Action Anticipation with RBF Kernelized Feature Mapping RNN

Yuge Shi[0000−0003−1905−9320], Basura Fernando[0000−0002−6920−9916], and Richard Hartley[0000−0002−5005−0191]

The Australian National University, Australia

Abstract. We introduce a novel Recurrent Neural Network-based algorithm for future video feature generation and action anticipation called feature mapping RNN . Our novel RNN architecture builds upon three effective principles of machine learning, namely parameter sharing, Radial Basis Function kernels and adversarial training. Using only some of the earliest frames of a video, the feature mapping RNN is able to generate future features with a fraction of the parameters needed in traditional RNN. By feeding these future features into a simple multilayer perceptron facilitated with an RBF kernel layer, we are able to accurately predict the action in the video.

In our experiments, we obtain 18% improvement on *JHMDB-21* dataset, 6% on *UCF101-24* and 13% improvement on *UT-Interaction* datasets over prior state-of-the-art for action anticipation.

Keywords: Human action prediction, novel Recurrent Neural Network, Radial Basis Function kernel, Adversarial training

1 Introduction

Action anticipation (sometimes referred to as action prediction) is gaining a lot of attention due to its many real world applications such as human-computer interaction [2,33,30], sports analysis [3,4,56] and pedestrian movement prediction [9,21,18,5,46] especially in the autonomous driving scenarios.

In contrast to most widely studied human action recognition methods, in action anticipation, we aim to recognize human action as early as possible [39,23,28,42,49]. This is a challenging task due to the complex nature of video data. Although a video containing a human action consists of a large number of frames, many of them are not representative of the action being performed; large amount of visual data also tend to contain entangled information about variations in camera position, background, relative movements and occlusions. This results in cluttered temporal information and makes recognition of the human action a lot harder. The issue becomes even more significant for action anticipation methods, as the algorithm has to make a decision using only a fraction of the video at the very start. Therefore, finding a good video representation that extracts temporal information relevant to human action is crucial for the anticipation model.

To over come some of these issues, we resort to use deep convolutional neural networks (CNNs) and take the deep feature on the penultimate layer of CNN as video
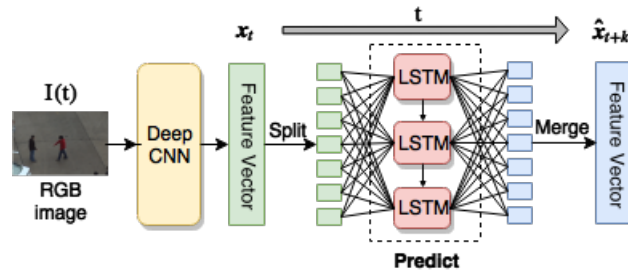
Fig. 1: **Overview of Proposed feature mapping RNN :** Given a frame extracted from video data, the algorithm first passes the RGB image $I(t)$ through a deep CNN to acquire high level features of the image x_t. The vector is then split into smaller segments x_t^i of equal length. Each scalar element in the segmented vector is used as input to a single LSTM cell that produces the prediction of corresponding feature element in frame $(t + k)$, where $k \geq 1$. After all segments are processed with LSTMs, all the prediction segments $\hat{x}_{t+k}^i$ are concatenated back together to form $\hat{x}_{t+k}$, which contains high level features of $I(t + k)$.

representation. Another motivation to use deep CNNs stems from the difficulty of generating visual appearances for future. Therefore, similar to Vondrick *et al.* [49], we propose a method to generate future features tailored for action anticipation task: given an observed sequence of deep CNN features, a novel Recurrent Neural Network (RNN) model is used to generate the most plausible future features and thereby predicting the action depicted in video data. An overview of this model can be found in Fig. 1.

The objective of our RNN is to map the feature vector at time t denoted by x_t to the future feature vector at $(t + k)$ denoted by x_{t+k}. Because only a fraction of the frames are observed during inference, the future feature generator should be highly regularized to avoid over-fitting. Furthermore, feature generator needs to model complex dynamics of future frame features.

This can be resolved by parameter sharing. Parameter sharing is a strong machine learning concept that is being used by many modern leaning methods. Typically, CNNs share parameters in the spatial domain and RNNs in the temporal dimension. In our work, we propose to utilize parameter sharing in an unconventional way for RNN models by expanding it to the feature domain. This is based on the intuition that the CNN feature activations are correlated to each other.

By utilizing parameter sharing across feature activations, our proposed RNN is able to learn the temporal mapping from x_t to x_{t+k} with significantly fewer parameters. This greatly boosts the computational efficiency of the prediction model and correspondingly shortens the response time. We call our novel RNN architecture feature mapping RNN .

To model complex dynamic nature of video data, we make use of a novel mapping layer inside our RNN. In principle, the hidden state of the RNN captures the temporal information of observed sequence data. In our method, hidden state of the RNN is processed by a linear combination of Gaussian Radial Basis Function (RBF) kernels to

produce the future feature vector. While a linear model defines a simple hyperplane as mapping functions, the kernelized mapping with RBF kernels can model complex surfaces and therefore has the potential of improving the prediction accuracy. In our work, we also implement RBF kernels on the action classification multi-layer perceptron to improve the performance of classifiers.

Ideally, we are interested in learning the probability distribution of future given the past features. To learn this conditional distribution, inspired by the of Generative Adversarial Networks [12], an adversarial approach is used to evaluate the cost of the feature mapping RNN. The RNN is trained with an adversarial loss and re-constrictive L2 loss. In this way, the model is optimized not only with the intention of reducing the Euclidean distance between the prediction and ground truth, but also taking probability distribution of the feature vector into consideration.

In a summary, our contributions are:

– We propose a novel RNN architecture that share parameters across temporal domain as well as feature space.
– We propose a novel RBF kernel to improve the prediction performance of RNNs.
– We demonstrate the effectiveness of our method for action anticipation task beating state-of-the-art on standard benchmarks.

2   Related Work

The model proposed in this paper focuses on future video content generation for action prediction and action anticipation [23,50,36,35,55,27,20,42,39,29,49,10]. In contrast to the widely studied action recognition problem, the action anticipation literature focuses on developing novel loss functions to reduce the predictive generalization error [39,29,16] or to improve the generalization capacity of future content such as future appearance [10] and future features [49]. The method propose in this paper also focuses on future content generation and therefore could further benefit from novel loss functions as proposed in [39,29,16].

In the early days, Yu *et al.* [55] make use of spatial temporal action matching to tackle early action prediction. Their method relies on spatial-temporal implicit shape models. By explicitly considering all history of observed features, temporal evolution of human actions is used to predict the class label as early as possible by Kong *et al.* [20]. Li *et al.* 's work [27] exploits sequence mining, where a series of actions and object co-occurrences are encoded as symbolic sequences. Soomro *et al.* [43] propose to use binary SVMs to localize and classify video snippets into sub-action categories, and obtain the final class label in an online manner using dynamic programming. In [50], action prediction is approached using still images with action-scene correlations. Different from the above mentioned methods, our work is focused on action anticipation from videos. We rely on deep CNNs along with a RNN that shares parameters across both feature and time dimensions to generate future features. To model complex dynamics of video data, we are the first to make use of effective RBF kernel functions inside RNNs for the action anticipation task.

On the other hand, feature generation has been studied with the aim of learning video representation, instead of specifically for action anticipation. Inspired by natural

language processing technique [1], authors in [34] propose to predict the missing frame or extrapolate future frames from an input video sequence. However, they demonstrate this only for unsupervised video feature leaning. Other popular models include the unsupervised encoder-decoder scheme introduced by [45] for action classification, probabilistic distribution generation model by [25] as well as scene prediction learning using object location and attribute information introduced by [8]. Research in recent years on applications of Generative Adversarial Network on video generation have given rise to models such as MoCoGAN [48], TGAN [40] and Walker *et al.* 's work [53] on video generation using pose as a conditional information. The mechanisms of these GAN variations are all capable of exploiting both the spatial and temporal information in videos, and therefore have showed promising results in video generation.

Moreover, trajectory prediction [22], optical-flow prediction [52], path prediction [51,54] and motion planning [11,19], sports forecasting [7], activity forecasting of [31] are also related to our work. All these methods generate future aspects of the data. Our novel RNN model, however, focuses on generating future features for action anticipation.

3 Approach

3.1 Overview

Similar to methods adopted by other action anticipation algorithms, our algorithm makes predictions of action by only observing a fraction of video frames at the beginning of a long video. The overall pipeline of our method is shown in Fig. 1. First, we extract some CNN feature vectors from frames and predict the future features based on the past features. Subsequently, a multilayer perceptron (MLP) is used to classify generated features. We aggregate predictions from observed and generated features to recognize the action as early as possible.

3.2 Motivation

Denote observed sequence of feature vectors up to time t by $X = \langle \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \cdots \boldsymbol{x_t} \rangle$ and future feature vector we aim to produce by $\hat{\boldsymbol{x}}_{\boldsymbol{t+k}}$, where $k \geq 1$ and $\boldsymbol{x_t} \in \mathbb{R}^d$. We are interested in modeling the conditional probability distribution of $P(\boldsymbol{x_{t+k}}|\ \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \cdots \boldsymbol{x_t}; \Theta)$, where Θ denotes the parameters of the probabilistic model.

It is natural to use RNNs or RNN variants such as Long Short Term Memory (LSTM) [14] to model the temporal evolution of the data. However, learning such a mapping could lead to over-fitting since these methods tend not to utilise the temporal coherence and the evolutionary nature of video data [32].

Furthermore, a naive CNN feature mapping using a LSTM from past to the future is also prone to over-fitting. A LSTM with hidden state of dimensionality H and takes feature vectors of dimensionality d as input uses parameters in the order of $4(dH + d^2)$. As an example, if we use the penultimate activations of Inception V3 [47] as feature vectors ($d = 2048$), a typical LSTM ($H = 512$) would require parameters in the order of 10^7. We believe that the effectiveness of such models can be largely improved by utilising the correlation of high level activations of modern CNN architectures [47,13].

Motivated by these arguments, we propose to train a LSTM model where parameters are not only shared in the time domain, but also across feature activations. By doing so, we aim to self-regularize the feature generation of the algorithm. We name our novel architecture *feature mapping RNN* . Furthermore, to increase the functional capacity of RNNs, we make use of Radial Basis Functions (RBF) to model temporal dynamics of the conditional probability distribution $P(\boldsymbol{x_{t+k}} \mid \boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \cdots \boldsymbol{x_t}; \Theta)$. These mechanisms will be introduced in details in the following subsection.

3.3   Feature Mapping RNN with RBF Kernel Mapping

A traditional feature generation RNN architecture takes a sequence of vectors up to time t as input and predicts the future feature vector $\hat{\boldsymbol{x}}_{t+k}$. Typically, the following recurrent formula is used to model the prediction:

$$\boldsymbol{h_t} = f(\boldsymbol{x_t}, \boldsymbol{h_{t-1}}; \theta) \tag{1}$$

Where $\boldsymbol{h_t}$ is the hidden state ($\boldsymbol{h_t} \in \mathbb{R}^H$) which captures the temporal information of the sequence and θ are the parameters of the recurrent formula. Then we utilize this hidden state to predict the future feature vector $\boldsymbol{x_{t+k}}$ using the following formula:

$$\hat{\boldsymbol{x}}_{t+k} = \boldsymbol{h_t} \times \boldsymbol{W} \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{H \times D}$ is the parameter that does the linear mapping to predict the future feature vector.

As introduced previously, in our feature mapping RNN the parameters Θ are shared across several groups of feature activations. This is achieved by segmenting the input feature vector of dimensionality d into equal size sub-vectors of dimensionality D, where D is referred to as *feature step size*.

Now let us denote the ith sub-feature vector of size D by $\boldsymbol{x_t^i}$. Intuitively, if we concatenate all such sub-feature vectors in an end-to-end manner, we will be able to reconstruct the original feature vector $\boldsymbol{x_t}$. The time sequence of data for the ith sub-feature vector is now denoted by $X^i = \langle \boldsymbol{x_1^i}, \boldsymbol{x_2^i}, \boldsymbol{x_3^i}, \cdots \boldsymbol{x_t^i} \rangle$. If we process each sequence X^i in units of $\boldsymbol{x_t^i}$ with the RNN model in equation 1 and equation 2, we will be able to predict $\boldsymbol{x_{t+k}^i}$ and by concatenating them end-to-end, generate $\boldsymbol{x_{t+k}}$. This approach reduces the number of parameters used in the RNN model from $4(dH + d^2)$ to $4(DH + D^2)$, which results in a considerable boost in computational efficiency especially when $D \ll d$. However, the parameter complexity of the model would remain polynomial and is relevant to multiple hyperparameters.

To further improve the efficiency of our model, we adopt an even bolder approach: we propose to convert the sequence of vectors of $X^i = \langle \boldsymbol{x_1^i}, \boldsymbol{x_2^i}, \boldsymbol{x_3^i}, \cdots \boldsymbol{x_t^i} \rangle$ to a sequence of scalars. Let us denote the *j-th* dimension of sub-vector $\boldsymbol{x_t^i}$ by $x_t^{i(j)}$. Now instead processing sequence of vectors X^i, we convert the sequence X^i to a new sequence of scalars $X^{'i} = \langle x_1^{i(1)}, x_1^{i(2)}, \cdots x_1^{i(D)}, x_2^{i(1)}, x_2^{i(2)}, \cdots, x_t^{i(k)}, \cdots x_t^{i(D)} \rangle$. Length of the sequence of scalars $X^{'i}$ is equal to $t \times D$ and we generate $\frac{d}{D}$ number of such sequences from each original sequence of feature vector X.

We then propose to process sequence of scalars using a RNN (LSTM) model. The computation complexity is now linear, with number of parameters used in the recurrent model (LSTM) reduced to $4(H + 1)$ and depends only on the hidden state size.

Again, given the current sequence of vectors X, we want to generate future feature vector $\boldsymbol{x_{t+k}}$. In the our RNN model, this is translated to predicting sequence of scalars $\left\langle x_{t+k}^{i(1)}, \cdots x_{t+k}^{i(D)} \right\rangle$ from sequence $X^{'i}$ for all sub-feature vectors $i = 1$ to $\frac{d}{D}$. Then we merge all predicted scalars for time $t + k$ to obtain $\boldsymbol{x_{t+k}}$.

Therefore, mathematically our new RNN model that share the parameter over feature activations can be denoted by the following formula:

$$\boldsymbol{h_t^{i(l)}} = f(x_t^{i(l)}, \boldsymbol{h^{i(l)}}_{t-1}; \Theta^{'}) \tag{3}$$

where $\Theta^{'}$ is the new parameter set of the RNN (LSTM) and the future *l-th* scalar of *i-th* sub-feature vector is given by:

$$\hat{x}_{t+k}^{i(l)} = \boldsymbol{h_t^{i(l)}} \cdot \boldsymbol{w}^{'}. \tag{4}$$

To further improve the functional capacity of our feature mapping RNN , we make use of Radial Basis Functions (RBF). Instead of using a simple linear projection of the hidden state to the future feature vector, we propose to exploit the more capable Radial Basis Functional mapping. We call this novel RNN architecture the RBF kernelized feature mapping RNN , denoted by the following formula:

$$\hat{x}_{t+k}^{i(l)} = \sum_{j=1}^{n} \alpha_j^l \exp\left[- \frac{(\boldsymbol{h_t^{i(l)}} - \boldsymbol{\mu_j^l})^2}{\sigma_j^{l\,2}} \right] \tag{5}$$

where $\boldsymbol{\mu_j^l}$, σ_j^l and α_j^l are parameters learned during training and n the number of RBF kernels used. These parameters are shared across all sub-feature vectors. The future feature vector $\hat{\boldsymbol{x}}_{t+k}^i$ is calculated as the linear combination of RBF kernels outputs. Since the RBF kernels are better at modeling complex planes in the feature space, this functional mapping is able to accurately capture more complicated dynamics. Implementing the kernalised RBF on our feature mapping RNN enables the model to so with fewer parameters than classical RNNs.

Note that the method we have presented here only uses non-overlapping feature-sub-vectors, *i. e.* no overlapping exists between 2 consecutive sub-vectors. However, overlapping feature-sub-vectors can be used to improve the robustness of feature generation. Therefore, instead of using a non-overlapping *feature stride* of D, we use an overlapping stride of size S. In this case, we take the average between all overlapping parts of 2 consecutive sub-vectors to obtain $\hat{x}_{t+k}^{i(l)}$.

3.4   Training of feature mapping RNN

Data generation, especially visual data generation with raw images, has remained a challenging problem for years mainly due to the absence of suitable loss function. The most commonly used function for this task is the $\mathcal{L}_2$ loss. However, it works under the

assumption that data is drawn from a Gaussian distribution, which makes the loss function ineffective when dealing with data that follows other distributions. As an example, if there exists only two equally possible value v_1 and v_2 for a pixel, the possibility for $v_{avg} = (v_1 + v_2)/2$ to be the true value for that pixel is minimal. However, v_{avg} will be assigned to the output in a neural network that uses $\mathcal{L}_2$ loss to evaluate the cost. This property of the $\mathcal{L}_2$ loss function causes a "blurry" effect on the generated output. Similar observations can be seen for feature vector generation.

Recent developments in Generative Adversarial Networks address this issue successfully [12]. Traditional GAN consists of 2 CNNs, one of them is named *generator* (denote as $\mathcal{G}$) and the other *discriminator* (denote as $\mathcal{D}$). The GAN effectively learns the probabilistic distribution of the original data, and therefore eliminates the "blockiness" effect caused by $\mathcal{L}_2$ loss function. Here, we propose to train the feature mapping RNN algorithm using a combination of $\mathcal{L}_2$ and adversarial loss, which is realized by implementing the feature mapping RNN as the *generator* denoted by $\mathcal{G} : x_t^i \rightarrow \hat{x}_{t+k}^i$. By doing so, we are able to produce prediction that is both accurate and realistic.

$\mathcal{L}_2$ *loss:* The $\mathcal{L}_2$ loss is defined as the mean squared error between the generated feature and the real feature vector of the future frame given as follows:

$$\mathcal{L}_2^{\mathcal{G}}(x_t) = ||x_{t+k}^i - \hat{x}_{t+k}^i|| = ||x_{t+k}^i - \mathcal{G}(x_t^i)||. \tag{6}$$

Adversarial loss: We use generator adversarial loss proposed by [12] where we train $\mathcal{G}$ so that $\mathcal{D}$ believes $\mathcal{G}(x_t^i)$ comes from the dataset, at which point $\mathcal{D}(\mathcal{G}(x_t^i)) = 1$. The loss function is defined as:

$$\mathcal{L}_{adv}^{\mathcal{G}} = -\log(\mathcal{D}(\mathcal{G}(x_t^i))). \tag{7}$$

By adding this loss to our objective function, the RNN is encouraged to generate feature prediction with probabilistic distribution similar to the original data. Finally, the loss function of our RNN generator $\mathcal{G}$ is given by:

$$\mathcal{L}^{\mathcal{G}} = \lambda_1 \mathcal{L}_2^{\mathcal{G}} + \lambda_2 \mathcal{L}_{adv}^{\mathcal{G}}. \tag{8}$$

The discriminator is trained to judge whether its inputs are real or synthetic. The objective is to output 1 when given input is the real data x_{t+k}^i and 0 when input is generated data $\mathcal{G}(x_t^i)$. Therefore, the discriminator loss is defined as:

$$\mathcal{L}^{D} = -log(\mathcal{D}(x_{t+k}^i)) - log(1 - \mathcal{D}(\mathcal{G}(x_t^i))). \tag{9}$$

3.5   Action classifier and inference

To evaluate the authentication of predicted features generated by the feature matching RNN, we again use the frame features to train a 2-layer MLP appended with a RBF kernel layer (equation 5) to classify videos as early as possible. Illustration of our RBF kernelized MLP is shown in Fig 2. The classification loss is evaluated using a cross-entropy loss. Feature mapping RNN and the action classification MLP is trained separately. One might consider training both MLP and the feature mapping RNN jointly. However, in terms of performance, we did not see that much of advantage.
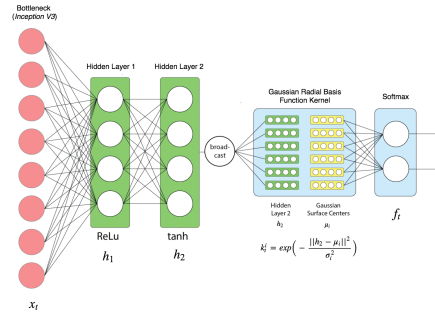
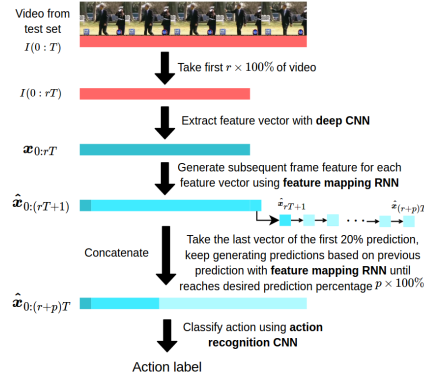Fig. 2: Illustration of RBF keneralized multilayer perceptron.

Fig. 3: Testing Procedure of Feature Mapping RNN

During inference, we take advantage of all observed and generated features to increase the robustness of the results. Accuracy is calculated by performing temporal average pooling on all predictions (see Fig 3).

4   Experiments

4.1   Datasets

Three datasets are used to evaluate the performance of our model, namely *UT-Interaction* [37], *JHMDB-21* [17] and *UCF101-24* [44]. We follow the standard protocols for each of the datasets in our experiments. We select these datasets because they are the most related to action anticipation task that has been used in prior work [39,35].

UT-Interaction The *UT-Interaction* dataset (UTI) is a popular human action recognition dataset with complicated dynamics. The dataset consists of 6 types of human interactions executed under different backgrounds, zoom rates and interference. It has a total of 20 video sequences split into 2 sets. Each video is of approximately 1 minute long, depicting 8 interactions on average. The available action classes include handshaking, pointing, hugging, pushing, kicking and punching. The performance evaluation methodology requires the recognition accuracy to be measured using a 10-fold leave-one-out cross validation per set. The accuracy is evaluated for 20 times while changing the test sequence repeatedly and final result is yielded by taking the average of all measurements.

JHMDB-21 *JHMDB-21* is another challenging dataset that contains 928 video clips of 21 types of human actions. Quite different from the *UT-interaction* where video clips of different actions are scripted and shot in relatively noise-free environments, all videos in *JHMDB-21* are collected from either movies or online sources, which makes the dataset a lot more realistic. Each video contains an execution of an action and the dataset is split into 3 sets for training, validation and testing.

UCF101-24 *UCF101-24* is a subset of *UCF101*. The dataset consists of more than 3000 videos from 24 action classes of *UCF101*. Since all the videos are collected from YouTube, the diversity of data in terms of action types, backgrounds, camera motions, lighting conditions etc are guaranteed. In addition, each video depicts up to 12 actions of the same category with different temporal and spatial features, which makes it one of the most challenging dataset to date.

4.2   Implementation Details

Feature Mapping RNN The Feature Mapping RNN is trained with batch size of 128, using a hidden size (H) of 4 in all experiments unless otherwise specified. The default dimensionality of feature sub vector referred to as *feature step size(D)* is set to 128. We make use of six RBF kernels within the RBF kernelized feature mapping RNN . Feature stride is set to 64 and weight of the adversarial loss (λ_1) is set to 1 and the weight for $\mathcal{L}_2$ loss is set to 10 (*i. e.* λ_2).
Action classifier MLP The a simple two layer MLP classifier consists of two hidden layers with 256 and 128 activation respectively. We also use RBF kernels along with the MLP where number of kernels set to 256. MLP is trained with batch size of 256.
Training and Testing Procedures We use pre-trained Inception V3 [47] penultimate activation as the frame feature representation. The dimensions of each feature vector is 2048 ($d = 2048$). The action classification MLP is trained on the feature vectors from the training split of the datasets. These features are also used to train our feature mapping RNN to generate future features. Both models are trained with learning rate 0.001 and exponential decay rate 0.9.
Protocols Following the experimental protocol [39,35], we used only the first $r\%$ (50% for *UT-Interaction* and 20% for *JHMDB-21*) of the video frames to predict action class for each video. To utilise our model, we generate extra $p\%$ (referred to as prediction percentage) of the video features using our RBF kernalized feature mapping RNN . Therefore, we make use of $(r+p)\%$ feature vectors of the original video length to make the final prediction. To generate the next future feature at test time, we recursively apply our feature mapping RNN given all previous features (including the generated ones). We then use our action classification MLP to predict the action label using max pooling or simply average the predictions. This procedure is demonstrated more intuitively in Fig.3.

4.3   Comparison to State-of-the-Art

We compare our model to the state-of-the-art algorithms for action anticipation task on the *JHMDB-21* dataset. Results are shown in Table 1. Our best algorithm (denoted as *fm+RBF+GAN+Inception V3* in the table) outperforms the state-of-the-art by 18%, and we can clearly see that the implementation of kernel SVM and adversarial training improves the accuracy by around 3 to 4%. In addition, to show the progression of how our method is able to outperform the baseline by such a large margin, we also implemented the Feature Mapping RNN on top of VGG16 so that the deep CNN pre-processing is consistent with other methods in Table 1. The *fm+VGG16* entry in the table shows an

Table 1: Comparison of our model against state-of-the-arts on *JHMDB-21* dataset for action anticipation. We follow the protocol of *JHMDB-21* for action anticipation and predictions are made from using only 20% of video sequence.

	Method	Accuracy
	ELSTM [39]	55%
	Within-class Loss [28]	33%
Others	DP-SVM [42]	5%
	S-SVM [42]	5%
	Where/What [43]	10%
	Context-fusion [16]	28%
	fm+VGG16	**63%**
Ours	fm+kSVM+GAN+VGG16	**67%**
	fm+Inception V3	**70%**
	fm+RBF+GAN+Inception V3	**73%**

Table 2: Comparison of our model against state-of-the-arts on *UT-Interaction* dataset for action anticipation. Following protocol of *UT-Interaction*, predictions are made from using only 50% of video sequence.

Method	Accuracy
ELSTM [39]	84%
Within-class Loss [28]	48%
Context-fusion [16]	45%
Cuboid Bayes [35]	25%
I-BoW [35]	65%
D-BoW [35]	70%
Cuboid SVM [38]	32%
BP-SVM [26]	65%
Ours	**97%**

8% improvement from baseline *ELSTM*, which is purely influenced by the implementation of Feature Mapping RNN .

Experiments are also carried out on the two other mentioned datasets, where our best method outperforms the state-of-the-art by 13% on *UT-Interaction* and 6% on *UCF101-24*, as shown in Table 2 and Table 3 respectively.

We believe these significant improvements suggests the effectiveness of two main principles, the parameter sharing and expressive capacity of RBF functionals. To further investigate the impact of each component, we perform a series of experiments in the following sections.

Table 3: Comparison of our model against state-of-the-arts on *UCF101-24* dataset for action anticipation. Again, predictions are made from using only 50% of video sequence.

Method	Accuracy
Temporal Fusion [6]	86%
ROAD [41]	90%
ROAD + BroxFlow [41]	92%
Ours	**98%**

4.4   Analysis

In this section we compare the influence of different components of our RBF kernelized feature mapping RNN . As shown in Table 4, we compare following variants of our RNN model, including:

  (a) Feature Mapping RNN : use only $\mathcal{L}_2$ loss to train the Feature Mapping RNN ;

(b) Feature Mapping RNN +RBF: our RNN with kernalised RBF, still only using $\mathcal{L}_2$ loss;

(c) Feature Mapping RNN + RBF + GAN: RBF kernelized feature mapping RNN with adversarial loss.

Apart from the Feature Mapping RNN -based models, we also conduct experiments on the following method as comparisons to our model:

(d) Linear: a matrix of size $D \times D$ is used for feature generation (D is dimension of input feature);

(e) Vanilla LSTM: generate future action features with traditional vanilla LSTM. $\mathcal{L}_2$ loss is used to train it;

(f) Vanilla LSTM + RBF: vanilla LSTM with kernalised RBF, using only $\mathcal{L}_2$ loss;

(g) Vanilla LSTM + RBF + GAN: RBF kernalized vanilla LSTM with added adversarial loss.

Note that all the results are obtained using features extracted by Inception V3 network, and the accuracy are acquired using max pooling at prediction percentage $p = 50\%$.

Table 4: Comparison of different approach on *JHMDB-21* dataset

Method	Accuracy
Linear	62.7%
Vanilla LSTM	66.3%
Vanilla LSTM + RBF	67.9%
Vanilla LSTM + RBF + GAN	-
Feature Mapping RNN	72.2%
Feature Mapping RNN + RBF	72.8%
Feature Mapping RNN + RBF + GAN	73.4%

The results in Table 4 shows the proposed scheme outperforms the linear model significantly while using fewer parameters. Most interestingly, the feature mapping RNN outperforms vanilla LSTM by almost 6% indicating the impact of parameter sharing in the feature space. We can also conclude from Table 4 that the application of adversarial loss as well as RBF kernel layers encourages the model to generate more realistic future features, which is reflected by the improvement in accuracy with Feature Mapping RNN +RBF and Feature Mapping RNN +RBF+GAN. It is also shown in the Table 4 that vanilla LSTM trained with RBF kernel yields almost 2% higher accuracy than plain vanilla LSTM, which proves further that the RBF layer is something the baseline can benefit from. Regrettably, the vanilla LSTM with adversarial training model failed to stabilise due to large number of parameters needed in the LSTM cells to reconstruct the original feature distribution.

The influence of RBF kernalized feature mapping RNN is quite distinctive. If we compare the red curve to the green one, we can see that the discrepency between them becomes larger as the prediction percentage increases. This indicates that the RBF kernalized feature mapping RNN generate more accurate future features in the long term, and hence it is a more robust model than plain feature mapping RNN . Comparing the red and green curve to the orange and blue one, we can also conclude that the adversarial
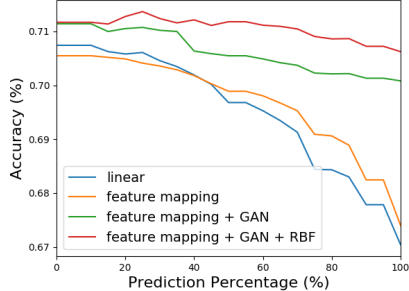
Fig. 4: Prediction accuracy without pooling for JHMDB-21 dataset at different video prediction percentages p. RBF kernalized Feature mapping RNN is trained using adversarial loss is able to achieve the highest stable accuracy.
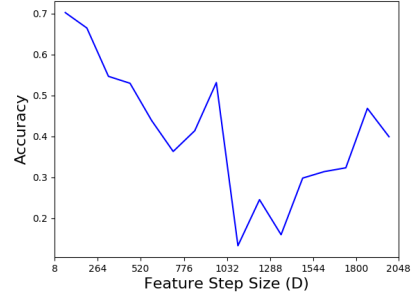

Fig. 5: Prediction accuracy evaluated at different feature step sizes on *JHMDB-21* dataset. The accuracy plotted in the image is found by implementing feature step size between $D = 8$ to 2048 with increment of 8 on the model and the rolling average is taken among every 16 measurements. No temporal pooling is used.

loss assist the RNN training in a similar way. Even without the assistance of GAN loss and RBF kernel, the feature mapping RNN still performs better than liner projection RNN.

4.5   Influence of Hyper-parameters

Feature Step Size The accuracy of the generated data indicates the existence of strong correlations between the D-dimensional segments of the feature vectors. By default, we resort to feature step size of 128 ($D = 128$). In order to further explore this property, we experimented with different feature step sizes. In Fig.5, we plot the recognition accuracy against feature step size. We observe that small feature step size guarantees effective feature generation. Specifically, the prediction remains above 70% when feature step size is smaller than 200. This phenomena can be explained by the intuition that when feature step size is large, the model tries to generalize a large set of features with mixed information at one time step, which results in degraded performance.

It is also interesting to note that the prediction accuracy oscillates drastically as the feature step size exceeds 250. This indicates that perhaps the feature vector summarizes information of the original image in fixed-size clusters, and when we attempt to break these clusters by setting different feature step size, the information within each time step lacks continuity and consistency, which subsequently compromises the prediction performance.

Although smaller feature step size builds a more robust model, the training time with feature step size 16 takes only half the amount of time of training with step size 4, with no compromise on prediction accuracy. Therefore, it might be beneficial sometimes to choose a larger feature step size to save computational time.

Table 5: Prediction accuracy at different feature stride size (S)

Interval Size	Accuracy
$S = 4$	74.3%
$S = 8$	73.8%
$S = 16$	74.3%
$S = 32$	73.2%
$S = 64$	73.2%
$S = 128$	72.4%

Table 6: Prediction accuracy using LSTM cells with different state size (H).

Hidden State Size	Accuracy
$H = 2$	71.7%
$H = 4$	73.2%
$H = 8$	72.7%
$H = 16$	73.2%
$H = 32$	73.2%
$H = 64$	73.8%

Table 7: Prediction accuracy using different number of RBF kernels.

No. of Kernels	Accuracy
$k = 4$	72.7%
$k = 8$	72.7%
$k = 16$	73.3%
$k = 32$	73.3%
$k = 64$	72.7%
$k = 128$	73.8%
$k = 256$	72.2%

Interval Size In this section we experiment the effect of overlapping sub-feature vectors on our RBF kernalized feature mapping RNN . Recall that the feature mapping RNN is denoted by $\mathcal{G} : \boldsymbol{x}_t^i \rightarrow \hat{\boldsymbol{x}}_{t+k}^{i:}$. Instead of incriminating i by the multiple of feature step size D, in an attempt to improve the prediction accuracy, we define an feature stride S that is smaller than D. The prediction accuracy of Feature Mapping RNN with several different feature stride value is shown in Table 5.

LSTM state size This section aims at investigating the influence of LSTM cell's hidden state size (H) on the model's performance. Since the hidden state stores essential information of all the input sequence data, it is common to consider it as the "memory" of the RNN. It is intuitive to expect an improvement in performance when we increase the size of the hidden state up to some extent.

However, the results in Table 6 shows that increasing the LSTM state size does not have much effect on the prediction accuracy, especially when the state size becomes larger than 8. This is because in the proposed feature mapping RNN model, each LSTM cell takes only one scalar as input, as opposed to the traditional RNN cells that process entire vectors. As the hidden state size is always greater than the input size (equal to 1), it is not surprising that very large H does not have much influence on the model performance.

Number of RBF Kernels In this section we study the influence of number of Gaussian surfaces used in feature mapping RNN . We calculate prediction accuracy while increasing the number of Gaussian kernels from 2^1 to 2^8. Results are as shown in Table 7. The results show a general trend of increasing prediction performance as we add more number of kernels, with the highest accuracy achieved at when $k = 128$. However, result obtained when $k = 256$ is worse than when $k = 4$. This phenomena could be explained by over-fitting, resulted from RBF kernel's strong capability of modeling temporal dynamics of data with complex boundaries.

Conclusions for hyper-parameters tuning The conclusion from these experiments is that the model is not too sensitive to the variation of these hyper-parameters in general, which demonstrates its robustness. Results further demonstrated the computational efficiency of our approach. Since it is possible to effectively train the model with very few parameters, it can be stored on mobile devices for fast future action anticipation.

5   Conclusions

The proposed RNN which uses a very few parameters outperforms state-of-the-art algorithms on action anticipation task. Our extensive experiments indicates the model's ability to produce accurate prediction of future features only observing a fraction of the features. Furthermore, our RNN model is fast and consumes fraction of the memory which makes it suitable for real-time execution on mobile devices. Proposed feature mapping RNN can be trained with and without lables to generate future features. Our feature generator does not use class level annotations of video data. Therefore, in principle, we can increase the robustness of the model utilizing large amount of available unlabelled data. The fact that the model is able to generate valid results using very few parameters provides strong proofs for the existence of inner-correlation between deep features, which is a characteristic that can have implications on many related problems such as video tracking, image translation, and metric learning.

In addition, by appending a RBF layer to the RNN, we observe significant improvement in prediction accuracy. However, it was also noted that over-fitting occurs when the model is implemented with too many kernel RBFs. To fully explore functional capacity of RBF function, in future studies, we aim to implement kernel RBFs on fully connected layer of popular deep CNN models such as *ResNet* [13], *AlexNet* [24] and *DenseNet* [15].

In conclusion, proposed RBF kernalized feature mapping RNN demonstrates the power of parameter sharing and RBF functions in a challenging sequence learning task of video action anticipation.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. Journal of machine learning research **3**(Feb), 1137–1155 (2003) 4

2. Dix, A.: Human-computer interaction. In: Encyclopedia of database systems, pp. 1327–1331. Springer (2009) 1

3. Duan, L.Y., Xu, M., Chua, T.S., Tian, Q., Xu, C.S.: A mid-level representation framework for semantic sports video analysis. In: 2003 ACM International Conference on Multimedia. pp. 33–44. ACM (2003) 1

4. Ekin, A., Tekalp, A.M., Mehrotra, R.: Automatic soccer video analysis and summarization. IEEE Transactions on Image processing **12**(7), 796–807 (2003) 1

5. Enzweiler, M., Gavrila, D.M.: Integrated pedestrian classification and orientation estimation. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition. pp. 982–989 (2010) 1

6. Fan, Z., Lin, T., Zhao, X., Jiang, W., Xu, T., Yang, M.: An online approach for gesture recognition toward real-world applications. In: Zhao, Y., Kong, X., Taubman, D. (eds.) Image and Graphics. pp. 262–272. Springer International Publishing, Cham (2017) 10

7. Felsen, P., Agrawal, P., Malik, J.: What will happen next? forecasting player moves in sports videos. In: 2017 IEEE International Conference on Computer Vision (2017) 4

8. Fouhey, D.F., Zitnick, C.L.: Predicting object dynamics in scenes. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2027–2034 (2014). https://doi.org/10.1109/CVPR.2014.260 4

9. Gandhi, T., Trivedi, M.M.: Image based estimation of pedestrian orientation for improving path prediction. In: 2008 IEEE Intelligent Vehicles Symposium. pp. 506–511 (2008) 1

10. Gao, J., Yang, Z., Nevatia, R.: Red: Reinforced encoder-decoder networks for action anticipation. arXiv preprint arXiv:1707.04818 (2017) 3

11. Gong, H., Sim, J., Likhachev, M., Shi, J.: Multi-hypothesis motion planning for visual object tracking. In: 2011 IEEE International Conference on Computer Vision. pp. 619–626 (Nov 2011). https://doi.org/10.1109/ICCV.2011.6126296 4

12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014) 3, 7

13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 4, 14

14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997) 4

15. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (2017) 14

16. Jain, A., Singh, A., Koppula, H.S., Soh, S., Saxena, A.: Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In: 2016 IEEE International Conference on Robotics and Automation. pp. 3118–3125 (2016) 3, 10

17. Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J.: Towards understanding action recognition. In: 2013 IEEE International Conference on Computer Vision. pp. 3192–3199 (Dec 2013) 8

18. Keller, C.G., Gavrila, D.M.: Will the pedestrian cross? a study on pedestrian path prediction. 2014 IEEE Transactions on Intelligent Transportation Systems **15**(2), 494–506 (2014) 1

19. Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M.: Activity Forecasting, pp. 201–214. Springer Berlin Heidelberg, Berlin, Heidelberg (2012) 4

20. Kong, Y., Kit, D., Fu, Y.: A discriminative model with multiple temporal scales for action prediction, pp. 596–611 (2014) 3

21. Kooij, J.F.P., Schneider, N., Flohr, F., Gavrila, D.M.: Context-based pedestrian path prediction. In: 2014 European Conference on Computer Vision. pp. 618–633. Springer (2014) 1

22. Kooij, J.F.P., Schneider, N., Flohr, F., Gavrila, D.M.: Context-Based Pedestrian Path Prediction, pp. 618–633. Springer International Publishing, Cham (2014) 4

23. Koppula, H.S., Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(1), 14–29 (Jan 2016). https://doi.org/10.1109/TPAMI.2015.2430335 1, 3

24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012) 14

25. Lampert, C.H.: Predicting the future behavior of a time-varying probability distribution. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition. pp. 942–950 (2015). https://doi.org/10.1109/CVPR.2015.7298696 4

26. Laviers, K., Sukthankar, G., Aha, D.W., Molineaux, M., Darken, C., et al.: Improving offensive performance through opponent modeling. In: 2009 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2009) 10

27. Li, K., Fu, Y.: Prediction of human activity by discovering temporal sequence patterns. 2014 IEEE transactions on pattern analysis and machine intelligence **36**(8), 1644–1657 (2014) 3

28. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in lstms for activity detection and early detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1942–1950 (2016). https://doi.org/10.1109/CVPR.2016.214 1, 10

29. Ma, S., Sigal, L., Sclaroff, S.: Learning activity progression in lstms for activity detection and early detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1942–1950 (2016) 3

30. MacKenzie, I.S.: Fitts' law as a research and design tool in human-computer interaction. Human-computer interaction **7**(1), 91–139 (1992) 1

31. Mahmud, T., Hasan, M., Roy-Chowdhury, A.K.: Joint prediction of activity labels and starting times in untrimmed videos. In: 2017 IEEE International Conference on Computer Vision. pp. 5784–5793 (2017) 4

32. Mobahi, H., Collobert, R., Weston, J.: Deep learning from temporal coherence in video. In: 2009 International Conference on Machine Learning. pp. 737–744. ACM (2009) 4

33. Newell, A., Card, S.K.: The prospects for psychological science in human-computer interaction. Human-computer interaction **1**(3), 209–242 (1985) 1

34. Ranzato, M., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., Chopra, S.: Video (language) modeling: a baseline for generative models of natural videos. CoRR **abs/1412.6604** (2014) 4

35. Ryoo, M.S.: Human activity prediction: Early recognition of ongoing activities from streaming videos. In: 2011 International Conference on Computer Vision. pp. 1036–1043 (Nov 2011). https://doi.org/10.1109/ICCV.2011.6126349 3, 8, 9, 10

36. Ryoo, M.S., Aggarwal, J.K.: Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In: 2009 IEEE International Conference on Computer Vision. pp. 1593–1600 (2009). https://doi.org/10.1109/ICCV.2009.5459361 3

37. Ryoo, M.S., Aggarwal, J.K.: UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA). http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html (2010) 8

38. Ryoo, M., Chen, C.C., Aggarwal, J., Roy-Chowdhury, A.: An overview of contest on semantic description of human activities (sdha) 2010. In: Recognizing Patterns in Signals, Speech, Images and Videos, pp. 270–285. Springer (2010) 10

39. Sadegh Aliakbarian, M., Sadat Saleh, F., Salzmann, M., Fernando, B., Petersson, L., Andersson, L.: Encouraging lstms to anticipate actions very early. In: 2017 IEEE International Conference on Computer Vision (Oct 2017) 1, 3, 8, 9, 10

40. Saito, M., Matsumoto, E., Saito, S.: Temporal generative adversarial nets with singular value clipping. In: 2017 IEEE International Conference on Computer Vision. vol. 2, p. 5 (2017) 4

41. Singh, G., Saha, S., Sapienza, M., Torr, P., Cuzzolin, F.: Online real time multiple spatiotemporal action localisation and prediction (2017) 10

42. Soomro, K., Idrees, H., Shah, M.: Online localization and prediction of actions and interactions. CoRR **abs/1612.01194** (2016) 1, 3, 10

43. Soomro, K., Idrees, H., Shah, M.: Predicting the where and what of actors and actions through online action localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2648–2657 (2016) 3, 10

44. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR **abs/1212.0402** (2012) 8

45. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: 2015 International Conference on Machine Learning. pp. 843–852 (2015) 4

46. Suard, F., Rakotomamonjy, A., Bensrhair, A., Broggi, A.: Pedestrian detection using infrared images and histograms of oriented gradients. In: 2006 IEEE Intelligent Vehicles Symposium. pp. 206–212. IEEE (2006) 1

47. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016). https://doi.org/10.1109/CVPR.2016.308 4, 9

48. Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. arXiv preprint arXiv:1707.04993 (2017) 4
49. Vondrick, C., Pirsiavash, H., Torralba, A.: Anticipating visual representations from unlabeled video. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. pp. 98–106 (2016) 1, 2, 3
50. Vu, T.H., Olsson, C., Laptev, I., Oliva, A., Sivic, J.: Predicting Actions from Static Scenes, pp. 421–436. Springer International Publishing, Cham (2014) 3
51. Walker, J., Gupta, A., Hebert, M.: Patch to the future: Unsupervised visual prediction. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3302–3309 (2014). https://doi.org/10.1109/CVPR.2014.416 4
52. Walker, J., Gupta, A., Hebert, M.: Dense optical flow prediction from a static image. In: 2015 IEEE International Conference on Computer Vision. pp. 2443–2451 (Dec 2015). https://doi.org/10.1109/ICCV.2015.281 4
53. Walker, J., Marino, K., Gupta, A., Hebert, M.: The pose knows: Video forecasting by generating pose futures. In: 2017 IEEE International Conference on Computer Vision. pp. 3352–3361 (2017) 4
54. Xie, D., Todorovic, S., Zhu, S.C.: Inferring dark matter and dark energy from videos. In: 2013 IEEE International Conference on Computer Vision. pp. 2224–2231 (Dec 2013). https://doi.org/10.1109/ICCV.2013.277 4
55. Yu, G., Yuan, J., Liu, Z.: Predicting human activities using spatio-temporal structure of interest points. In: 2012 ACM International Conference on Multimedia (2012) 3
56. Zhong, D., Chang, S.F.: Structure analysis of sports video using domain models. In: 2001 IEEE International Conference on Multimedia & Expo. Citeseer (2001) 1