OSJ OPEN SCIENCE JOURNAL

RESEARCH ARTICLE

Going Deeper into Third-Person Action Anticipation

Gulu Mbongeni[1], Sareh Rowlands[1]*

[1]University Exeter, Exeter, United Kingdom

*Corresponding author: Sareh Rowlands: s.rowlands@exeter.ac.uk

OPEN ACCESS

Citation: Mbongeni G., Rowlands S...(2023) Going Deeper into Third-Person Action Anticipation. Open Science Journal 8(2)

Received: 14th July 2023

Accepted: 14th September 2023

Published: 16th October 2023

Funding: The author(s) received no specific funding for this work

Competing Interests: The authors have declared that no competing interests exists.

Abstract:

Autonomous systems have become more dominant in our lives and therefore the ability of these sys-tems to predict future plays an instrumental role to guarantee assistance and safety. As a result, analysing human actions and anticipation in videos is gaining a great deal of interest in academic re-search. This paper explores and reviews different deep learning techniques used in third-person action anticipation to provide an updated view of advancements in this field. The task of action anticipation is divided into feature extraction and a predictive model for many architectures. This paper outlines a project plan for action anticipation in the third person using step-based activity. We will use several data sets to compare some of these different architectures based on their prediction accuracy and ability to predict actions in varying time frames.

Keywords: Action anticipation, Third-person vision, Deep learning, LSTM

Introduction

Analysing human actions in videos is gaining a lot of interest in the field of computer vision due to its broad real-world applications, such as self-driving cars, video surveillance, and many other varying places where there are a lot of human-computer interactions. There are 3 related fields within computer vision: action recognition, action detection, and action anticipation. Action recognition aims to take a full video as input and classify the actions within the video; action detection is similar as it takes a full video and tries to see if the video contains a specific action. Both these fields are interested in labelling after the video has finished. Action anticipation is different to these two areas and fewer studies have been conducted in this area.

Action anticipation tries to predict future actions; it uses deep learning algorithms to infer future actions from recorded videos. Making a prediction of future labels for a video before fully observing the entire video is more challenging,

as a classifier has to predict a label for the next action to come while only seeing the first few frames, and many actions can be very ambiguous. Many of the techniques used in action recognition and detection are transferable to action anticipation. Action anticipation can be split into two crucial steps:

Feature extraction is the process of extracting the most relevant information from an image and representing the image in a lower-dimensional space. Techniques like CNN (Convolution Neural Networks) use pooling to reduce the dimensions of an image. Features are extracted to make feature vectors which can be used by a classifier to assign the input to a target label. Feature extraction in action anticipation is used with classification to identify the action that has been observed from the first few frames. This information can then be passed to the predictive model.

A prediction model is a model that takes the input from the feature extraction and makes a prediction. For some architectures, this could be predicting future frames from the input image or predicting the labels of the next frames. Algorithms that can predict the labels of the next frames are sometimes capable of predicting the duration of the action.

Predicting human actions is difficult, humans leverage a wide range of knowledge to infer about what will happen next, but computers can only use the information that has been fed to the model. Some action anticipation has tried using contextual information to aid prediction. [1] Proposes multiple sensors on a car to aid in action anticipation. They built an end-to-end deep learning architecture that jointly learns to anticipate and fuse information from different sensors. This increased the precision from 78% to 90.5%. Current works like [1] anticipate actions for a short time period of fewer than 10 seconds.

This study designs and runs experiments to compare some of the techniques used in action anticipation. We will compare these architectures based on several metrics, primarily the accuracy of the architecture, using two datasets. The two datasets chosen for this project are, 50 salads and breakfast dataset, more information on these two data sets can be found in Section.3.

Background

In this section we will outline the various information needed to understand this study.

Feature extraction

Feature extraction is the first step of many action anticipation models. Some models use handcrafted features (background subtraction, edge detectors), and other deep learning techniques. In the original literature review we discussed how this could be split into two groups:

- Holistic/global representations [2] - Feature extraction that extracts global representations. Using a top-down architecture, a person is localised using methods like background subtraction, then the region of interest is encoded [2]. This allows for the extraction of more information; however, it is very sensitive to noise and occlusion. It can also introduce irrelevant information from the cluttered background.
- Local representations [2] - Feature extraction that extracts local features. These methods encode a video sequence as a collection of local

spatio-temporal features. Using a bottom-up architecture, in-terest points are detected first, and local patches are extracted from spatio-temporal interest points [2]. Patches are then combined into a final representation. There are benefits to local representa-tions compared to holistic. Local representations avoid preliminary steps such as background sub-traction that are used in holistic methods. They can, however, be overly localised and incapable of capturing sufficient amounts of data. The deep learning technique we decided to focus on is CNN, which is the most popular deep learning method for feature extraction for action anticipation and has a multitude of uses.

CNN for feature extraction

CNN is a feed-forward Neural Network (NN), inspired by animal visual cortexes and has been ap-plied in many fields such as pattern recognition, vocal recognition, natural language processing, and video analysis. CNN in image classification is used to map an input image X, to an output class Y. For example, given an image of a dog, a classifier would out- put a class label dog, using its multiple layers of computation that extract features to learn a better representation of image data. CNNs are layered structures, with most CNNs having a convolutional layer, pooling layers, and a classification layer. Each hidden layer of a CNN performs a function on the input data. Earlier layers in a CNN extract low-level features (e.g. edges, shapes) and higher layers extract more complex features (e.g. face, hair). Various works use CNN to extract features, the models we will be investigating use a TSN (Temporal Segment Network) model [3], which uses CNN, to extract its features.

Predictive model

The prediction model is the part of the system that produces predictions based on either ground truth labels or the ground truth and extracted features. For many models, after feature extraction comes the prediction technique. There are many different models for predicting future actions, some gener-ate future representations of the images while some use the labels to build up relations between action classes.

Recurrent neural networks (RNNs)

RNNs (Recurrent Neural Networks) are different to feed forward NNs, the inputs are not independ-ent of each other, for a sequence of data xt = x0 to xn, with t being the step, RNNs will calculate the hidden state of h(t) by using the current input + the previous hidden state,

$$h(t) = f(Ux(t) + Wh(t-1))$$

These hidden states act as the memory of the RNN. Works such as [4], propose an RNN with an HMM (Hidden Markov Model) to predict actions in a time range of up to 5 minutes. This paper, as well as [5] [6], will be spoken about in more detail later. [7] Proposes an RNN for predicting the mo-tion of humans. This model uses an RNN with a GRU (Gated Recurrent Unit) as "GRU, do not re-quire a spatial encoding layer". This allows them to train much faster than with an LSTM (Long Short-Term Memory). Similar works like [8] [9] [10], build upon [7]'s use of RNN and human skele-tons to predict future representations. These works divide

the human skeleton into five groups. One thing [7] highlights is RNN's incapability to recover from their own mistakes.

LSTM and GRU

LSTMs were designed to overcome the vanishing gradient problem. RNNs can only look back for approximately 10 timesteps. This issue is addressed by LSTM that are capable of learning more than 1,000 timesteps depending on the complexity of the network. They do this by enforcing constant error flow-through constant error carrousels within special units [11], called cells. LSTMs have gates that can learn which data in a sequence is relevant and can choose to keep it in the cell state or throw it away, by passing along relevant data through the cell state, information from earlier steps can make it to later steps. The three gates LSTM has the forget gate, update gate and output-gate. Various works such as [12] use LSTMs to make their predictions. This paper as well as [5] [6] [13], will be spoken about in more detail in later sections.

[8] Proposes a GRU to improve the performance of the architecture. GRU is similar in structure to an LSTM but has a simpler structure by combining the three gates from LSTM into two gates: updat-ing gate and resetting gate. The structure of the architecture is bidirectional RNN which consists of 300 GRU cells. It feeds the data to an RNN, after the RNN comes the batch normalization layer. Batch normalization is a training technique that standardises and normalises the inputs [14] to keep the data in the same scale. As the input goes through multiple layers and activation functions the data is transformed, this can lead to internal covariate shift in the data. Work such as [15] use a GRU, and will be explored in more detail later.

CNN

[16] Proposes a CNN architecture that can predict future frames from unlabelled videos. This archi-tecture is appropriate for unsupervised learning as it doesn't require the ground truth for predictions and future frame representations have shown promise in a large range of tasks. Another method ex-tends an RNN to create an rCNN (recurrent CNN) to predict future frames, the architecture is able to complete motions from observed frames but the motion is slowed down and the model eventually converges to a still image [17]. [16] uses a deep regression network, as the complexity can easily be expanded and can train on large data. Using regression, which for one input can have multiple out-puts because of the ambiguity of seen frames.

Datasets

The data set used by an architecture to train is an important factor in its prediction capabilities. Some data sets have long sequences and large variations in the actions that are being performed. Data sets tend to also be highly controlled and use a sequence of scripted actions recorded in one environment using only a few viewpoints, which can lead to less generalisation to the real.

The data set used to measure the accuracy of a model is an important component in how well the model performs. Some datasets have long sequences and large variations in the actions performed, while some have shorter videos and

a much smaller sample size. Different data sets are made for dif-ferent purposes. The breakfast dataset [18], uses real world recording setup so that the dataset expresses real world conditions

50salads [19] and breakfast are the data sets that will be used in this project. I chose these two da-tasets to see how the model will perform when a smaller dataset is being used versus a larger dataset. Furthermore, 50 salads is in a controlled environment, whereas the breakfast dataset is in an uncon-trolled environment, which may affect the ability of the prediction models to predict actions. The average video length in the breakfast data set is 2.3 minutes, while the average video length in the 50 salads data set is 6.4 minutes. In both datasets, the longest video is ten minutes long.

Label representation

The ground truth is passed in as a sequence of annotations for models that use labels, and these anno-tations describe what is happening through each frame or a range of frames for an action. The model is fed the ground truth during training. For predictions, the models will generate a file that is similar to the ground truth and describes the action at a specific time. This can be used to evaluate how well the model predicts by comparing the predictions with the ground truth labels of the unseen testing set.

Feature representation

We use the provided TSN-rgb [3], which the authors of Sections.4.4 [12] have pre- trained on the epic kitchens dataset which uses egocentric videos. This information is then stored in a Lightning Memory-Mapped Database (LMDB), which allows for concurrent writes and reads using transaction management.

Models

This section outlines the models used in this study. There are seven models in total that will be com-pared, and each will be described in detail. The models can be divided into two groups: those that make predictions based on video features coupled with ground truth annotations and those that make predictions based on ground truth labels only. The following sections use ground truth labels: Section.4.1, Section.4.2 and Section.4.3. The following sections use features: Section.4.4, Section.4.5 and Section.4.6.

When will you do what? - Anticipating Temporal Occurrences of Activities

An RNN network and a CNN network are used in this paper. The goal of this project was to be able to predict activities that would occur over a long period of time (up to 5 minutes). To be able to accu-rately predict the classes, their order, and the duration of the actions' start and end. The RNN model should be able to predict actions recursively using previous output as input. The CNN model tries to predict actions all at once.

Pipeline

The RNN system predicts the output in a recursive manner. The RNN predicts the remainder of the last action as well as the next action [4]. This is done again and again until the video ends. The pipe-line for the RNN architecture works as follows:

The network is fed with a sequence of (length,1-hot class encoding)-tuples as input. This is fed into a batch generator to create the batches for training. Individual files are chosen for testing, and predic-tions are made on each one individually.

- The network uses the input to predict three elements: the last observed segment's remaining length, the label for this segment, and the length of the next segment.
- The prediction is combined with observed segments to create a new network input.
- To produce the next prediction, the new input is forwarded through the network.
- The final result is obtained by repeatedly forwarding the previously generated prediction until it predicts the desired number of frames [4], as shown in Fig. 1.

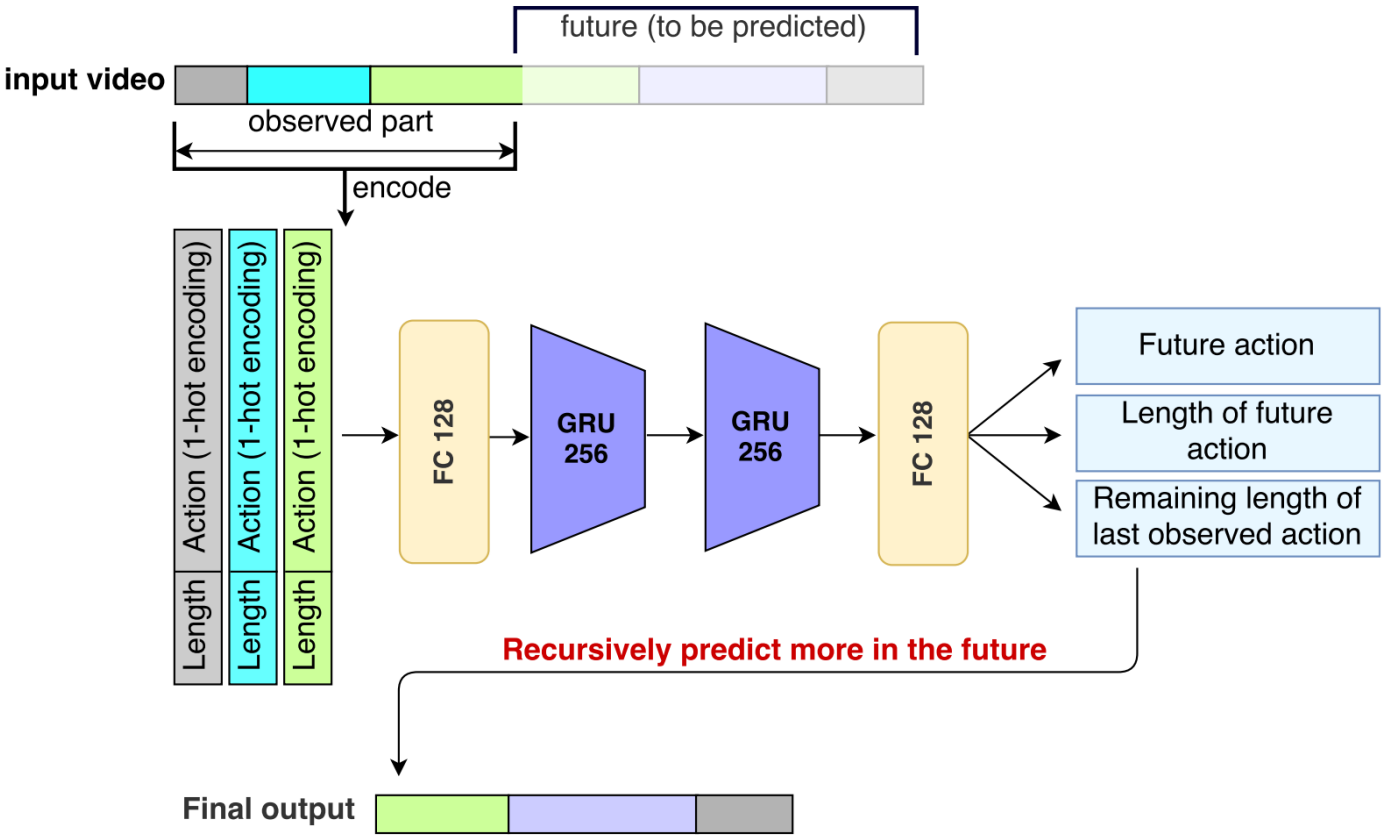


Figure. 1. Architecture of the RNN system [4]

The RNN model uses two stacked layers of 256 gated recurrent units. At the input and output, the RNN model employs two stacked layers of 256 gated

recurrent units and fully connected layers. To ensure positive length outputs, rectified linear units are used as the output layer for both length pre-dictions and remaining length.

A softmax layer is used to make label predictions [4]. The CNN system aims to predict all actions in one single step. The pipeline for the CNN architecture work as follows:

- The input labels are encoded into a matrix with columns for classes and rows for action segments.
- The matrix is filled in the order in which the actions take place [4] as shown in Fig.2.
- The matrix is then forwarded through a CNN [4].
- The output is reshaped and normalised after the connected layers to produce a new matrix of the same size as the input matrix.
- For temporal smoothing, a 1D Gaussian filter is applied to each column. After that, a function is applied to each row to convert it back into sequence labels [4].

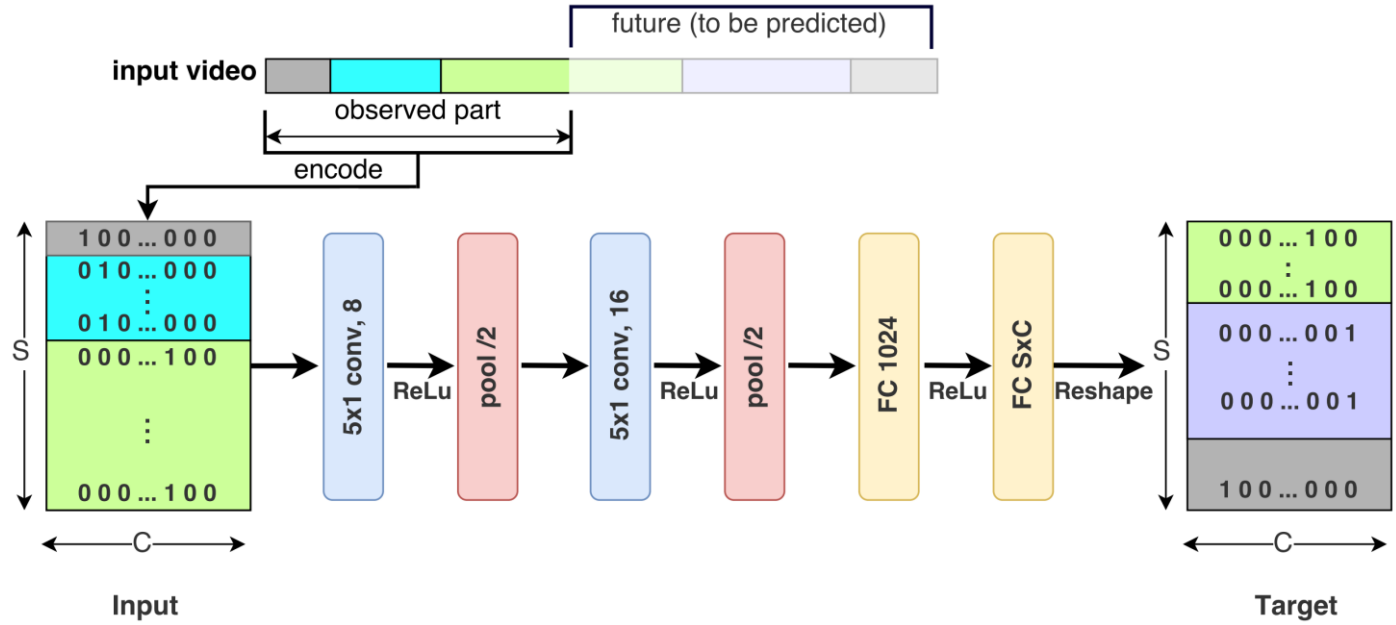


Figure. 2. Architecture of the CNN system [4]

Weakly-supervised dense action anticipation

The aim of this paper was to develop a model that could predict future actions based on incomplete action and duration labels in video sequences. The aim was to predict future sequences from videos that only had specific events labelled, as well as to predict future sequences from videos that did not have duration information. This would allow for prediction without the need for full annotations. Creating annotations is a time-consuming task. To do so, the model will learn on a small set of fully labelled data and weak labels, in which the video segment is only annotated with the first action class of the expected sequence, as shown in Fig. 3a. This can help reduce labelling time by requiring only a single action class label rather than all of the labels in the sequence [5].

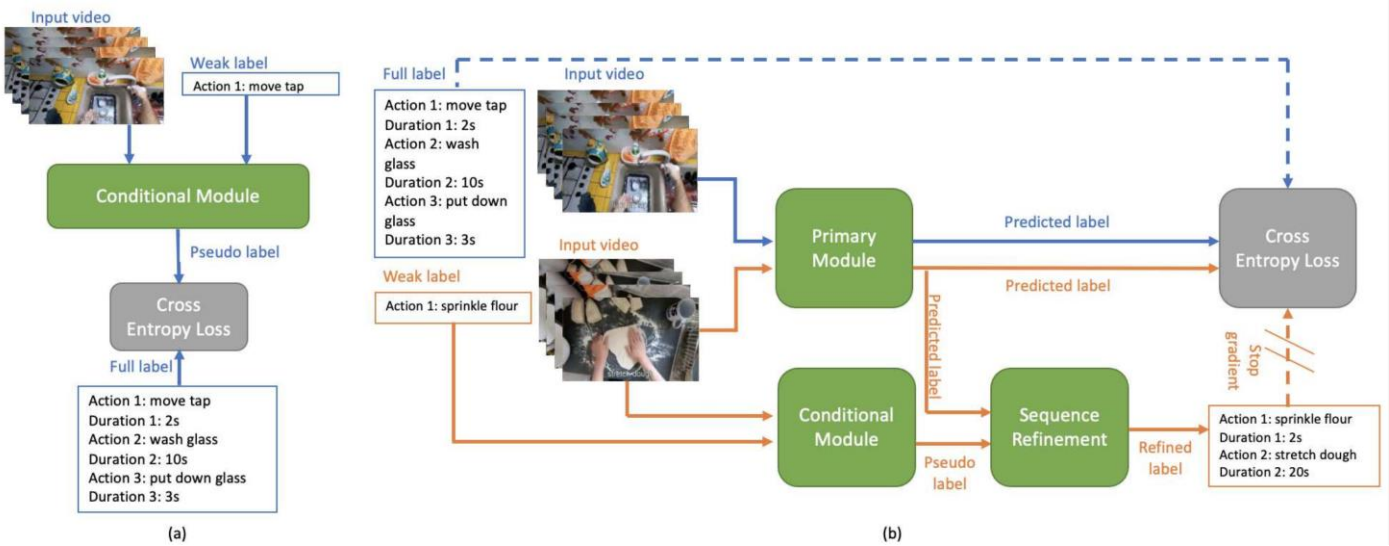


Figure. 3. Architecture of the system that include the primary module, conditional module and sequence refinement [5]

Pipeline

The model has three components. A primary module is used during inference, a conditional module is used to generate pseudo labels, and a sequence refinement module is used to refine the estimated pseudo-labels [5].

Conditional module

The conditional module is a support component that has been trained to generate pseudo labels for the weak set. The conditional module remains fixed after training. This component uses a CNN as part of its Temporal Aggregation Block (TAB) blocks and a LSTM for sequence prediction [5].

Primary module

The primary module, which is used for inference, predicts the future action and duration of a se-quence given an input video. During training the module attempts to minimise a loss using the la-belled ground truth and the refined pseudo-labels. This component has a similar layout to the condi-tional module and uses a CNN as part of its TAB blocks and a LSTM for sequence pre- diction [5].

Sequence refinement module

The pseudo-labels from the conditional module are refined using the sequence refinement module. To produce a refined prediction label, this module combines the predicted labels from the primary module with the estimated pseudo-labels from the conditional module. Because the conditional module is trained on only the weak labels, which is a small set, there is a risk of confirmation bias, according to the authors. To mitigate this, a refinement module is used that uses input from both the primary and conditional modules. This component uses multiple linear layers [5].

This model works by feeding one label into the system, then using an auxiliary model to generate pseudo-labels for the remaining labels in the sequence. The conditional model is an auxiliary model that is trained with a small set of fully labelled labels to generate pseudo-labels, which are then used on a larger weakly labelled training set. After that, the pseudo-labelled weak data is used to train the

primary anticipation module, which will be used to make predictions. An additional attention module accounts for the correlations between actions and is used to predict the duration of the actions [5].

Learning to abstract and predict human actions

The aim of this project was to look at action anticipation in the same way that humans do when plan-ning tasks. This was done because traditional methods that only look at the sequential properties of action sequences can cause earlier tasks to fade and lead to long-term prediction error accumulation. A more human way of planning processes is used to approach this issue. This process begins with high-level tasks and progresses to more detailed sub tasks. The authors created Hierarchical Encoder Refresher Anticipator (HERA) for activity anticipation to accomplish the above tasks. The HERA model is made up of three networks: one that encodes the past, another that refreshes the states, and another that decodes the future (as shown in Fig. 4.).

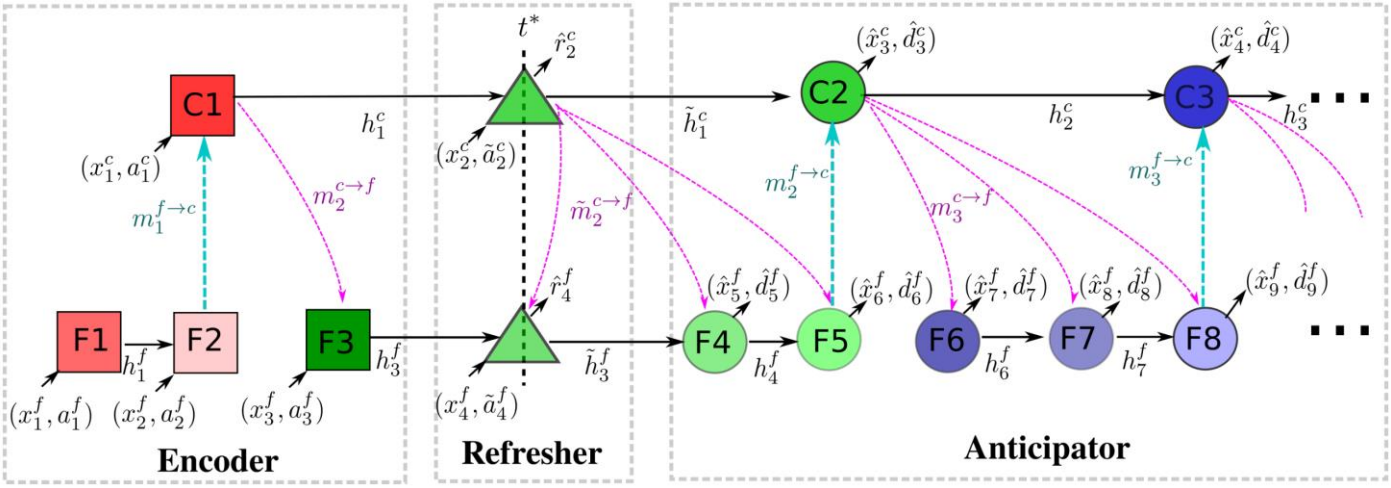


Figure 4. Architecture of the HERA system (Square blocks are Encoder GRU cells. Triangles are re-freshers. Circles are anticipators [6]

Pipeline

The HERA model can have multiple levels. We will be focusing on HERA with two levels. When there are two levels, the first level is coarse activities (Cx) and the second level is fine actions (Fx). An example of Coarse level activity would be ”eat”, which would have fine actions of ”starter”, ”main course” and ”desert”.

Encoder

The encoder’s job is to create multi-level representations of the recent video snippets, which the re-fresher and anticipator then use [6]. The encoder and anticipator have an identical architecture of two levels of RNN’s. The input is rolled out by each unit. The top level is used for coarse activities. It takes the coarse labels and their duration as input, which is then encoded into a matrix. The lower level is for fine activities, which does the same operation as the coarse level. These levels work asyn-chronously.

Anticipator

The input from the previous prediction step is fed back to the anticipator. Using a multi-layer per-ceptron, the hidden state at both levels is used to infer the label and duration of the next action. The predictions are given out in stages until the entire video has been watched. The coarse level deter-mines the total task duration, while the fine level concludes when the parent coarse level length pre-diction is reached.

At the interruption point t∗, where one prediction ends and another begins, the system connects the encoder and anticipator [6]. The encoder's predictions are sent to the anticipator if the interruption occurs at the end of an action boundary. If the interruption occurs in the middle of an action, leaving an unfinished action, the refresher is used to collect all data and predict the remaining length of the interrupted action using a multi-layer perceptron.

What would you expect? Anticipating egocentric actions with rolling-unrolling LSTMs and modality attention

The aim of this paper was to do egocentric video action anticipation and recognition. The goal of the model is to predict actions for different anticipation times, such as 2 seconds before they happen. The R-LSTM (Rolling LSTM) is in charge of recursively encoding a video snippets so that the past can be summarised. When the method must anticipate actions, the "Unrolling" LSTM (ULSTM) takes over the RLSTM's hidden and cell states and makes future predictions [12].

The processing is divided into two stages: an encoding stage for S_enc time-steps and an anticipa-tion stage for S_ant time-steps. The model summarises the semantic content of the S_enc input video snippets without making any predictions in the encoding stage, whereas in the anticipation stage, the model continues to encode the semantics of the S ant input video snippets and produces S ant action scores that can be used to perform action anticipation [12]. The two stages can be seen in Fig. 5.

Two LSTMS are used in the system. One LSTM is used to encode past observations, while the other is used to predict future actions. The RLSTM is in charge of encoding previous observations and summarising what has occurred up to that point in time. The ULSTM focuses on anticipating future actions based on the RLSTM's cell vectors.

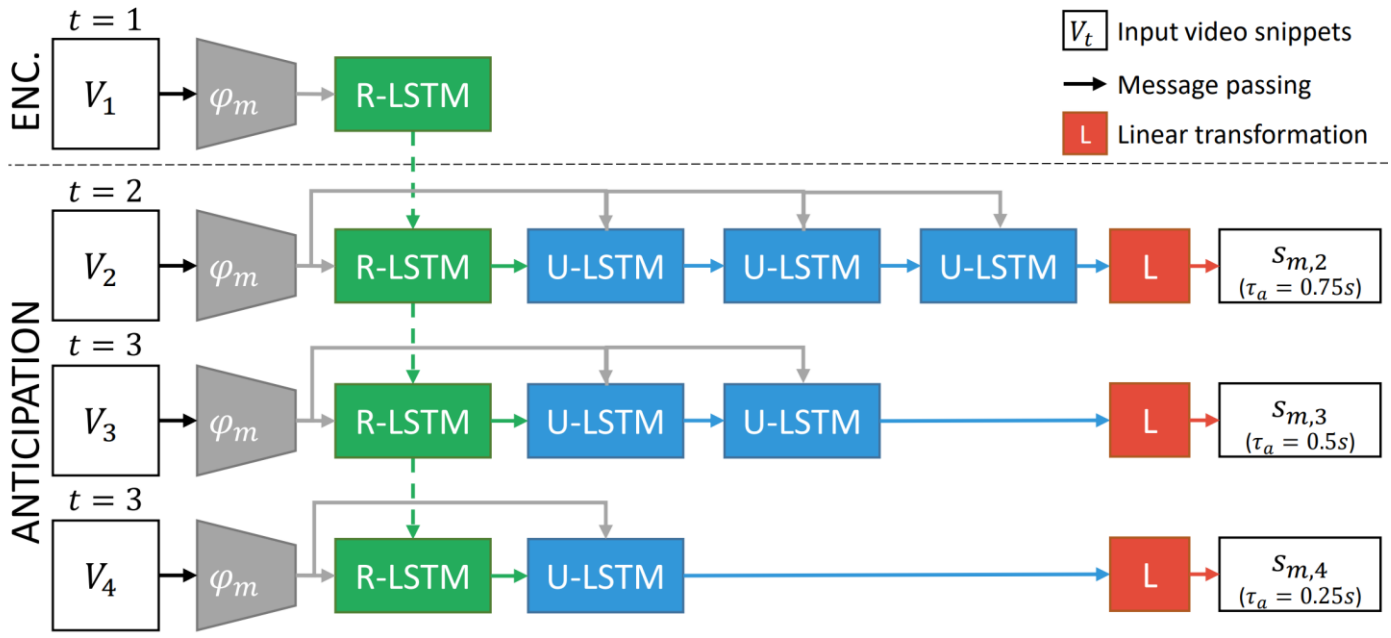


Figure. 5. Example of RULSTM modality branch, with S_enc = 1 and S_ant = 3 [12]

Pipeline

We will focus solely on the RGB pipeline because we're only using RGB. The RGB branch com-putes the feature vector fm,t by using a batch normalised inception CNN to extract features from the last frame of a video snippet. This yields action representation of the input frame, which the RLSTM can use to summarise what happened previously. There are several identical branches that analyse the video based on the modalities, such as RGB, optical, and I3D. Fig. 5. shows an example branch of the system. For this example, φm will represent RGB videos. In this example, a feature vector fm,t is fed into the system's mth branch. In the encoding section, the system recursively encodes its semantic content.

In the anticipation stage, the ULSTM is used to make future predictions. It uses the hidden states and cells of the RLSTM and loops over the current video snippet until it reaches the beginning of an action. Sequence completing pre-training (SCP) is used in the system so that the two LSTMs can spe-cialise in encoding and anticipating, respectively [12]. The ULSTM's connections are changed during SCP to aid in training. The RLSTM concentrates on summarising previous representations rather than attempting to predict the future.

Temporal aggregate representations for long-range video understanding

The system is designed to work with a variety of inputs, ranging from low-level visual features to high-level semantic labels, and to efficiently integrate recent observations with long-term context [13].

Pipeline

A non-local block (NLB), a coupling block (CB), and a temporal aggregation block (TAB) make up the architecture. As shown in the Fig. 6., NLB are nested in CBs, and CBs are nested in TABs. The system's flow is as follows: the NLB receives recent and spanning features, the CB's output is sent to the TAB, which

can be used for either next action anticipation or dense anticipation, and the TAB's output is chained together.

The NLB's job is to capture relationships between spanning and recent snippets. This allows the system to capture long-range dependencies. This component uses a 2 CNN's to achieve this. The coupling block uses concatenation and a linear layer to connect the recent and spanning snippets. These yield fixed-length representations that account for the beginning point of a recent snippet as well as the length of the spanning snippet. This component uses two NLB, one for past and one for recent snippets.

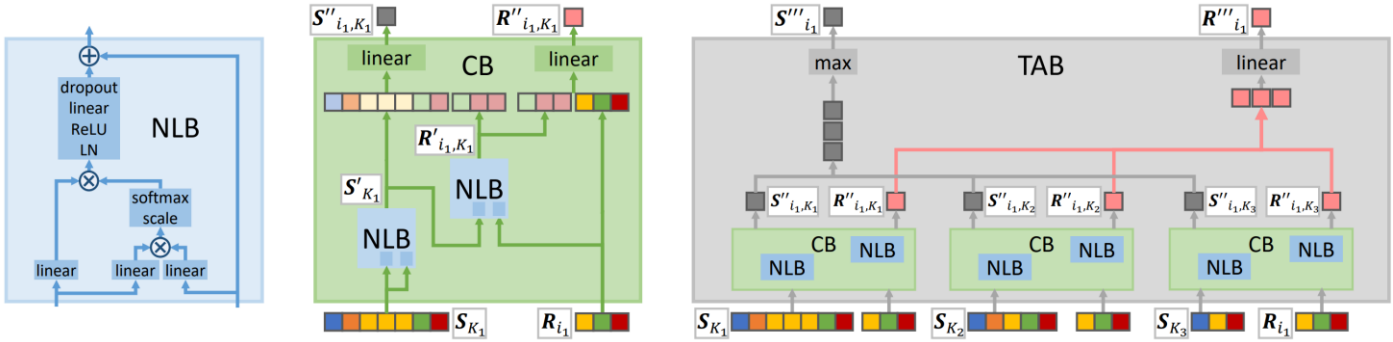


Figure. 6. Architecture of NLB, CB and TAB used for generating temporal aggregate representations [13]

As shown in the TAB section of the Fig.6., by grouping outputs from multiple CBs, the final repre-sentation for recent and spanning past is computed. The final spanning representation is a max over all the spanning fixed length representations. TAB outputs a representation that aggregates and en-codes both recent and long-range past to make temporal aggregate representations [13]. This compo-nent uses three coupling blocks.

The model then predicts actions based on these representations. A classification layer (linear + softmax) can be used directly with the temporal aggregate representations. To accomplish this, the TAB layer's spanning fixed length representations are concatenated and passed through a classifica-tion layer [13]. The sequence predictor is used to foresee what will happen next. It concatenates all recent and spanning temporal aggregates, as well as the classification layer's outputs, and feeds this output to a linear layer before sending it to a single layer LSTM. The label and duration are outputted by the LSTM (See Fig. 7.).

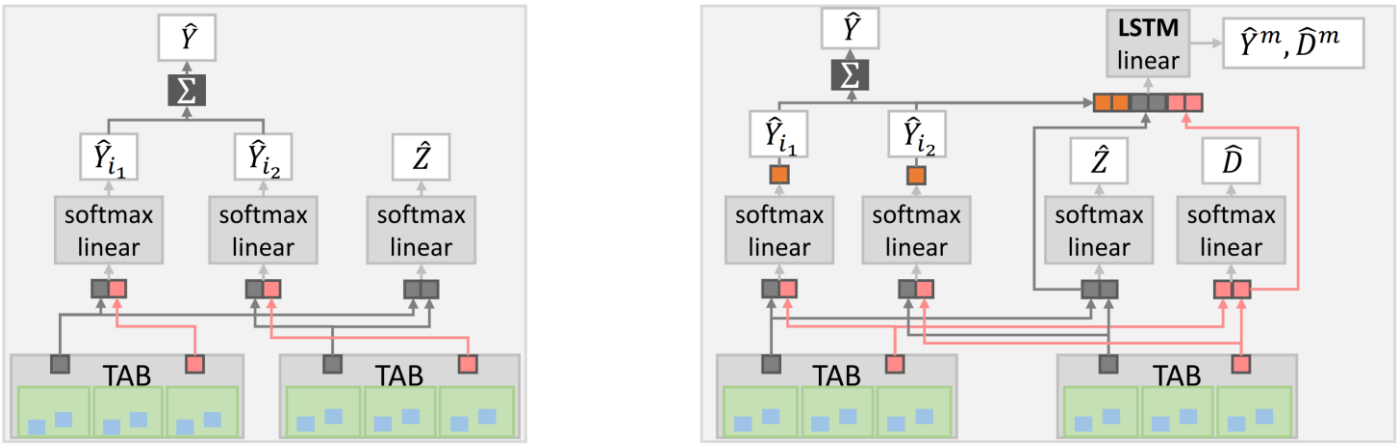


Figure. 7. Architecture of anticipation model. Classification model (left) and sequence prediction (right) [13]

Self-regulated learning for egocentric video activity anticipation

The aim of this paper is similar to that of Section.4.4 and Section.4.5. The aim of this paper was to do egocentric video action anticipation and recognition. The goal of the model is to predict actions for different anticipation times, such as 2 seconds before they happen.

Pipeline

This architecture has three primary components, as indicated in Fig. 8.

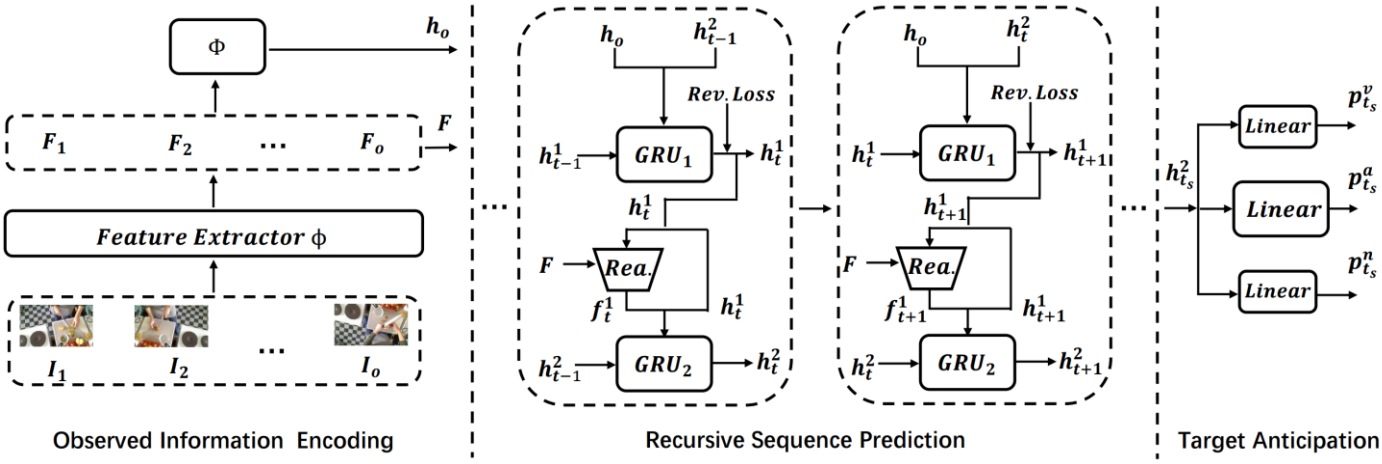


Figure. 8. Architecture of the system [15]

Observed information

Encoding When a video is input into the system, a feature extractor and an aggregation function are used to extract the features and hidden representation at the final seen bit of the clip. To encode the video, an RNN is used, yielding F and h0, which are the feature and hidden representations, re-spectively [15].

Recursive sequence prediction

A GRU layer is utilised in the recursive sequence prediction stage to build a new representation, which is then fed into the Rea layer [15] to generate another new representation, which encodes the current video content. The video

representation is fused with the hidden representation after it has been processed by these two modules. The procedure is continued until the anticipated time-step is achieved; for example, if the anticipation time-step was 2 seconds, the operation would be repeated until the 2 second mark was reached.

Target anticipation

The final representation is obtained after the recursive pre- diction. The target anticipation stage uses semantic information to improve the final representation and outputs three values for the action, objects, and expected activity probability distribution [15].

Results and evaluation

In this section, I discuss the experiments I carried out for the models discussed in Section.4. It is im-portant to note that all the training and experiments were done on Google Colab's GPUs. After the models have been trained, the next step is to conduct experiments and analyse the results to deter-mine how well the project met its objectives. The datasets were divided into three categories: train-ing, testing, and validation, with validation accounting for 20% of the training data. This was done because one of the models needs validation data to run correctly. Only the training and testing data are used in the remaining models. Some of the videos were removed from the splits because there were no corresponding RGB videos to do feature extraction with, as stated in Section.3, so they will not be included. Each model will run with the default parameters, which are the parameters specified in the paper, as stated in Section.4, so that we can compare the models to the test paper as well as each other.

We'll divide the results and experiments into two groups because the models can be split in two. One of the models, however, will be evaluated separately; this model is a recent model that aims to progress action anticipation research into a new field, which is why it has been included here to demonstrate what works that deviate from the standard look like.

Metrics

The original literature review outlines a few of the metrics that could be used to evaluate the mod-els. We have decided to go for the following:

- Accuracy
- Mean over all classes / balanced accuracy score. Average accuracy obtained on each class
- Precision

$$\frac{TruePositives}{True\ Positives\ +\ False\ Positives}$$

- Recall

$$\frac{TruePositives}{True\ Positives\ +\ False\ Negatives}$$

- F1 score

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

All of the metrics discussed above are used in models that use ground truth labels as input. It was simple to add more metrics because these methods return a sequence of y pred. Due to the way fea-ture extraction models calculate accuracy, they will only use accuracy and recall.

Results

The results of the experiments can be found in the Tables 1 and 2. The graphs 9 and 10 make it easier to understand the results.

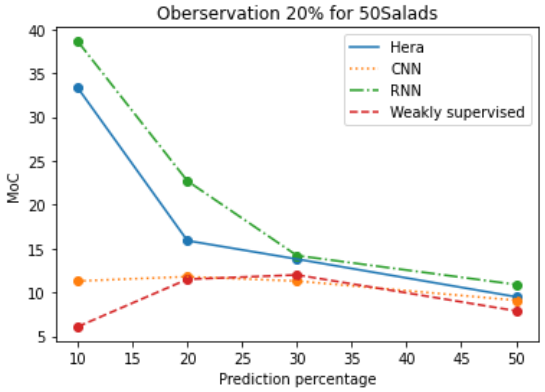
Table 1. Results of the feature extraction methods, to one decimal place

Ta(s) %	Metric	2	1.75	1.5	1.25	1	0.75	0.5	0.25
50 Salads									
SRL [15]	Accuracy	23.9%	23.5%	24.5%	25.3%	25.2%	26.8%	28.6%	30.9%
	Recall	22.8%	22.0%	22.8%	23.4%	23.5%	24.0%	26.0%	27.8%
RULSTM [12]	Accuracy	28.1%	27.8%	29.4%	29.4%	27.5%	28.3%	29.6%	30.4%
	Recall	26.1%	26.0%	27.8%	27.3%	25.2%	25.8%	27.2%	27.6%
Breakfast									
SRL	Accuracy	29.4%	29.5%	29.5%	28.9%	29.4%	30.1%	30.8%	31.6%
	Recall	11.9%	12.2%	11.8%	11.2%	11.9%	12.2%	11.9%	11.1%
RULSTM	Accuracy	31.8%	31.8%	32.3%	32.3%	32.4%	32.6%	32.0%	33.3%
	Recall	13.5%	12.8%	12.9%	12.8%	12.6%	12.1%	11.7%	11.8%

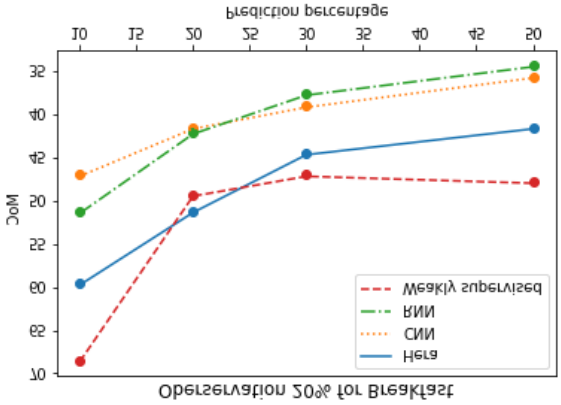
Table 2. Results of ground truth label models, to one decimal place

Observation %		20%				30%			
Prediction %	Metric	10%	20%	30%	50%	10%	20%	30%	50%
50 Sal ads									
Weakly supervised [5]	Accuracy	10.5%	18.7%	10.7%	8.4%	10.1%	18%	13.0%	9.6%
	MOC	6.1 %	11.5%	12%	7.9%	5.0%	16.5%	7.9%	9.9%
	Precision	8.8 %	17.4%	2.7%	11.4%	8.4%	6.8%	3.6%	3.7%
	Recall	10.5 %	18.7%	10.7%	8.4%	10.1%	18%	13.0%	9.6%
	F1	9.6 %	18.0%	4.3%	9.7%	9.2%	9.9%	5.6%	5.3%
HERA [6]	Accuracy	24.8%	9.9%	7.1%	4.7%	17.2%	2.9%	7.1%	3.9%
	MOC	33.3%	15.9%	13.8%	9.5%	25.8%	11.2%	7.2%	3.4%
	Precision	38.3%	23.9%	23.9%	21.7%	28.3%	23.9%	20.6%	18.9%
	Recall	27.5%	12.9%	8.4%	5.5%	27.1%	11.1%	9.1%	4.5%
	F1	30.4%	16.5%	12.3%	8.5%	25.8%	14.5%	12.3%	7.2%
CNN [4]	Accuracy	38.9%	32.4%	26.6%	20.3%	17.4%	17.1%	18.2%	13.6%
	MOC	11.3%	11.8%	11.3%	9.1%	12.5%	11.2%	10.3%	6.0%
	Precision	27.2%	18.8%	14.3%	6.5%	8.6%	12.0%	10.0%	4.2%
	Recall	27.2%	13.5%	13.1%	8.3%	16.7%	10.5%	10.1%	6.5%
	F1	27.2%	14.9%	12.6%	7.2%	11.4%	11.2%	10.1%	5.1%
RNN [4]	Accuracy	32.1%	16.3%	10.9%	6.5%	38.7%	21.7%	14.5%	9.3%
	MOC	38.6%	22.7%	14.2%	10.9%	26.6%	15.5%	10.4%	4.7%
	Precision	18.0%	4.8%	2.1%	0.7%	23.7%	7.9%	3.5%	0.9%
	Recall	32.1%	16.3%	10.9%	6.5%	38.7%	21.7%	14.5%	9.3%

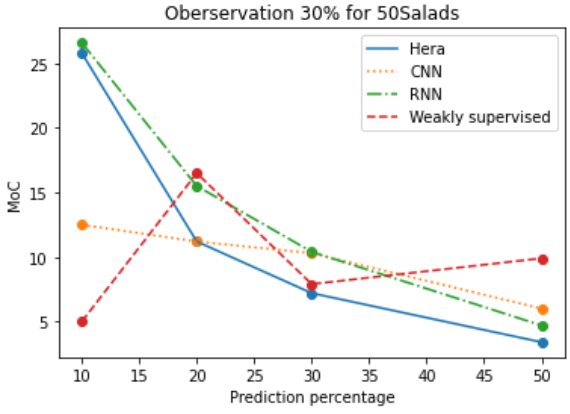
	F1	20.9%	6.8%	3.3%	1.3%	28.0%	11.0%	5.4%	1.7%
					Breakfast				
Weakly supervised	Accuracy	68.7%	51.3%	45.1%	46.6%	71.8%	63.1%	56.0%	55.1%
	MOC	68.5 %	49.3%	47.0%	47.8%	72.7%	65.4%	53.8%	55.4%
	Precision	65.7 %	52.1%	52.3%	50.2%	70.6%	60.3%	62.0%	65.4%
	Recall	68.7%	51.3%	45.1%	46.6%	71.8%	63.1%	56.0%	55.1%
	F1	67.2 %	51.7%	48.4%	48.3%	71.2%	61.7%	58.8%	59.8%
HERA	Accuracy	64.6%	50.7%	42.9%	36.3%	69.9%	57.7%	46.4%	36.7%
	MOC	59.6%	51.2%	44.5%	41.5%	71.8%	57.4%	46.5%	37.8%
	Precision	77.2%	63.6%	54.3%	48.0%	78.7%	67.5%	56.0%	47.4%
	Recall	69.0%	61.6%	55.3%	48.4%	75.7%	67.7%	59.5%	51.6%
	F1	70.6%	59.7%	52.1%	45.9%	75.2%	65.3%	55.5%	48.0%
CNN	Accuracy	74.4%	67.5%	66.7%	65.1%	75.9%	73.0%	70.8%	70.9%
	MOC	47.0%	41.5%	39.0%	35.6%	53.5%	46.3%	44.4%	42.8%
	Precision	76.8%	73.8%	72.3%	70.8%	72.8%	74.8%	73.6%	72.1%
	Recall	74.4%	67.5%	66.7%	65.1%	75.9%	73.0%	70.8%	70.9%
	F1	75.6%	70.6%	69.4%	67.8%	74.3%	73.9%	72.2%	71.5%
RNN	Accuracy	66.6%	61.1%	58.3%	56.6%	77.2%	68.9%	67.6%	62.6%
	MOC	51.3%	42.1%	37.6%	34.3%	61.3%	51.1%	48.5%	42.7%
	Precision	76.5%	69.3%	66.8%	66.9%	79.2%	77.4%	76.6%	74.0%
	Recall	66.6%	61.1%	58.3%	56.6%	77.2%	68.9%	67.6%	62.6%
	F1	71.2%	65.0%	62.3%	61.3%	78.2%	72.9%	71.8%	67.9%



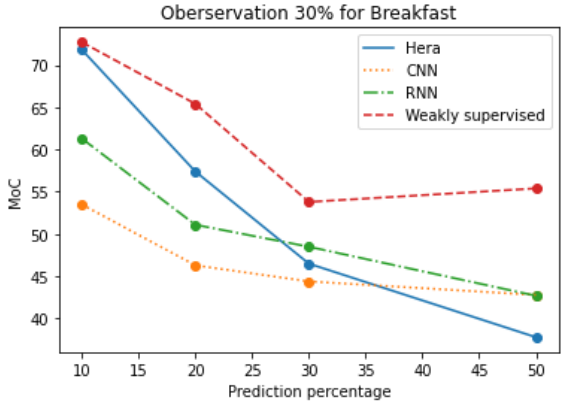
(a) 20% Observation 50 Salads



(b) 20% Observation breakfast

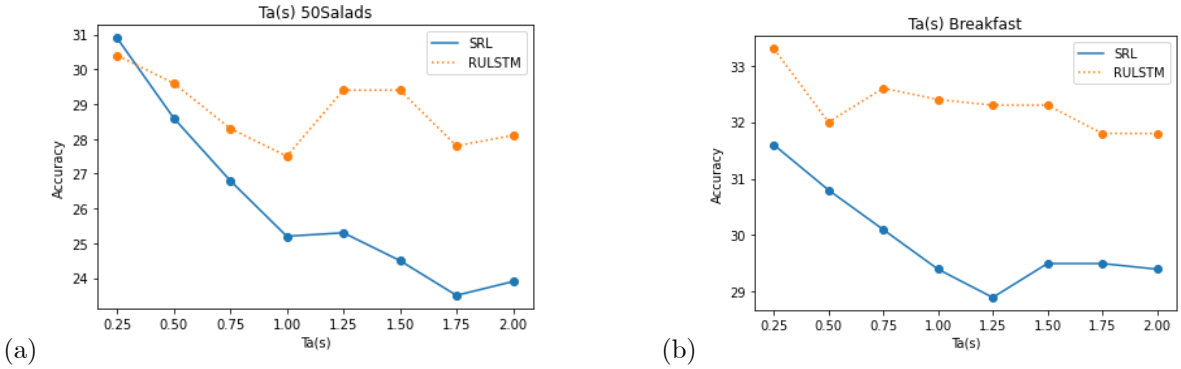


(c) 30% Observation 50 Salads



(d) 30% Observation breakfast

Figure. 9. Plots of different observation percentages



(a) (b)

Figure. 10. Plots of different anticipation times and the accuracy

Evaluation

Datasets

The data sets chosen were a good choice for demonstrating the importance of the data set used for action anticipation. There are far fewer videos in the 50 Salads data set, which must have hampered the models' ability to learn about the data. Precision and accuracy suffered as a result of a lack of data to learn from.

For example, for 50 salads, the average accuracy among the ground truth models was 20.9 percent, while for breakfast it was 73.7 percent, resulting in a difference of 52.9 percent. In terms of MOC, some models, such as weakly supervised, which performed best on breakfast, performed the worst on 50 salads. Except for temporal aggregate representations, feature extraction models perform slightly better on the breakfast data set than on the 50 salads data set. For the breakfast dataset, temporal aggregate representation has a similar accuracy compared to the ground truth models, while for 50 salads, it has a similar but better result than the other two feature extraction models.

Models

- Weakly supervised. For 50 Salads, this model performs poorly, and the metrics fluctuate depend-ing on the prediction percentage. This demonstrates that the model did not have enough time to properly learn. Because this model uses a set of weak labels, without enough data to learn on the weak set, which is already a small portion of the data, the model's predictions become almost ran-dom. This model, on the other hand, performs exceptionally well in terms of MOC for the break-fast dataset, which is a much larger data set, with a peak MOC of 72.7 percent, the highest in the study. When compared to the original paper, the model's experiments for 50 Salads are less than the reported average of 35.4 percent. The model is very similar to the reported average of 61.3 per-cent for the breakfast dataset.
- HERA. When only predicting recent events, this model performs well for 50 salads, but then rap-idly deteriorates. When the model has to predict even further into the future for both datasets, it begins to deteriorate, though the deterioration is much slower for the breakfast data set. HERA's MOC range is the widest among the ground truth models, indicating that the model has difficulty predicting in much

longer time frames. When compared to the original paper's model, the model performs very similarly to the original paper's model. The average prediction percentage for 20 per-cent observation for F1 score, which is the metric used in the paper for HERA, was 51.5 percent, which is similar to the 57.1 percent achieved in this study. However, when comparing MOC, the MOC from the paper was 63 percent, whereas we only got 49.2 percent in this study. The original MOC score for 30 percent observation was 62.8, and this study obtained 53.4 percent.

- CNN. In terms of MOC, this model, like the others, does not perform well for 50 salads, but the range of predictions is very consistent. This model achieves the highest accuracy for 50 salads, with 38.9 percent accuracy based on 20 percent observation and 10 percent prediction. This model performs consistently within a range of 10 percent +- for breakfast. When compared to the original paper, the MOC was 47 percent for breakfast and 25.1 percent for 50 salads, based on the average of the different prediction percentages for 20 percent observation. This study's results were 40 percent and 10.9 percent, respectively.
- RNN. When only predicting recent events, this model performs well for 50 salads, but then rapidly deteriorates. When the model has to predict even further into the future, it begins to deteriorate, similar to the CNN created by the same people, though this deterioration is much slower for the breakfast dataset. When it comes to predicting 50 percent, however, this model has the lowest pre-cision of all the models. However, for the breakfast dataset, this model has the second highest ac-curacy score. When compared to the original paper, the MOC was 40 percent for breakfast and 28.9 percent for 50 salads, based on the average of the different prediction percentages for 20 percent observation. 41.3 percent and 21.6 percent, respectively, were achieved in this study.
- SRL. When predicting in different time steps, this model is very consistent, with the best accura-cy of 30.9 percent for 50 salads. When predicting 0.25s into the future, the model performs better for the breakfast dataset, achieving 31.6 percent. When compared to ground truth models, this model and other feature extraction methods predict in a much shorter time, making comparisons difficult. The original paper does, however, include a table with different observation percentages, similar to Table.2. The 50salads and breakfast dataset results for different anticipation times, similar to the tests I conducted, are not included in the original paper.
- RULSTM. When using a larger dataset, this model outperforms SRL, achieving 33.3 percent accuracy when predicting 0.25s into the future. This paper, like SRL, does not include results for the 50 salads and breakfast datasets.
- Temporal aggregate representations. This model outperforms the other two feature extraction methods in terms of accuracy, achieving 60.6 percent for break- fast and 31.6 percent for 50salads. It has a lower recall for 50 Salads, but a higher recall for breakfast than the other two feature extraction methods. The original paper does, however, include a table with different observa-tion percentages, similar to

Table.2. This model could be changed to predict with different observation percentages.

Table 3. Results for temporal aggregate

Dataset	Accuracy	Recall
50 Salads	31.6%	24.4%
Breakfast	60.6%	20%

Conclusion

In conclusion, in this study, we have discussed and laid out the design of the action anticipation ar-chitectures and experiments, as well as compared the models. The results were tabulated, and graphs were created to show the model's quality. As a result, several methods were implemented, evaluated, and the results were displayed. There is a general way of augmenting a data set for use with these methods. Overall, this work sets out to create a survey of third-person action anticipation models. The motivation behind the survey is that, while this task is in a niche, increased research interest has led to the creation of many new publicly available models. So, an evaluation of current architectures focuses on the type of model at the core of creating predictions, which may allow researchers to use more suitable models within this task. Potential avenues for future research could be creating addi-tional datasets and using generative AI.

Declarations

The datasets used in this study are ''Breakfast" [18] and ''50 Salads" [19] that are publicly available. No funding was received for conducting this study. The authors have no competing interests to de-clare that are relevant to the content of this article.

References

1.Jain A, Singh A, Koppula HS, Soh S, Saxena A. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In2016 IEEE International conference on robotics and automation (ICRA) 2016 (pp. 3118-3125). https://arxiv.org/pdf/1509.05016

2.Zhen X. Feature extraction and representation for human action recognition (Doctoral disserta-tion, University of Sheffield). https://etheses.whiterose.ac.uk/5141/1/Thesis_ZhenXT_revised_final.pdf

3.Wang L, Xiong Y, Wang Z, Qiao Y, Lin D, Tang X, Van Gool L. Temporal segment networks: Towards good practices for deep action recognition. In European conference on computer vision 2016 (pp. 20-36). https://arxiv.org/pdf/1608.00859.pdf%EF%BC%89

4.Abu Farha Y, Richard A, Gall J. When will you do what? anticipating temporal occurrences of activities. In Proceedings of the IEEE conference on computer vision and pattern recognition 2018 (pp. 5343-5352). http://openaccess.thecvf.com/content_cvpr_2018/papers/Abu_Farha_When_Will_You_CVPR_2018_paper.pdf

5.Zhang H, Chen F, Yao A. Weakly-supervised dense action anticipation. arXiv preprint arXiv:2111.07593. 2021. https://arxiv.org/pdf/2111.07593

6.Morais R, Le V, Tran T, Venkatesh S. Learning to abstract and predict human actions. arXiv preprint arXiv:2008.09234. 2020. https://arxiv.org/pdf/2008.09234

7.Martinez J, Black MJ, Romero J. On human motion prediction using recurrent neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 2891-2900). https://openaccess.thecvf.com/content_cvpr_2017/papers/Martinez_On_Human_Motion_CVPR_2017_paper.pdf

8.Zhao R, Ali H, Van der Smagt P. Two-stream RNN/CNN for action recognition in 3D videos. In2017 IEEE/RSJ International conference on intelligent robots and systems (IROS) 2017 (pp. 4260-4267). https://arxiv.org/pdf/1703.09783

9.Du Y, Wang W, Wang L. Hierarchical recurrent neural network for skeleton based action recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 1110-1118). https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Du_Hierarchical_Recurrent_Neural_2015_CVPR_paper.pdf

10.Zhu W, Lan C, Xing J, Zeng W, Li Y, Shen L, Xie X. Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In Proceedings of the AAAI conference on artificial intelligence 2016 Mar 5 (Vol. 30, No. 1). https://ojs.aaai.org/index.php/AAAI/article/download/10451/10310

11.Schmidt RM. Recurrent neural networks (rnns): A gentle introduction and overview. arXiv preprint arXiv:1912.05911. 2019. https://arxiv.org/pdf/1912.05911

12.Furnari A, Farinella GM. What would you expect? anticipating egocentric actions with rollingunrolling lstms and modality attention. In Proceedings of the IEEE/CVF International conference on computer vision 2019 (pp. 6252-6261). https://openaccess.thecvf.com/content_ICCV_2019/papers/Furnari_What_Would_You_Expect_Anticipating_Egocentric_Actions_With_Rolling-Unrolling_LSTMs_ICCV_2019_paper.pdf

13.Sener F, Singhania D, Yao A. Temporal aggregate representations for long-range video understanding. In Proceedings of the IEEE European conference on computer vision, 2020 (pp. 154-171). https://arxiv.org/pdf/2006.00830

14.Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning 2015(pp. 448-456). http://proceedings.mlr.press/v37/ioffe15.pdf

15.Qi Z, Wang S, Su C, Su L, Huang Q, Tian Q. Self-regulated learning for egocentric video activity anticipation. IEEE transactions on pattern analysis and machine intelligence. 2021. https://arxiv.org/pdf/2111.11631

16.Vondrick C, Pirsiavash H, Torralba A. Anticipating the future by watching unlabeled video. arXiv preprint arXiv:1504.08023. 2015;2:2. http://www.cs.columbia.edu/~vondrick/prediction/paper.pdf

17.Ranzato M, Szlam A, Bruna J, Mathieu M, Collobert R, Chopra S. Video (language) modeling: a baseline for generative models of natural videos. arXiv preprint arXiv:1412.6604. 2014. https://arxiv.org/pdf/1412.6604

18.Kuehne H, Arslan A, Serre T. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In Proceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 780-787). https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Kuehne_The_Language_of_2014_CVPR_paper.pdf

19.Stein S, McKenna SJ. 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). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b9410401cec076baef045e83953f3ff24f25d149