project, there is no one-size-fits-all solution. Model and feature representation choice should depend highly on the specific dataset, available resources, and application. As the ever-evolving field of deep learning evolves, researchers should remain agile, continuously re-evaluating established concepts and embracing new approaches to expand the capabilities of action anticipation and deep learning as a whole.

References

[1] Richard Szeliski. "A Brief History". In: *Computer vision: Algorithms and applications*. Springer, 2011.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[3] Sera Buyukgoz et al. "Two ways to make your robot proactive: Reasoning about human intentions or reasoning about possible futures". In: *Frontiers in Robotics and AI* 9 (Aug. 2022). DOI: 10.3389/frobt.2022.929267.

[4] M. S. Ryoo. "Human activity prediction: Early recognition of ongoing activities from streaming videos". In: *2011 International Conference on Computer Vision*. 2011, pp. 1036–1043. DOI: 10.1109/ICCV.2011.6126349.

[5] Mingtao Pei, Yunde Jia, and Song-Chun Zhu. "Parsing video events with goal inference and intent prediction". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 487–494.

[6] Tomislav Pejsa et al. "Room2Room: Enabling Life-Size Telepresence in a Projected Augmented Reality Environment". In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work amp; Social Computing*. CSCW '16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1716–1725. ISBN: 9781450335928. DOI: 10.1145/2818048.2819965. URL: https://doi.org/10.1145/2818048.2819965.

[7] Javier Sánchez, Eva Llopis, and Gabriele Facciolo. "TV-L1 optical flow estimation". In: *Image Processing On Line* 3 (July 2013), pp. 137–150. DOI: 10.5201/ipol.2013.26.

[8] Antonino Furnari and Giovanni Maria Farinella. "What Would You Expect? Anticipating Egocentric Actions with Rolling-Unrolling LSTMs and Modality Attention." In: *International Conference on Computer Vision*. 2019.

[9] Congqi Cao et al. "VS-TransGRU: A Novel Transformer-GRU-based Framework Enhanced by Visual-Semantic Fusion for Egocentric Action Anticipation". In: *arXiv preprint arXiv:2307.03918* (2023).

[10] Dayoung Gong et al. *Future Transformer for Long-term Action Anticipation*. 2022. arXiv: 2205.14022 [cs.CV].

[11] Christopher Olah. *Understanding LSTM networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[12] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[13] Azmat Anwar et al. "Constructing Uyghur Named Entity Recognition System using Neural Machine Translation Tag Projection". In: (Oct. 2020).

[14] Papers With Code. *Papers with code - epic-kitchens-55 (unseen test set (S2) benchmark (action anticipation). Date Accessed: 20/04/2023*. Apr. 2023. URL: https://paperswithcode.com/sota/action-anticipation-on-epic-kitchens-55-1?metric=Top+5+Accuracy+-+Verb.

[15] Limin Wang et al. *Temporal Segment Networks for Action Recognition in Videos*. 2017. arXiv: 1705.02953 [cs.CV].

[16] Zeyun Zhong et al. *Anticipative Feature Fusion Transformer for Multi-Modal Action Anticipation*. 2022. arXiv: 2210.12649 [cs.CV].

[17] Rohit Girdhar and Kristen Grauman. "Anticipative Video Transformer". In: *CoRR* abs/2106.02036 (2021). arXiv: 2106.02036. URL: https://arxiv.org/abs/2106.02036.

[18] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *CoRR* abs/2103.14030 (2021). arXiv: 2103.14030. URL: https://arxiv.org/abs/2103.14030.

[19] João Carreira and Andrew Zisserman. "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset". In: *CoRR* abs/1705.07750 (2017). arXiv: 1705.07750. URL: http://arxiv.org/abs/1705.07750.

[20] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[21] Hilde Kuehne, Ali Arslan, and Thomas Serre. "The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 780–787. DOI: 10.1109/CVPR.2014.105.

[22] Sebastian Stein and Stephen J McKenna. "Combining embedded accelerometers with computer vision for recognizing food preparation activities". In: *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 2013, pp. 729–738.

[23] Dima Damen et al. "Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100". In: *International Journal of Computer Vision (IJCV)* 130 (2022), pp. 33–55. URL: https://doi.org/10.1007/s11263-021-01531-2.

[24] Li Ding and Chenliang Xu. *TricorNet: A Hybrid Temporal Convolutional and Recurrent Network for Video Action Segmentation*. May 2017.

[25] Clare Lapworth. *Epic-kitchens: Bringing helpful AI closer to Reality*. May 2021. URL: https://engineering.blogs.bristol.ac.uk/epic-kitchens-machine-learning-ai-computer-vision-reality/.

[26] URL: https://paperswithcode.com/sota/action-anticipation-on-epic-kitchens-100.