A Convolutional Approach to Action Anticipation

Shelly Srivastava
190385633
Dr Lorenzo Jamone
M.Sc. in Artificial Intelligence with Industrial Experience

***Abstract*—For humans and robots to share the same workspace, the robot must be able to anticipate actions even before its completion. Recent works have shown that an action can be anticipated accurately with the help of additional information such as context when combined with limb movements. The model architectures which were used are based on Recurrent Neural Networks (RNN). This paper aims to extend the architecture of the anticipation model to convolution based networks as they are computationally less expensive. The main contributions of this paper are: (i) An architecture for action anticipation solely based on Convolutional Neural Networks (CNN); (ii) To realise the effect of different hyper-parameters on the model's accuracy. The proposed model is based on successive convolution with residual connections. The results achieved on the Acticipate dataset suggest a few parallels between an RNN and CNN based models.**

***Index Terms*—Action Anticipation, Early Action Prediction, Deep Learning, Convolutional Neural Networks**

I. Introduction

Numerous research is currently being done in the field of artificial intelligence (AI) to incorporate it in our lives (Pannu, 2015; Tong et al., 2019; Ganapathy et al., 2018). AI is rapidly evolving and spreading across many domains. One such domain is the field of human-robot collaboration (HRC) (Bauer et al., 2008).

Movement and action anticipation is required in any collaborative activity. Hoffman (2010) found that in human-robot interaction, anticipation was the key to improving the efficiency of the team. Anticipation comes quite naturally to humans, but the challenge here is to model the behaviour of anticipation for robots. A successful integration of AI in our lives, which involves collaboration with humans, is greatly dependent on the AI's capability of correctly anticipating human movements (Mainprice and Berenson, 2013). Such an AI can be seamlessly integrated into our lives.

An action anticipation task is very similar to the task of action recognition except that the former works on incomplete sequences. As described by Kong and Fu (2018), action prediction can be of two types: (i) short term prediction and (ii) long term prediction. Short term prediction (Santos et al., 2019; Schydlo et al., 2018) is based on predicting an action when given partial sequence information. Long term prediction (Ke et al., 2019) or intention prediction is based on long action sequences. It focuses on predicting the next action in the future. This paper looks into short term prediction and refers to it as action anticipation.

An action anticipation task is based on sequential data. The model has to remember and understand the past and make a prediction based on it. Therefore, the architecture employed for such models is based on recurrent neural networks (RNN). An RNN can store sequential information in their internal memory. Due to this, RNNs have many applications in the field of natural language processing (NLP) for text classification (Jang et al., 2020; Semberecki and Maciejewski, 2017), sentiment analysis (Wang et al., 2016), etc. It is also used in other fields which require the processing of sequential data (Zhang et al., 2018; Sethia and Raut, 2019). LSTM (Long short-term memory) is the most popular choice of RNN. It can remember long as well as short dependencies along a sequence.

LSTM (Hochreiter and Schmidhuber, 1997) maintains two internal states called the cell state and hidden state. There are four trainable gates which help in maintaining the internal states. Training an LSTM is slow and difficult due to the presence of the four gates. As a result, many NLP domains are making a shift to convolutional neural networks (CNNs) (Gehring et al., 2016, 2017). Previously, CNNs were predominantly used for computer vision (Girshick, 2015; Krizhevsky et al., 2012). It offers many advantages over RNNs. It has fixed-sized windows (kernels) which scans over the input to produce feature maps. A CNN is computationally less expensive than RNN as the number of trainable parameters are comparatively less. In many cases, CNNs have outperformed RNNs in NLP (Yin et al., 2017).

Following the trend, this paper introduces a new architecture of neural networks for the task of action anticipation. This implementation is solely based on CNNs and was inspired by Gehring et al. (2017). It also evaluates the performance of the new architecture based on a few hyper-parameters. To better understand the problem statement and the proposed model, the next sections will cover Related Work II, Methodology III, Experiments IV, Results and Discussion V, and Conclusion and Future Work VI.

II. Related Work

Action recognition (Ji et al., 2012; Liu et al., 2017; Ullah et al., 2017; Lee et al., 2017) and action anticipation (Furnari and Farinella, 2020, 2019; Shi et al., 2018; Rodriguez et al., 2018; Gammulle et al., 2019b) are quite closely related. Both the problems are based on making a prediction given a video sequence. As described by Kong and Fu (2018), action anticipation model is required to predict an action class before the sequence is completed. Normally such tasks require a recurrent neural network like Simple RNNs, GRUs or LSTMs (Liu et al., 2017; Ullah et al., 2017; Lee et al., 2017; Furnari

and Farinella, 2020). LSTMs are quite a popular choice for learning sequences.

Wang et al. (2019) uses a student-teacher learning framework based on LSTMs and Bidirectional LSTMs to predict early actions. The teacher model is in charge of action recognition and the student model is in charge of early action prediction or action anticipation. Sadegh Aliakbarian et al. (2017) present a multistage LSTM architecture for action anticipation using context-aware and action-aware features. It also introduces a new loss function to train the model for very early action prediction. Pang et al. (2019) utilizes a deep bi-directional dynamics network which consists of LSTMs for early action prediction.

There are a few architectures proposed for action anticipation which are not based on LSTMs. For example, Chen et al. (2018) uses deep reinforcement learning for action prediction. Gammulle et al. (2019a) uses a recurrent Generative Adversarial Network to predict future frames. The work of Cho and Foroosh (2018) is inspired by NLP and uses Temporal ConvNets to learn sequential information. It represent each frame as a "word" and the whole video as a "sentence". Action anticipation is performed using chunks of partial "sentences" (0% - 10%, 0% - 20%, etc) as input. It does not predict the action class on each frame of the video sequences.

Other works such as Schydlo et al. (2018); Santos et al. (2019) are based on collaborative tasks which involve ambiguous actions. Both works are based on the Acticipate Dataset (Duarte et al., 2018). The action anticipation is presented as a multi-class classification problem in both the works.

Schydlo et al. (2018) showed the importance of context information such as gaze for action anticipation. They used an encoder-decoder LSTM network. The encoder was used to create a context vector of the observed partial sequence. The context vector was passed to the decoder, which then predicted the coherent future, given the discrete action classes. The input to the model was the 3D motion capture information along with gaze coordinates. The main drawback of the proposed model was its inability to recognize all the action in the dataset.

Building on its work, Santos et al. (2019) proposed uncertainty based models to encompass the inherent uncertain nature of action anticipation. It proposed two versions of the LSTM model, namely Deterministic and Bayesian LSTM. It also extended the original Acticipate Dataset from 6 actions to 12 actions making it more ambiguous. It added another source of context information (object position) and proved its importance over the extended dataset. Their best model was a probabilistic model and had an average anticipation accuracy of 98.75%, given only 25% of the action sequence.

Bayesian approaches have also been used by Bütepage et al. (2017) to predict human motion. It, however, focuses on predicting future motion trajectory when given a past motion.

Like Santos et al. (2019) and Schydlo et al. (2018), this paper also looks at action anticipation as a multi-class classification problem and uses the Acticipate dataset to produce results. Santos et al. (2019) proposed an architecture which was based on two layers of LSTMs. Extending on its work, this paper aims to:

- Implement a model architecture using only CNNs
- Evaluate the effect of various hyper-parameters such as kernel size and layers of convolution on the anticipation accuracy
- Discuss the similarities between the performance of state-of-the-art LSTM model and the implemented CNN model

The next section provides more details about the implemented model.

III. Approach

The approach taken for the implementation is described in this section. It is divided into three subsections: (i) Dataset; (ii) Feature selection and embedding; and (iii) Proposed CNN architecture.

A. Dataset

Acticipate Dataset (Duarte et al., 2018; Schydlo et al., 2018; Santos et al., 2019) is used for the purpose of the paper. It was acquired to study the importance of context such as gaze, in a collaborative task. The dataset is a video which consists 240 action sequences performed by an actor. The actor had to place a red ball on the table (right, middle or left) or pass it to people sitting opposite to him. The sequences are divided into 12 action classes: "place left", "give left", "place middle", "give middle", "place right", "give right", "pick left", "receive left", "pick middle", "receive middle", "pick right", and "receive right". The dataset is unbalanced as it has 17 "place right" and "pick right", 23 "place middle" and "pick middle", 20 "place left" and "pick left", 24 "give right" and "receive right", 19 "give middle" and "receive middle" and 17 "give left" and "receive left" classes. Each frame in the video is captured at a rate of 30Hz. The actor was also wearing a motion capture system with eye gaze tracking glasses. For the purpose of this paper, only the information extracted from the RGB images were used.

B. Feature Selection and Embedding

The dataset is quite small and consists of ambiguous actions. Santos et al. (2019) proposed a method for feature selection which involved the use OpenPose (Cao et al., 2019) and OpenCV.

OpenPose is used to extract skeletal joints of the actor performing the actions. The 25 joint positions extracted from OpenPose are then divided into two vectors: limb and head key points. The head key points include x-y coordinates of nose, neck, right eye, left eye, right ear and left ear. The limb key points include x-y coordinates of right shoulder, right elbow, right wrist, left shoulder, left elbow and left wrist. The limb and head key points are both 12-dimensional vectors. This information is extracted for each frame in the video. The extracted features are then linearly transformed into a 16-dimensional vector as suggested by Santos et al. (2019).

OpenCV is used to extract x-y coordinates of the ball. Each frame in the video is subjected to a segmentation algorithm

which segments out the position of the ball. The approximate centroid of the ball is calculated using its area. The centre is an x-y coordinate which is then linearly transformed into a 16-dimensional vector as proposed by Santos et al. (2019).

The head key points and ball positions are considered as context vectors. They are concatenated together and then linearly transformed into a single 16-dimensional vector. The source embedding is a concatenation of the context vector and limb key points. This way limb movement and context get equal weightage in the prediction. The source embedding is used as input for the proposed model. The pre-processing is described in fig 1.

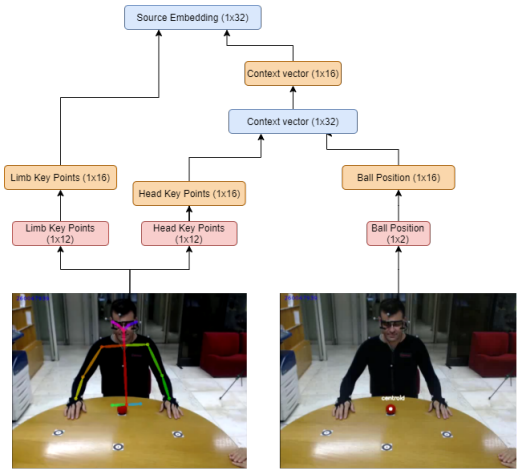


Fig. 1. The figure presents pre-processing technique used for the dataset. The extracted vectors are represented with red, the linearly transformed vectors are represented with orange and concatenated vectors are represented with blue colored rectangles respectively.

C. Proposed Model

The proposed model is based on a convolutional neural network. Fig 3 shows the overall architecture of the model.

A convolutional block (fig 2) is based on the works of Gehring et al. (2017). Each block has a convolutional layer with an output dimension equal to twice the input dimension. The output of the convolution is then passed to an activation layer called the gated linear units (GLU) (Dauphin et al., 2017). GLUs have gated mechanisms (similar to RNNs) in the activation and it essentially reduces the output dimension of GLU to half of its input dimension. The convolution is not followed by a pooling layer to preserve the same dimensions. To better ease, the learning of the network, the input of the convolution block is added back to the output of the GLU. This forms a residual connection (He et al., 2016) which helps during back-propagation. This creates the input to the next block.

As seen in the fig 3, the model is divided into two parts both of which comprise of stacked convolution blocks. As the architecture is inspired by Gehring et al. (2017), where a stack of convolution creates an encoder or a decoder, thus, this paper refers to the stack of convolutional blocks connected to the input as conv-E and the one connected directly to the output

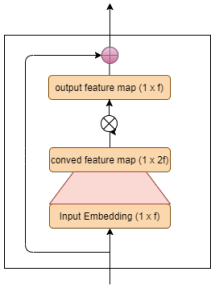


Fig. 2. The figure presents the convolutional block with feature size f and GLU activation. The input of the conv block is added to the output of the convolution to create a residual connection.

as conv-D. There can be "l" number of convolution blocks in conv-E and conv-D, with each convolution having a kernel size of "k". The output of the conv-E is given as input to the conv-D. The output of the last convolutional block in conv-D is connected to a linear layer with softmax activation which creates a probability distribution over all the classes, making it a multi-class classification problem.

The input to the model is the sequence of transformed source embedding $\mathbf{s} = (s_1, s_2, s_3, ..., s_m)$ of feature size f. Here m is the length of the sequence. The source embedding is linearly transformed from size = 32 to size = f using linear layer.

The source embedding is passed sequentially to the model. A feedback loop is present in the architecture which mimics the sequential learning of an RNN in Santos et al. (2019). Let the output of conv-E be represented by $\mathbf{o} = (o_1, o_2, o_3, ..., o_m)$. The output of the conv-E at time step *t-1* (given by $o(t-1)$) is added to input of conv-E at time step t. Similarly, let the output of conv-D be $\mathbf{q} = (q_1, q_2, q_3, ..., q_m)$. The output of conv-D at time step *t-1* (given by $q(t-1)$) is also added to the input of conv-D at time step t. Information about the position of the embedding are also added to the input of both the convolution stacks using positional embedding. It is given by $\mathbf{g} = (g_1, g_2, g_3, ..., g_m)$. Positional embedding is quite useful for CNNs when learning a sequential flow of information (Gehring et al., 2016, 2017). Therefore, the input to conv-E ($input_e$) and conv-D ($input_d$) stacks at time t is given by (1) and (2).

$$input_e(t) = s(t) + g(t) + o(t-1) \tag{1}$$

$$input_d(t) = o(t) + g(t) + q(t-1) \tag{2}$$

At the start of the sequence, the feedback loop is represented by zeros. Thus, the input to the model is one frame at a time, which goes through a series of convolution to provide a prediction at the end. The prediction at time *t-1* is taken into consideration for the prediction at time t. The model looks at the sequence in a series and provides predictions in a manner very similar to the LSTM model.

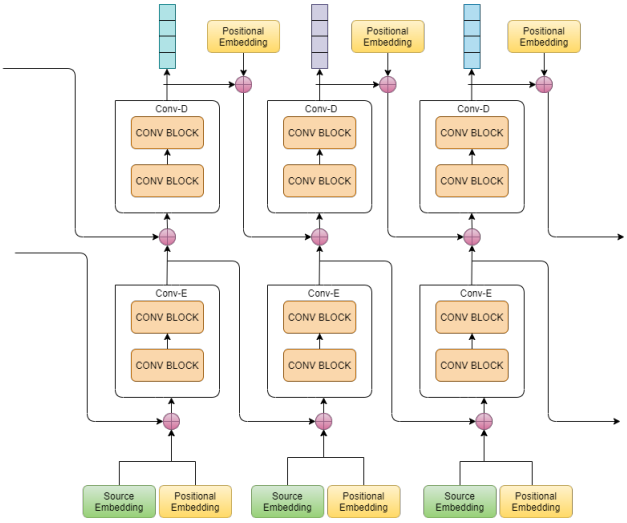


Fig. 3. The overall architecture of the proposed CNN model

TABLE I
TRAINING PARAMETERS

Hyper-parameters	CNN	LSTM
Feature size/Hidden dim	64	64
Epochs	60	60
Initial LR	0.001	0.001
LR factor	0.5	0.1
Patience (before LR reduction)	5	5
Optimizer	Adam	Adam
Gradient clip	0.1	0.1
Dropout	0	0.5

IV. EXPERIMENTAL SETUP

This section describes the experimental setup used to achieve the results. The experiments are performed using the dataset as described in III.

A. System Requirements

All the models were implemented using Python 3.6.9 along with Pytorch 1.6.0. The OpenPose demo (Cao et al., 2019) was used to extract pose information of the actor. OpenCV 4.1.2 was used for segmentation.

B. Training Parameters

The parameters used for the training of the models are given in table I. The batch size used for training is 1 as it allows training on full-length sequences without truncating a longer sequence or padding a shorter sequence. A smaller batch is also beneficial when working with small datasets such as Acticipate.

C. Experiments and Evaluation Metrics

A family of CNNs based models (as described in table II) were trained on a subset of the dataset for action recognition. For the paper, an LSTM model based on the architecture of Santos et al. (2019) was also implemented and trained on the features which were extracted using the method described in III. Due to possible noisy feature extraction and difference in training parameters, the LSTM model failed to replicate the exact results as shown in Santos et al. (2019). As the CNN models were also trained on the same data, this version of the LSTM model was considered as the base model.

The models are evaluated for action anticipation using the metrics, *accuracy at each observation ratio* as described by Santos et al. (2019). The formula is given below:

$$Accuracy(r) = \frac{1}{K}\sum_{i=1}^{K} pred(X^i_{[r*N]}, y^i, p) \tag{3}$$

The above metric takes into consideration the variable length of each sequence. Accuracy(r) defines the accuracy at observation ratio *r*. K is the total number of action sequences. N is the length of the particular sequence X^i. $X_{[r*N]}$ gives the partial sequence. The *pred* function returns 1 when the model predicts the right class, else it returns 0.

The above metric incorporates the uncertainty aspect of action anticipation as well. The action class is only assigned if the probability of the class is greater than some threshold p. The effect of varying threshold p on the anticipation accuracy is described in Santos et al. (2019) and is not covered in this paper. All the models are evaluated with p = 0.9.

V. RESULTS AND DISCUSSION

This sections gives a detailed report of the results acquired using the experimental setup described in III.

A. Action Recognition

A family of CNN models were implemented and their anticipation accuracy per observation ratio was calculated. An observation ratio of 1.0, makes it an action recognition task as the model sees the whole sequence before making a prediction. An anticipation model should have a good recognition accuracy (Santos et al., 2019). The action recognition of the CNN models is given in table II. It can be seen that out of the 12 models implemented, 6 models have 100% action recognition accuracy, while other 4 have about 99% accuracy. The LSTM version of the model has a recognition accuracy of 81.66%. Fig 4 shows the accuracy to observation ratio for the LSTM model.

The lowest recognition accuracy of the CNN models is about 96.25% which is still better than the LSTM model at 81.66%. However, the state-of-the-art implementation (Santos et al., 2019) had a recognition accuracy of 100% for the deterministic LSTM model which used full context (movement + head + ball position). The proposed CNN model has a comparable performance with state-of-the-art implementation. This confirms that the proposed architecture can be used for action recognition. The model can learn the sequential flow of information just like the LSTM anticipation model.

B. Effect of hyper-parameter

The CNN models are also evaluated for action anticipation accuracy to observation ratio as given in (3). The algorithm used for the calculation is described by Santos et al. (2019).

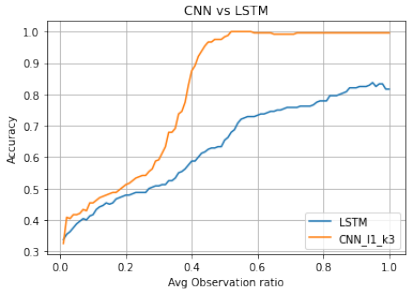


Fig. 4. The variation of anticipation accuracy wrt observation ratio for CNN and LSTM model

TABLE II
ACTION RECOGNITION ACCURACY FOR CNN MODELS

# conv blocks / kernel size	3	5	7	9
1	99.58	100	99.58	99.16
2	93.33	100	96.25	100
3	99.16	100	100	100

The graphs in fig 5 and fig 6 shows the variation in anticipation accuracy to observation ratio for the family of models.

From fig 5, it is clear that the anticipation accuracy at the starting of the sequence (observation ratio = 0.01) can be increased by increasing the layers of convolution blocks in conv-E and conv-D. The accuracy of CNN models with hyper-parameters l=1 has the lowest accuracy at the start of the sequence. It can also be seen from fig 5 (a), (b) and (c), that the models with $l = 3$ have higher anticipation accuracy at any given observation ratio. The task of anticipation requires high accuracy at lower observation ratio. Therefore, increasing the number of layers is a viable solution.

Additionally, it can be observed from fig 6, that increasing the kernel size doesn't affect the anticipation accuracy. Kernels in a 1D convolution are used to capture the relation between different time step. The proposed architecture works on a single time step, therefore the kernel size doesn't affect the performance of the model. This model learns the relationship between past time steps using a feedback loop. The inputs to the model were zero-padded on both sides. Maybe an extensive look at different padding techniques can help in improving model accuracy.

C. Similarities between CNN and LSTM model for action anticipation

There are many similarities between the LSTM and the CNN model for action anticipation. The CNN model learns the same details about the sequence as the LSTM model implemented by Santos et al. (2019). The results shown in figures 7, 8, 9 and 10 are produced using the CNN model with $l = 1$ and $k = 3$. As it can be observed, that even with the difference in architecture, the CNN model can learn sequence representation in the same way as described in Santos et al. (2019).

It can be seen from the figs 7 and 8, that the model's confidence about its prediction increases with every passing frame. In fig 7 (b), the high threshold value of $p = 0.9$ helps in avoiding a false classification. Therefore, having a high threshold helps in avoiding overconfident predictions as suggested by Santos et al. (2019). However, the proposed model makes a mistake in fig. 10 (c) at the starting of the sequence even when a high threshold is provided.

In the figs, 9 and 10, shows the probability distribution over the action classes to the frame. It is seen that actions like *pick* and *receive* can be anticipated with high accuracy at the starting of the sequence. These actions are heavily dependent on contextual information (ball position). This shows that the proposed model can understand contextual information which is given as input. It proves that the implemented CNN model works similar to the state-of-the-art LSTM model (Santos et al., 2019).

D. CNN model vs LSTM model

Fig 4 shows the comparison between the CNN model and LSTM model. The CNN model used for comparison has $l = 1$ and $k = 3$. This is the simplest model of the CNN family. The accuracy at observation ratio of 0.1 is almost same for both the models with the CNN model at 32.5% accuracy and the LSTM model at 33.75% accuracy.

Overtime (with increasing observation ratio), the CNN model outperforms the LSTM model. This shows that CNN can learn a better representation of sequential data over time when compared to the base model. CNN has an accuracy of approx 100% (98.75%) at an observation ratio of 0.5. Meanwhile, LSTM has an accuracy of 66.25%.

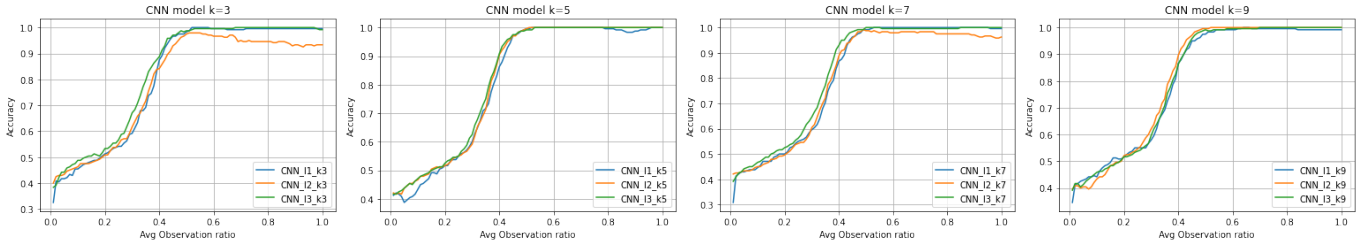
The CNN model achieves an accuracy of 92.08% with an observation ratio of 0.42. The LSTM model has an accuracy of 60% at the same observation ratio. Both the models use a threshold p of 0.9. The frames in the Acticipate dataset were captured at 30Hz. The average length of the action sequences in the dataset is about 80 frames. This suggests that the CNN model can accurately anticipate 92.08% of actions in the dataset in the first 33 frames (1 sec).

This is a significant improvement on the base LSTM model. This shows that the CNN model can outperform the base LSTM model even with noisy pre-possessing results.

VI. CONCLUSION AND FUTURE WORK

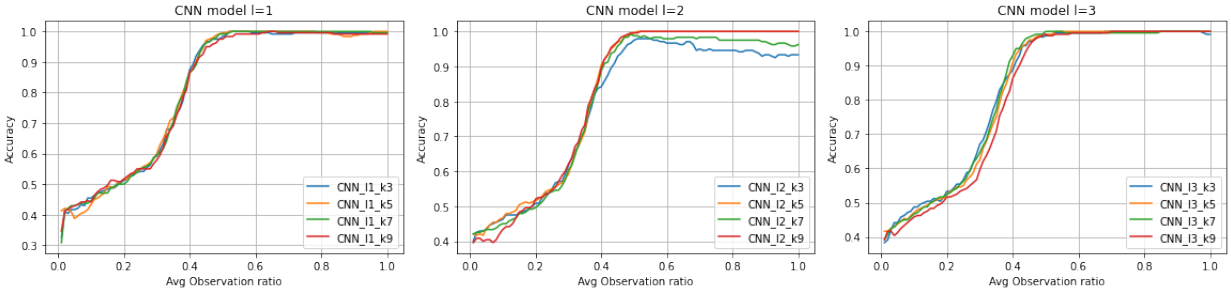
Action Anticipation is an important field of research due to its vast applications. It is based on processing sequential data. Therefore, recurrent neural networks are predominantly used for it. This paper suggested a new architecture only using convolutional neural networks. The architecture seems to outperform the base LSTM model which was trained on a similar setup.

The convolution models have various hyper-parameter. A few of the hyper-parameters were explored. It was shown that an increase in the number of layers of convolution can increase the anticipation accuracy at lower observation ratio. The CNN model works similarly to the LSTM model. It can accurately predict some tasks based on a single frame. The probability



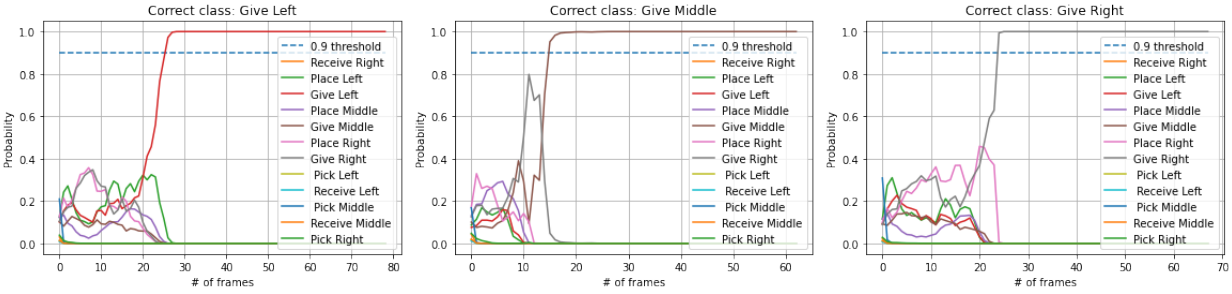
(a) (b) (c) (d)

Fig. 5. The variation of anticipation accuracy wrt observation ratio based on different numbers of convolution blocks in conv-E and conv-D



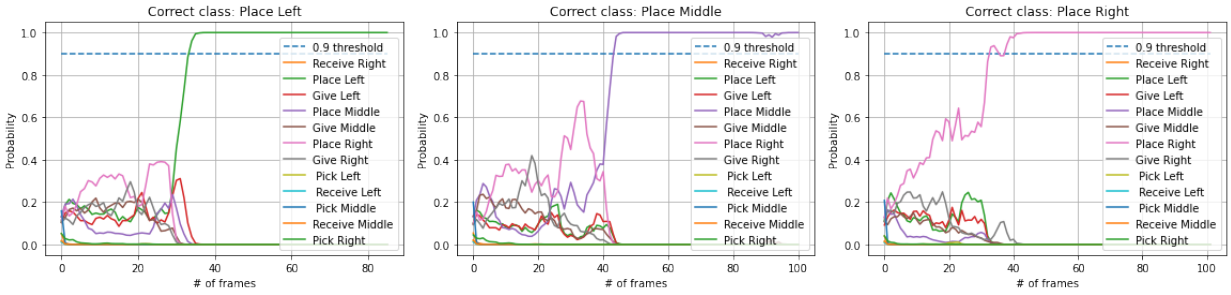
(a) (b) (c)

Fig. 6. The variation of anticipation accuracy wrt observation ratio based on different kernel sizes



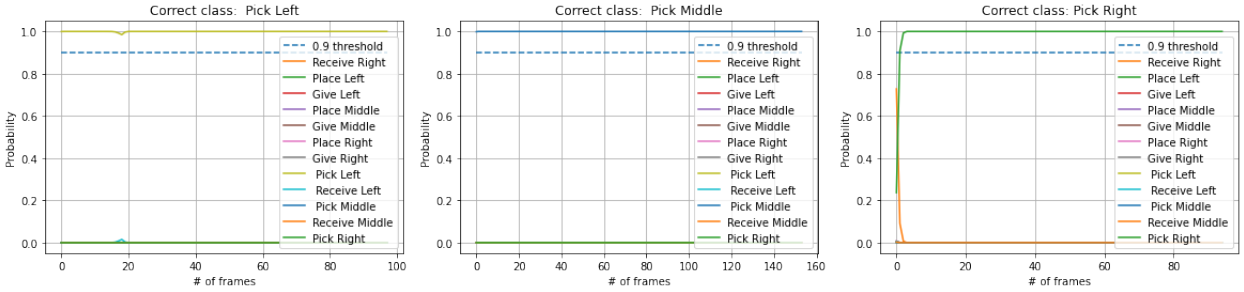
(a) (b) (c)

Fig. 7. probability distribution per frame for the action *give*



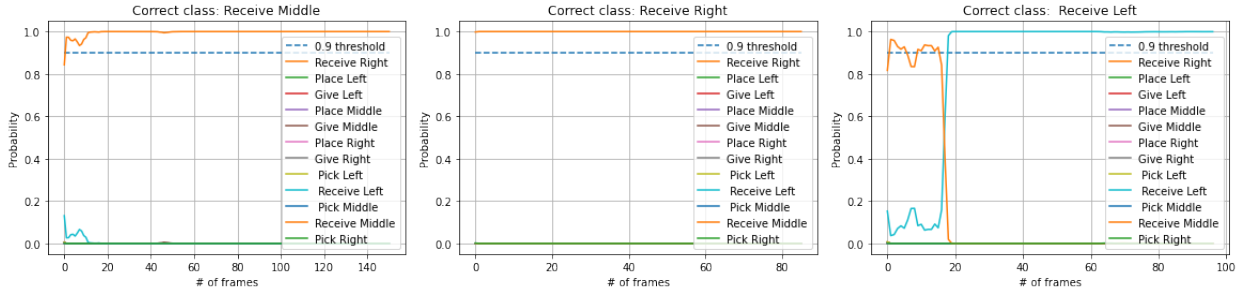
(a) (b) (c)

Fig. 8. Probability distribution per frame for the action *place*



(a) (b) (c)

Fig. 9. Probability distribution per frame for the action *pick*



(a) (b) (c)

Fig. 10. Probability distribution per frame for the action *receive*

score for the correct action class increases at it sees more of the given sequence.

The suggested CNN model has great potential for improvement. It is a good area of research given that CNNs have a lot of computational advantages as compared to LSTMs. In the future, an extensive list of hyper-parameters can be explored such as the size of the feature maps, independent number of convolution blocks in conv-E and conv-D, etc. The CNN model should also be evaluated on other metrics presented by Santos et al. (2019). It would be interesting to implement a Bayesian version of the deterministic model. A Bayesian model is better at handling uncertainty and has out-performed deterministic models at action anticipation.

References

A. Bauer, D. Wollherr, and M. Buss. Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.

J. Bütepage, H. Kjellström, and D. Kragic. Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration. *arXiv preprint arXiv:1702.08212*, 2017.

Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

L. Chen, J. Lu, Z. Song, and J. Zhou. Part-activated deep reinforcement learning for action prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

S. Cho and H. Foroosh. A temporal sequence learning for action recognition and prediction. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 352–361, 2018.

Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941, 2017.

N. F. Duarte, M. Raković, J. Tasevski, M. I. Coco, A. Billard, and J. Santos-Victor. Action anticipation: Reading the intentions of humans and robots. *IEEE Robotics and Automation Letters*, 3(4):4132–4139, 2018.

A. Furnari and G. Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

A. Furnari and G. M. Farinella. Egocentric action anticipation by disentangling encoding and inference. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 3357–3361. IEEE, 2019.

H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Predicting the future: A jointly learnt model for action anticipation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019a.

H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Predicting the future: A jointly learnt model for action anticipation. In *Proceedings of the IEEE International*

Conference on Computer Vision, pages 5562–5571, 2019b.

K. Ganapathy, S. S. Abdul, and A. A. Nursetyo. Artificial intelligence in neurosciences: A clinician's perspective. *Neurology India*, 66(4):934, 2018.

J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*, 2016.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

G. Hoffman. Anticipation in human-robot interaction. In *2010 AAAI Spring Symposium Series*. Citeseer, 2010.

B. Jang, M. Kim, G. Harerimana, S.-u. Kang, and J. W. Kim. Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism. *Applied Sciences*, 10(17):5841, 2020.

S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

Q. Ke, M. Fritz, and B. Schiele. Time-conditioned action anticipation in one shot. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

I. Lee, D. Kim, S. Kang, and S. Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1012–1020, 2017.

J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1647–1656, 2017.

J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 299–306, 2013.

G. Pang, X. Wang, J. Hu, Q. Zhang, and W.-S. Zheng. Dbdnet: Learning bi-directional dynamics for early action prediction. In *IJCAI*, pages 897–903, 2019.

A. Pannu. Artificial intelligence and its application in different areas. *Artificial Intelligence*, 4(10):79–84, 2015.

C. Rodriguez, B. Fernando, and H. Li. Action anticipation by predicting future dynamic images. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

M. Sadegh Aliakbarian, F. Sadat Saleh, M. Salzmann, B. Fernando, L. Petersson, and L. Andersson. Encouraging lstms to anticipate actions very early. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

C. C. d. Santos, P. Moreno, J. L. A. Samatelo, R. F. Vassallo, and J. Santos-Victor. Action anticipation for collaborative environments: The impact of contextual information and uncertainty-based prediction. *arXiv preprint arXiv:1910.00714*, 2019.

P. Schydlo, M. Rakovic, L. Jamone, and J. Santos-Victor. Anticipation in human-robot cooperation: A recurrent neural network approach for multiple action sequences prediction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, 2018.

P. Semberecki and H. Maciejewski. Deep learning methods for subject text classification of articles. In *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 357–360, 2017.

A. Sethia and P. Raut. Application of lstm, gru and ica for stock price prediction. In *Information and Communication Technology for Intelligent Systems*, pages 479–487. Springer, 2019.

Y. Shi, B. Fernando, and R. Hartley. Action anticipation with rbf kernelized feature mapping rnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 301–317, 2018.

W. Tong, A. Hussain, W. X. Bo, and S. Maharjan. Artificial intelligence for vehicle-to-everything: A survey. *IEEE Access*, 7:10823–10843, 2019.

A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2017.

X. Wang, J.-F. Hu, J.-H. Lai, J. Zhang, and W.-S. Zheng. Progressive teacher-student learning for early action prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Y. Wang, M. Huang, X. Zhu, and L. Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

W. Yin, K. Kann, M. Yu, and H. Schütze. Comparative study of CNN and RNN for natural language processing. *CoRR*, abs/1702.01923, 2017. URL `http://arxiv.org/abs/1702.01923`.

L. Zhang, W. Meng, A. Chen, M. Mei, and Y. Liu. Application of lstm neural network for urban road diseases trend forecasting. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4176–4181. IEEE, 2018.