Spatiotemporal Feature Residual Propagation for Action Prediction

He Zhao
York University, Toronto
zhufl@eecs.yorku.ca

Richard P. Wildes
York University, Toronto
wildes@cse.yorku.ca

Abstract

Recognizing actions from limited preliminary video observations has seen considerable recent progress. Typically, however, such progress has been had without explicitly modeling fine-grained motion evolution as a potentially valuable information source. In this study, we address this task by investigating how action patterns evolve over time in a spatial feature space. There are three key components to our system. First, we work with intermediate-layer ConvNet features, which allow for abstraction from raw data, while retaining spatial layout, which is sacrificed in approaches that rely on vectorized global representations. Second, instead of propagating features per se, we propagate their residuals across time, which allows for a compact representation that reduces redundancy while retaining essential information about evolution over time. Third, we employ a Kalman filter to combat error build-up and unify across prediction start times. Extensive experimental results on the JHMDB21, UCF101 and BIT datasets show that our approach leads to a new state-of-the-art in action prediction.

1. Introduction

The human visual system is capable of both recognizing complete actions as well as anticipating future outcomes based on preliminary observations. For example, high performance athletes in multi-player sports respond to the actions of their opponents before they complete their executions [47]. In contrast, computational modeling of vision-based action prediction has received far less attention than action recognition. Action prediction shares many challenges with action recognition, *e.g.* the need to deal with viewpoint and execution variations as well as the fact that the evolution of actions across a video tends to be entangled with distracting information, *e.g.* clutter, camera motion, occlusion and motion blur. Additional challenges present themselves for the case of prediction, *e.g.* certain action categories share similar sub-components at different stages (*e.g.* pushing and patting both start with stretching of arms), which makes distinctions especially difficult when only partial information is available. More generally, incomplete executions resulting from lack of extended temporal context can lead to data that is not discriminitive enough for ready classification.

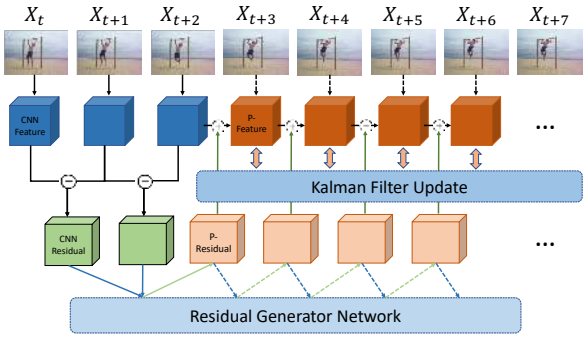


Figure 1: Overview of Proposed Feature Residual Propagation Approach to Action Prediction. Intermediate layer ConvNet features are extracted from an initial set of input frames; in the depicted example, these are given as $[X_t, X_{t+1}, X_{t+2}]$; subsequent frames (*e.g.* $[X_{t+3}, \ldots, X_{t+7}]$ are not seen by the system during testing (although they are during training) and are shown here merely for context. Inital feature residuals, CNN Residuals, are extracted via pointwise differencing of temporally adjacent feature maps. A generative model, Residual Generator Network (RGN), then recursively estimates future residuals, P-Residuals. Predicted features, P-Features, are recovered via addition of residuals to the initial reference feature map. A Kalman Filter serves to minimize error accumulation across time. The Kalman Filter operates across an entire video sequence during training, but only across the initially observed partial sequence during testing. Final action classification (not shown in figure) is performed with reference to both the initially observed and predicted features.

Traditionally, the problem of action prediction is formulated by transferring between full video information and partial observations. These approaches often overlook the rich motion patterns contained in videos, which has been demonstrated to play a pivotal role in action recognition [33, 3, 25, 21]. With the recent success of deep networks

on action recognition (*e.g*. [48, 9, 10, 4, 44]) deep representation learning based approaches offer more possibilities. For example, one can design a temporally adaptive objective function that encourages the model to produce the correct label as early as possible [35, 19]. Alternatively, one can adopt a recurrent neural network to infer recursively the next features conditioned on previous observations [37, 46]. However, the fact that such approaches depend on the activation of fully-connected layers may compromise performance, as the vectorized feature format collapses local information and contains much more noise [43].

In response to the aforementioned challeges, we focus on exploring the subtle changes among spatial features across time and propose a feature Residual Generator Network (RGN) to propagate into the future. We choose intermediate level activations of a pretrained deep network for propagation (*e.g*. final ConvLayer output, *c.f*. [51]), because features at such layers capture rich spatial structures [50]. Rather than propagate the features per se, we propagate feature residuals as they lead to a compact representation that still captures essentials of how features change over time. To ameliorate error accumulation over time, we incorporate a Kalman filter mechanism. Empirical evaluation shows that our approach yields new state-of-the-art performance on three popular action prediction datasets. Figure 1 provides a pictorial overview. Code is available at https://github.com/JoeHEZHAO/Spatiotemporal-Residual-Propagation.

2. Related work

Early work on video-based action prediction concentrated on use of handcrafted features to build temporally regularized discriminative models [34]. Other work along these lines developed a dynamic bag of words to infer a global action label [33]. An alternative approach solved a posterior maximization on sparse feature encodings [3]. Still other work focused on enforcing consistency between features computed across multiple temporal scales [21].

More recent work has focused on deep learning. Some such work has based prediction on action tubes over deep detectors [39, 38]; these approaches rely on success with the unsolved problem of early action detection and are correspondingly limited. In other work, a ConvNet with an LSTM was used to define a temporally adaptive objective function to assign labels as early as possible [35]. An alternative approach learned mappings between semantic features of full and partial videos [22, 23], which was extended with an LSTM to handle hard samples for improved performance [19]. By concentrating on relatively high-level semantic features, these approaches tend to overlook more temporally local information. To compensate for this potential shortcoming, yet other work has generated sequential features based on current observations [37, 46]. A limitation of these approaches is their reliance on a network's penultimate layers, which because of vectorization sacrifices potentially useful spatial structure.

Recent work in action recognition has shown benefits of explicitly exploiting intermediate layer features. As examples: Intermediate features have been used for local frame aggregation [26], building compact feature correlations via bilinear operations [7], spatial warping for real-time recognition [51] and recovering images from various deep abstraction stages [8]. The positive results these approaches have yielded may be explained by the fact that in comparison to fully-connected layers, intermediate layers preserve more spatial structure and thereby support finer distinctions (*e.g*. in motion layout) as well as have fewer parameters and thereby combat overfitting. For these reasons, we build on intermediate layer features in our work on action prediction.

Residual information can play an important role in processing of redundant data even while capturing important subtle differences in an efficient fashion. MPEG-4 compression is a well established outstanding example of such processing [12], as is more general coarse-to-fine motion estimation (*e.g*. [1]). Recent work that exploits residual processing has considered optical-flow estimation [31], image denoising [17], video artifact removal [27] and action recognition [49]. Our appoach to action prediction provides a novel use of residual processing.

Arguably, the most similar works to ours are Deep Feature Flow [51] and CoViAR [49]. The former applies estimated optical flow to propagate deep spatial features for real-time video recognition. However, optical flow estimates beyond observed frames are not available, which makes application to prediction ill-defined. The latter explores the residual property of raw video data and develops a deep network for action recognition in the compressed domain; however, it has not addressed temporal extrapolation. Indeed, it seems that no previous approach has based action prediction on recursive generation of residuals across intermediate level features and shown that such an approach leads to state-of-the-art performance.

3. Technical approach

3.1. Overview

We seek to predict the correct action label, y, given the initial portion of a partially observed video, $X_{1:k}$, where k represents the kth frame of a video that in total has K frames. The key ingredient in support of our goal is an effective approach for propagating the information contained in initially observed consecutive frames $X_{1:k}$ to unobserved $X_{k+1:K}$. The video action label, y, is then recovered via classification of the entire concatenated sequence $X_{1:K} = Cat\{X_{1:k}, X_{k+1:K}\}$. Follow existing methods, we define the term **observation ratio**, g, as the fraction

of the observed frame set, $X_{1:k}$, to the full set, $X_{1:K}$. We present results from experiments with $g \in [0.1, 1.0]$.

Rather than predict future frames per se, we instead predict intermediate layer features of a ConvNet trained for action recognition. We are particularly interested in intermediate layer features, because features at such layers enjoy a level of abstraction from the raw data that focuses on action relevant components, even while preserving spatial layout to capture relations between action components as well as scene context.

We decouple the prediction process into two steps: feature residual propagation and feature reconstruction. As discussed in Sec. 2, feature residual information previously has been used as a convenient proxy for full data observations as it retains salient changes to objects and motions, even while reducing redundancy entailed in explicit representation of non-changing portions of observed data. Here, we advance the application of residual extraction and processing in the domain of ConvNet features to yield a novel framework for action prediction.

For illustrative purposes, we use the TSN architecture for initial feature extraction and final classification, because of its generally strong performance on action recognition [48]. While we use the TSN features and classifier, our prediction framework does not rely on the specifics of that approach and therefore should be more widely applicable to action prediction.

3.2. Feature residuals

Given a partially observed video with a set of frames $X_{1:k}$, let (mid-level) features extracted at time t be denoted as $\mathbf{d}_t \in \Re^{C \times W \times H}$, with C the number of feature channels, W the feature map width and H the feature map height. Temporal feature residuals at time t are then calculated via pointwise differencing along each channel

$$\mathbf{r}_t\big|_c = \mathbf{d}_t\big|_c - \mathbf{d}_{t-1}\big|_c, \quad 2 \leq t \leq k, 1 \leq c \leq C \qquad (1)$$

where $\big|_c$ indicates application to channel c, *i.e.* the value at spatial position (w, h) in channel c at time $t-1$ is subtracted from the value at time t and assigned to the residual, $\mathbf{r}_t \in \Re^{C \times W \times H}$, at the same spatial position and channel. Owing to the differencing operation, the cardinality of the set of calculated residuals, $\{\mathbf{r}_{2:k}\}$, is one less than the set of features, $\{\mathbf{d}_{1:k}\}$.

From the limited feature set $\{\mathbf{d}_{1:k}\}$ and their residuals set $\{\mathbf{r}_{2:k}\}$, we seek to recover the feature representation of $\{\mathbf{d}_{k+1:K}\}$. To achieve this result, we proceed in two steps. First, we recursively generate feature residuals $\{\mathbf{r}_{k+1:K}\}$ via appeal to a feature Residual Generator Network (RGN). Second, we sequentially add the residuals to the features that have been observed or generated so far to reconstruct features into the future according to

$$\mathbf{d}_{t+1} = \mathbf{d}_t + \mathbf{r}_{t+1}, \quad k \leq t \leq K-1. \qquad (2)$$

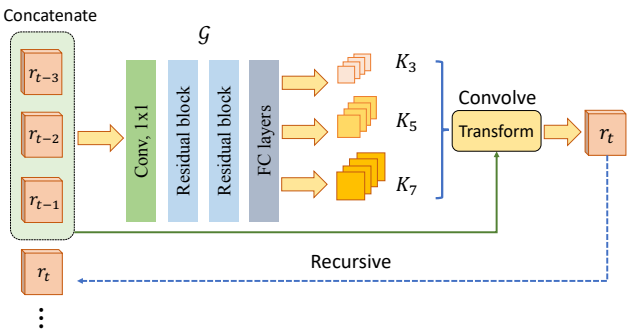


Figure 2: Temporal Extrapolation Residual Generator Network (RGN) for Predicting Next Time Step Residual. Our model recursively generates motion kernels, K_n, using a ConvNet, $\mathcal{G}$, based on a short historical temporal window, $m = 3$, and performs transformations in a convolutional fashion on the most recent residual. The newly generated residual joins the ongoing prediction sequence until the end of the desired sequence. Subscripts n specify kernel size (*i.e.* $n \times n$). Convolutions on the residuals are performed on a channel-by-channel basis; so, multiple kernels are depicted for each n.

In Fig. 1, P-Residuals and P-Features are used to distinguish predicted residuals and features, resp. In the next subsection, we define our feature residual generator.

3.3. Residual Generator Network (RGN)

Our Residual Generator Network (RGN) is schematized in Fig. 2. At its core is a kernel motion transformation, $\mathcal{G}$. Given a set of stacked temporal observations, $\mathcal{G}$ produces a set of kernels, $\{K_n\}$, that can be convolved with the most recent residual input to predict the next (unobserved) result. We choose the kernel motion transformation because it has proven useful in synthesis of future intensity frames [11, 32], can be applied with various kernel sizes, $n \times n$, to capture multiple motion scales and has lower complexity than its deep regression counterpart [46].

We generate motion kernels for each channel, c, with multiple sizes, $n \times n$, according to

$$K_n = \mathcal{G}(\mathbf{r}_t, \mathbf{r}_{t-1}, \ldots, \mathbf{r}_{t-m} \,|\, \mathbf{r}_{t-m-1}, \ldots, \mathbf{r}_2; \theta_f)\big|_c, \qquad (3)$$

where $\mathcal{G}$ is a ConvNet with learnable parameters, θ_f, that inputs residuals over its current observation window, m, but through its recurrent application depends on the entire history of residuals and thereby follows the Markov-Chain conditional distribution. The architecture of $\mathcal{G}$ is depicted in Fig. 2, with implementation details provided in Sec. 4.2. Subsequent to kernel generation, for each channel, c, we apply the kernels to the current residual $\mathbf{r}_t$ and average the

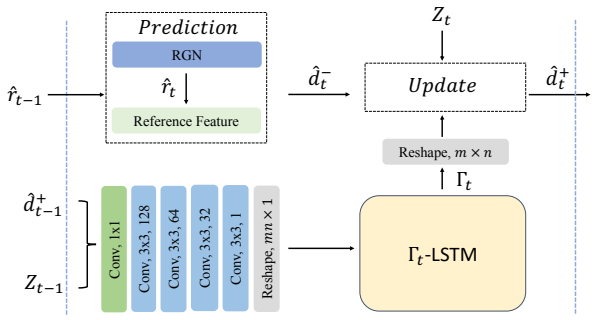


Figure 3: Depiction of Kalman Update Procedure. Prior estimation of feature $\hat{\mathbf{d}}_t^-$ is updated with Kalman gain Γ_t. The transition of Γ_t is modeled by a ConvNet with LSTM (Γ_t-LSTM) across time. At each time step, Γ_t corrects $\hat{\mathbf{d}}_t^-$ with observed measurement Z_t and produces posterior $\hat{\mathbf{d}}_t^+$ for next time step inference.

results to predict the next time step residual

$$\mathbf{r}_{t+1}\big|_c = \frac{1}{N}\sum_{n=1}^{N} K_n * \mathbf{r}_t\big|_c, \tag{4}$$

where $*$ stands for convolution. Based on preliminary experiments we use $N = 3$, with $n \in \{3, 5, 7\}$.

3.4. Kalman filter correction

Recent approaches to sequential feature generation prefer decomposing multi-step prediction into single-step prediction for training and apply the same model recursively for testing. Owing to error accumulation, such approaches often lead to quality degeneration as the sequence becomes longer. Current time-series optimization methods (*e.g.* BackPropagation Through Time (BPTT)) lack the ability to inject mid-stage supervision during optimization; thus, errors in initial stages negatively impact the following results. To avoid such scenarios, we incorporate a Kalman filter [18] into our approach, *c.f.* [6, 27]; see Fig. 3.

The Kalman filter recursively estimates an internal state from a time series of measurements via alternating *Predict* and *Update* steps along the temporal axis. In our case, the internal state corresponds to the features recovered from the predicted residuals according to (2), while *Predict* is formulated as the RGN defined in Sec. 3.3 and *Update* is formulated as

$$\hat{\mathbf{d}}_t^+ = \hat{\mathbf{d}}_t^- + \Gamma_t(Z_t - \hat{\mathbf{d}}_t^-), \tag{5}$$

where $\hat{}$ is used to distinguish estimated as opposed to groundtruth values, Z_t is the observed measurement at time t (groundtruth framewise feature), $\mathbf{d}_t^-$ is the prior estimate, (2), and Γ_t is the Kalman Gain defined as

$$\Gamma_t = \psi(Z_{t-1}, \hat{\mathbf{d}}_{t-1}^+; \theta_z). \tag{6}$$

The *Update* corrects the current prediction by balancing the observed measurement, Z_t, and prior estimation, $\hat{\mathbf{d}}_t^-$. Initially, Γ_t is estimated from the error covariance matrix and subsequently updated with a transition function, ψ. We realize ψ as a ConvNet with an LSTM and learnable parameters, θ_z, *c.f.* [6]. The architecture is depicted in Fig. 3, with implementation details provided in Sec. 4.2.

We explicitly incorporate the Kalman filter *Update* step into the training of the RGN, where correction happens after the estimate of $\hat{\mathbf{d}}_t^-$ is obtained, as depicted in Fig. 3. The corrected feature $\hat{\mathbf{d}}_t^+$ is subsequently used for $t+1$ prediction and loss computation thereafter. During training, the Kalman filter has access to true observations, Z_t throughout the video. In testing, however, the Kalman filter only has access to true observations up through the final input partial observation, X_k, and is only applied through that point, as detailed in Sec. 3.6. We find that the instantaneous correction offered by the Kalman filter helps stablize long-term inference, as documented in Sec. 4.4.

3.5. Learning scheme

In our approach, there are two sets of trainable parameters, θ_f and θ_z, that are associated with the kernel motion generator, $\mathcal{G}$, of the residual generative network and the Kalman gain transition, ψ, resp. Both sets of parameters are trained using backpropagation to minimze loss objective functions. We adopt a two stage training strategy that initially learns the θ_f values and subsequently learns the θ_z values, while also refining the θ_f values. We first train θ_f because it is more central to our overall approach in performing the essential prediction, rather than the correction. This design choice conforms to the standard Kalman filter paradigm that presupposes a sane transition module and a corrective module built on rational prior estimations [18]. Nevertheless, ultimately the prediction and correction must work together; so, θ_f and θ_z are jointly trained in our second stage.

The parameters θ_f are optimized with respect to four losses. The first loss pertains to the residuals

$$\mathcal{L}_2^{res}(\theta_f) = ||\mathbf{r}_t - RGN(\mathbf{r}_{t-1}, \mathbf{r}_{t-2}, ..., \mathbf{r}_{t-m}; \theta_f)||_2^2 \tag{7}$$

where m is the temporal window size. (In (7), note that $\mathcal{G}$ is embedded in RGN, but here we suppress the recursive dependence on all previous residuals beyond the current observation window that was given in (3) for the sake of compactness of notation.) The second loss pertains to the features

$$\mathcal{L}_2^{feat}(\theta_f) = ||\mathbf{Z}_t - \hat{\mathbf{d}}_t^-||_2^2 = ||\mathbf{d}_t - (\hat{\mathbf{d}}_{t-1} + \hat{\mathbf{r}}_t)||_2^2. \tag{8}$$

As reported elsewhere [29, 45, 2], $\mathcal{L}_2$ works under the Gaussian assumption that data is draw from a single parameterized Gaussian distribution and thus produces blurry outcomes. To counter this shortcoming, we include an additional two losses by applying the *Gradient Difference Loss*

[29], which emphasizes high frequency content, on both the features and residuals to yield

$$\mathcal{L}_{gdl}^{res}(\theta_f) = ||\frac{\partial}{\partial x}(\mathbf{r}_t - \hat{\mathbf{r}}_t)||_2^2 + ||\frac{\partial}{\partial y}(\mathbf{r}_t - \hat{\mathbf{r}}_t)||_2^2 \quad (9)$$

and

$$\mathcal{L}_{gdl}^{feat}(\theta_f) = ||\frac{\partial}{\partial x}\left(\mathbf{Z}_t - \hat{\mathbf{d}}_t^-\right)||_2^2 + ||\frac{\partial}{\partial y}\left(\mathbf{Z}_t - \hat{\mathbf{d}}_t^-\right)||_2^2. \quad (10)$$

The overall objective function for $\mathcal{G}$ is defined as

$$\mathcal{L}_2^{\mathcal{G}}(\theta_f) = \lambda_1\mathcal{L}_2^{res} + \lambda_2\mathcal{L}_2^{feat} + \lambda_3\mathcal{L}_{gdl}^{res} + \lambda_4\mathcal{L}_{gdl}^{feat}, \quad (11)$$

with the λ_i scalar weighting factors. Note that during the first stage of training, the Kalman filter would not be operating, as it has yet to be trained.

After training the RGN parameters, θ_f, the Kalman gain parameters, θ_z, are trained, while the θ_f parameters values are refined to yield a joint optimization. Now, there are only two losses, both pertaining to the features, $\mathbf{d}$, because that is where the Kalman filter operates. The losses are analagous to (8) and (10), except that they are calculated on the updated posterior $\hat{\mathbf{d}}_t^+$ according to

$$\mathcal{L}_2(\theta_f, \theta_z) = \alpha\mathcal{L}_2^{feat}(\hat{\mathbf{d}}_t^+; \theta_f, \theta_z) + \beta\mathcal{L}_{gdl}^{feat}(\hat{\mathbf{d}}_t^+; \theta_f, \theta_z), \quad (12)$$

with α and β scalar weighting factors.

3.6. Unified model for all observation ratios

Learning a separate model for each observation ratio is not applicable in the real world. To overcome this difficulty, we design a unified training and testing strategy, as follows.

Training. The RGN begins by inputting the very first batch of residuals $[\mathbf{r}_m, \mathbf{r}_{m-1}, \ldots, \mathbf{r}_2]$ and recursively produces all the rest. In other words, our model is trained for predicting the whole sequence from the same starting point, thus entirely ignoring observation ratios.

Testing. Our testing also is invariant to observation ratio by switching modes of the Kalman filter operation so that it only corrects the estimates while data observations are available according to g. For example, when $g = 0.6$, the proposed approach still starts from the beginning observations and propagates to the end, but in two modes: While the observation ratio is not yet reached, *i.e.* $g \in [0.1, 0.6]$, we update predictions via reference to the observed true data by using the Kalman filter update step, (5). After entering $g \in [0.7, 1.0]$, only prediction is performed, (4).

This procedure resembles tracking objects under a Kalman filter: When objects are observed, the system corrects its estimated coordinates based on true observation measurements; however, while objects are occluded, the system extrapolates possible locations based on "up-to-now" system parameter values, *i.e.* only the prediction step is performed.

4. Empirical evaluation

4.1. Datasets and experiment protocol

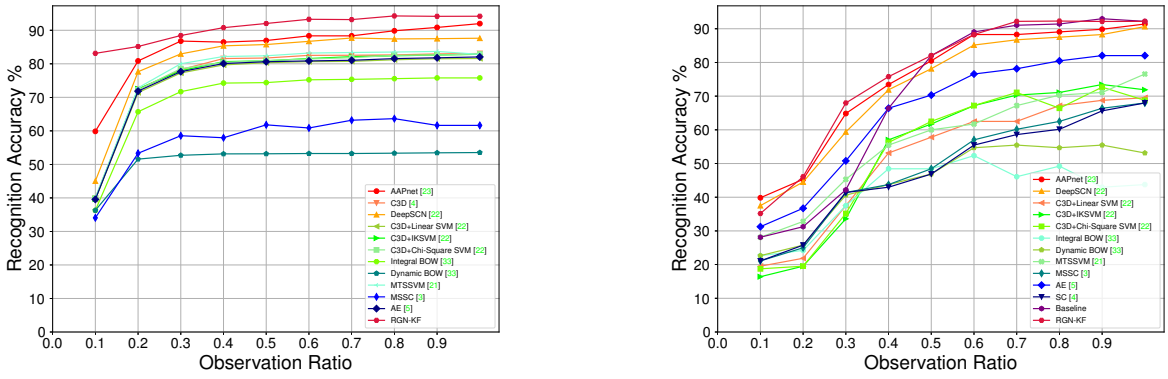
To evaluate our approach, we choose three widely examined datasets, UCF101 [42], JHMDB21 [16] and BIT [20]. UCF101 consists of 13,320 videos of 101 action categories containing a wide range of activities (*e.g.* sports, music and others). JHMDB21, a subset of HMDB [24], contains 928 videos of 21 realistic, nuanced human action categories (*e.g.* catching, throwing, picking). We use the provided RGB images rather than body joints of JHMDB-21. BIT consists of 8 classes of human interactions, with 50 videos per class. Different from the other datasets, BIT has similar behaviors of people in the initial stage of different actions (*e.g.* they tend to be standing still) [23], which leads to challenges from limited discriminatory information.

For all datasets, we use their standard train/test splits: UCF101 and JHMDB21 come with multiple train/test splits and we average over the results in our reporting, unless otherwise noted; BIT has a single train/test split, with the first 34 videos in each class for training and the rest for testing.

We present action classification accuracy as a function of observation ratio, g, which is the ratio of observed to total frames in a video, as used elsewhere [21]. Classification is always based on the concatenation of features derived from the observed frames and those that are predicted. For mid-layer features, which are the subject our propagation, we use the intermediate output of two convolutional layers and two max-poolings $\in \Re^{28\times28\times192}$, unless otherwise noted. This layer is selected because empirical comparison to others generally yielded superior performance; see Sec. 4.5. Beyond the results presented in this section, additional detailed results are provided in the supplement.

4.2. Implementation details

To examine the propagation module with minimal influence from other factors, classifiers for chosen datasets are obtained beforehand. While a pretrained TSN model is available for UCF101 [48], models for JHMDB21 and BIT are not available. To adapt the TSN model to the JHMDB21 and BIT datasets, we append a simple two layer MLP classifier consisting of two hidden layers to TSN pretrained for HMDB-RGB and UCF101-Flow. For JHMDB21, two hidden layers have 32 and 21 activations. For BIT, two hidden layers have 64 and 8 activations. Softmax is used for final probability generation in all cases. During the training process all pretrained weights are frozen. For training of weights added for adaptation to JHMDB21 and BIT, we randomly select 3 RGB samples or 3 Optical Flow samples (each sample has 5 frames) from the videos and get video labels by segment consensus. We employ a learning rate of 0.0001, batch size of 64, *Stochastic Gradient Descent* and the *Adam* optimizer. Data augmentation is the same as for



(a) UCF-101 dataset

(b) BIT dataset

Figure 4: Action Prediction Results on the UCF101 and BIT Datasets at all Observation Ratios $g \in [0.1, 1]$.

the original TSN [48].

Network configurations. For the kernel generator of the RGN, $\mathcal{G}$, stacked residuals are first convolved with a 1×1 kernel that reduces the feature dimension. Then, two residual convolutional blocks [14] with kernel size 3×3, bottleneck dimension 48 and stride 2 are used to capture temporal evolution. Subsequently, with batch and channel axis fixed, flattened spatial features are individually processed with 3 FC layers to produce 3×3, 5×5 and 7×7 kernels. So the shape of feature map is (28, 28, 192*m)-(28, 28, 192)-(28, 28, 192)-(28, 28, 192)-(9, 192), (25, 192) and (49, 192), with m as the empirically selected temporal window size. Convolution is performed on each channel.

For Kalman Gain, Γ_t, a set of convolutional layers with kernel size 3x3 and stride 2 are used to capture the covariance. Each layer is appended with a ReLU layer [13]. The shape of feature map is (28, 28, 128)-(28, 28, 64)-(28, 28, 32)-(28, 28, 1). Subsequently, the flattened feature is taken as input by Γ_t-LSTM to produce Kalman gain, $\Gamma_t \in \Re^{wh}$, which then is reshaped to $\Gamma_t \in \Re^{w \times h}$, corresponding to feature map spatial dimensions. The hidden state of the LSTM has the same size with input feature (784). The gain is then applied according to the update, (5).

Training strategy. We train our model with 4 NIVDIA TITAN X GPUs, under Pytorch [30]. Training of the spatiotemporal feature residual generative network (RGN) employs the *Adam* optimizer and a learning rate 0.005 with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to minimize the loss (11). Empirically, we set $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ with ratios of 1:1:5:5, which places more emphasis on the spatial gradient rather than raw mean square values. The batch size is set to 56. Following initial training of the RGN, we fine-tune it together with the Kalman gain transition ConvNet with LSTM, ψ, to minimze the loss (12). Mini-batch-Stochastic Gradient Descent is used with a learning rate of $2e^{-4}$ and exponential decay of $1e^{-5}$. α and β are set empirically with a ratio of 1:5.

For training on UCF101, we sample 30 frames from each video and use the first 3 to initialize our entire prediction system. For BIT and JHMDB21, we sample 25 frames from each video and use the first 3 to initialize our system. TSN architecture [48] serves to provide feature extraction and classification. We apply our system to the RGB stream for JHMDB21, flow stream for BIT and both streams for UCF101. We make these choices following typical state-of-the-art practice on JHMDB21 (*e.g.* RGB features previously yielded top performance [35, 37]), BIT (*e.g.* flow features greatly outperform spatial features [20, 22]) and UCF101 (*e.g.* two-stream previously yielded top performance [23]).

Once features are generated, no additional modifications to TSN are needed to yield action labels. Generated features are inserted into the selected TSN mid-layer and processed up through the network tower until the MLPs produce probability scores. Video level labeling is gathered by averaging scores from each frame.

4.3. Overall prediction results

UCF-101 dataset. Figure 4 (a) shows comparative results for our algorihm **RGN-KF** *vs.* various alternatives on UCF101. It is seen that **RGN-KF** outperforms all others at all observation ratios, improving accuracy by ≈3-4% on average. The performance improvement is especially striking at lower observation ratios, *e.g.* $g = 0.1$, where we outperform the second best (AAPnet) by 83.3% *vs.* 59.85%.

Notably, AAPnet also builds on TSN; however, it apparently does so less effectively than our approach does. There are likely two reasons for this state of affairs. First, AAPnet is not trained exclusively for inferring action labels, but also for adversarial learning, which might lessen its optimization for action prediction. Second, AAPnet more radically modifies the TSN architecture in aggregating across all frames at a given state of progress, which underlines the fact that our approach may be more adaptable to various architectures as

Table 1: Action Prediction Results on JHMDB21. Following the standard protocol, accuracy results are shown only for the case where initial observations are limited to the first 20% of frames, *i.e.* $g = 0.2$.

Method	Accuracy (%)
ELSTM [35]	55
Within-class Loss [28]	33
DP-SVM [41]	5
S-SVM [41]	5
Where/What [40]	10
Context-fusion [15]	28
RBF-RNN [37]	73
Ours	**78**
Baseline	74

it has less impact on their native operations.

BIT. Figure 4 (b) shows comparative results for our algorithm *vs*. various alternatives on BIT. It is seen that our results are equal to or better than all others, except at the lowest observation ratio, $g = 0.1$. For example, compared with AAPnet, our approach achieves 67.96% accuracy at $g = 0.3$, which is 3.13% higher.

In interpreting the results on BIT it is important to recall that the beginning and ending portions of the videos tend be very similar in appearance (*e.g*. two people standing facing one another), so that the most discriminatory information largely is limited to the intermediate portions. Correspondingly, there is a tendency for rapid performance rises after the initial portion, which levels out in the final portion. In our case, a peak performance of 92.28% at $g = 0.7$ increases that at the previous ratio by 4%, whereas AAPnet achieves no significant increase (0.78%) at the same stage.

Given that we train a modified TSN architecture in adapting TSN to BIT, Sec. 4.2, we compare how well that modified architecture works when forced to classify on just the initially provided frames without propagation into the future. These results are shown as Baseline in Fig. 4 b. It is seen that by propagating into the future our approach exceeds the baseline by large margins when $g \in [0.1, 0.4]$. For higher observation ratios, as the discriminating parts of the input videos become available to the baseline (as well as our full approach), performance becomes on par.

JHMDB21. The standard reporting protocol on JHMDB21 is to report recognition accuracy only when the initial 20% of the videos are observed, *i.e*. $g = 0.2$, which we show in Table 1. It is seen that our algorithm once again is the top performer, *e.g*. exceeding the second best by 5%. We also provide a baseline comparison, where we compare to classification made purely on the basis of adapting the TSN architecture to the JHMDB21 dataset, analogous to the baseline comparison provided on BIT. Once again, it is seen that our full propagation approach adds considerably to the performance of the baseline alone.

Table 2: Accuracy results for different temporal propagation approaches on JHMDB21 split 1. **Org** denotes applying motion kernel transformation on original features, **Res** denotes residual propagation and **KF** denotes inclusion of the Kalman filter. For ConvLSTM, (3x3), 128 & 192 represent kernel, hidden state & feature dimension, resp.

Temporal Propagation Approach	Accuracy (%)
ConvLSTM(3x3)-128-192-**Org**	71.1
ConvLSTM(3x3)-128-192-**Org-KF**	73.4
ConvLSTM(3x3)-128-192-**Res**	76.8
ConvLSTM(3x3)-128-192-**Res-KF**	77.1
RGN-**Org**	70.9
RGN-**Org-KF**	74.4
RGN-**Res**	77.4
RGN-**Res-KF**	**78.3**

4.4. Influence of temporal model

In this section, we examine the influence of different temporal modeling approaches to feature propagation using JHMDB21, with ConvLSTM as an extra baseline, *cf.* [36, 2]; see Table 2. For both scenarios, we find that propagation on residuals is superior to propagation on raw features and the Kalman filter provides further benefits. Performance of ConvLSTM is on par with our RGN approach applied to the original features without the Kalman filter; however, for all other configurations our RGN approach performs better. Overall, we find that our full approach to temporal modeling (mid-layer convolutional feature residuals, RGN propagation and Kalman filtering) yields best performance.

4.5. Influence of feature layers

We now examine the influence of different intermediate feature spaces on prediction. We consider layers that yield feature maps of [56, 56, 64], [28, 28, 192], [14, 14, 512] and [7, 7, 1024], where [w, h, c] indicate the width, height and number of channels, resp. Table 3 shows the results. For JHMDB21 and BIT, the [28, 28, 192] feature stage almost always achieves best results. Moreover, deeper layers, [14, 14, 512] and [7, 7, 1024], are more useful than the shallower layer [56, 56, 64]. This pattern of results may be explained by the earliest layer not providing adequate abstraction from the raw input, while the highest layers have lost too much distinguishing detail. Interestingly, for UCF101 different feature stages have less impact on accuracy. This may be due to the fact that UCF101 is generally less difficult than the other datasets, as indicated by the fact that for any given observation ratio, g, in Table 3 the results on UCF101 are always better than for the others; correspondingly, the specifics of feature selection are less important. More generally, however, the results of Table 3 support our use of intermediate layer features, especially as the prediction task becomes more difficult.

Table 3: Prediction Accuracy (%) at Various Intermediate Feature Stages with **RGN-KF**, Ordered by Decreasing Spatial Receptive Field Size. Observation ratio $g \in \{0.2, 0.4, 0.6, 0.8\}$ for UCF101 and BIT datasets. Set $g = 0.2$ for JHMDB21.

	UCF-101				BIT				JHMDB-21
observation ratio	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2
56x56x64	85.16	88.64	91.10	92.10	35.94	49.22	75.78	84.38	75.74
28x28x192	85.16	90.78	92.03	93.19	46.09	75.78	88.28	88.82	78.30
14x14x576	85.09	90.32	92.12	93.06	40.62	76.65	85.94	87.50	77.43
7x7x1024	84.94	90.07	91.60	92.77	42.06	75.78	87.50	87.50	77.03

4.6. Visualization of feature residuals

To understand further why intermediate layer features and their residuals are especially useful for action prediction, we show comparative visualizations as well as associated statistics. Figure 5 provides an example derived from the action *baseball-swing*. It is seen that the earliest layer feature map concentrates on low-level features (*e.g.* lines and edges) that may be too closely tied to a specific example, rather than the action class. In contrast, the latest layer feature map tends to lose too much distinguishing detail (*e.g.* merely a blob in the vicinity of the actor at the top-layer). Comparatively, the mid-layer features tend to concentrate on the actor, but also delineate details of the actors parts. In comparing the raw features to their residuals, it is seen that the residuals concentrate more on temporal changes, which are good for propagating variations into the future without redundant information. Thus, intermediate layer residuals appear to capture information that is especially useful for action prediction.

The provided visualization, Fig. 5, suggests that the residuals provide a more sparse (and hence compact) representation compared to the features per se. To quantify this observation, we define feature sparsity as the percentage of near-zero points (absolute value < 0.01) vs. total points. Fig. 6 shows comparative results for original features and their residuals. It is seen that the residuals have approximately five times the sparsity of the originals, which quantitatively confirms the relative sparsity of the residuals.

Overall, both the visualizations and the quantitative analysis confirm that mid-layer feature residuals are especially information laden for action prediction.

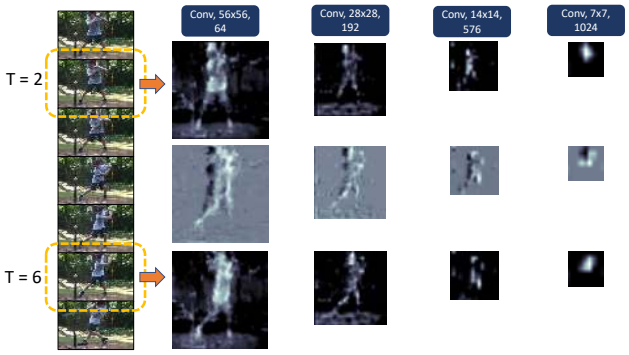


Figure 5: Visualization of Features and Residuals. The example shows a sequence of frames for the action *baseball-swing* along the left hand side. Example feature maps extracted at various layers are shown along the upper/bottom row, while their residuals are shown along the middle row.

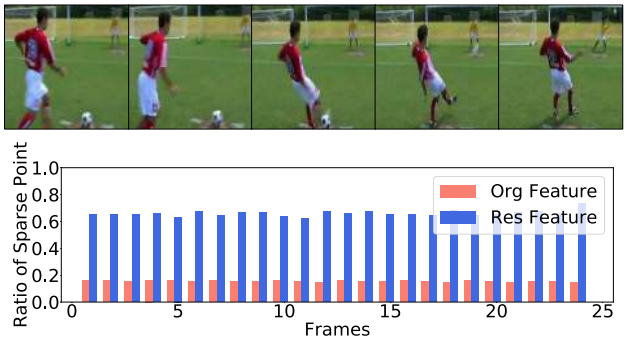


Figure 6: Sparsity Comparison of Features and Residuals. The top row shows frames from a video of a *kicking* action. The bottom row shows sparsity as the ratio of near-zero value points (absolute value < 0.01) to total points over time. On average, residual and original feaure points are 65% and 14% sparse, resp.

5. Conclusions

In this paper, we have presented a novel spatiotemporal feature residual propagation approach to predicting the action label of a video before action execution ends. Our approach learns to propagate framewise residuals in feature space to complete partial observations. The approach enjoys the advantages of the spatial structure preservation of mid-layer ConvNet features, compact representation that captures essential information via residual processing and long-term stability via instaneous Kalman filter corrections. The approach has been evaluated on the UCF101, JHMDB21 and BIT-Interaction datasets, where it sets a new state-of-the-art in comparison to a variety of alternative approaches. Our methodology has potential to be extended for application to other video-based computer vision tasks, which will be studied in the future.

Acknowledgments

This research was funded in part by NSERC, the Ontario Trillium Scholarships and CFREF VISTA. We thank Soo Min Kang for many fruitful discussions.

References

[1] Padmanabhan Anandan. A Computational Framework and An Algorithm for The Measurement of Visual Motion. *IJCV*, 2(3):283–310, 1989. 2

[2] Wonmin Byeon, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. ContextVP: Fully Context-Aware Video Prediction. In *ECCV*, 2018. 4, 7

[3] Yu Cao, Daniel Barrett, Andrei Barbu, Siddharth Narayanaswamy, Haonan Yu, Aaron Michaux, Yuewei Lin, Sven Dickinson, Jeffrey Mark Siskind, and Song Wang. Recognize Human Activities from Partially Observed Videos. In *CVPR*, 2013. 1, 2, 6

[4] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *CVPR*, 2017. 2, 6

[5] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized Denoising Autoencoders for Domain Adaptation. *arXiv preprint arXiv:1206.4683*, 2012. 6

[6] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. In *ICCV*, 2017. 4

[7] Ali Diba, Vivek Sharma, and Luc Van Gool. Deep Temporal Linear Encoding Networks. In *CVPR*, 2017. 2

[8] Alexey Dosovitskiy and Thomas Brox. Inverting Visual Representations with Convolutional Networks. In *CVPR*, 2016. 2

[9] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. Spatiotemporal Multiplier Networks for Video Action Recognition. In *CVPR*, 2017. 2

[10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. In *CVPR*, 2016. 2

[11] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised Learning for Physical Interaction through Video Prediction. In *NIPS*, 2016. 3

[12] Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Commun. ACM*, 34:46–58, 1991. 2

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, 2011. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 6

[15] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent Neural Networks for Driver Activity Anticipation via Sensory-Fusion Architecture. In *ICRA*, 2016. 7

[16] Hueihan Jhuang, Juergen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J Black. Towards Understanding Action Recognition. In *ICCV*, 2013. 5

[17] Jianbo Jiao, Wei-Chih Tu, Shengfeng He, and Rynson W. H. Lau. FormResNet: Formatted Residual Learning for Image Restoration. In *CVPRW*, 2017. 2

[18] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 4

[19] Yu Kong, Shangqian Gao, Bin Sun, and Yun Fu. Action Prediction From Videos via Memorizing Hard-to-Predict Samples. In *AAAI*, 2018. 2

[20] Yu Kong, Yunde Jia, and Yun Fu. Interactive Phrases: Semantic Descriptions for Human Interaction Recognition. *IEEE Trans. PAMI*, 36(9):1775–1788, 2014. 5, 6

[21] Yu Kong, Dmitry Kit, and Yun Fu. A Discriminative Model with Multiple Temporal Scales for Action Prediction. In *ECCV*, 2014. 1, 2, 5, 6

[22] Yu Kong, Zhiqiang Tao, and Yun Fu. Deep Sequential Context Networks for Action Prediction. In *CVPR*, 2017. 2, 6

[23] Yu Kong, Zhiqiang Tao, and Yun Fu. Adversarial Action Prediction Networks. *IEEE Trans. PAMI*, 2019. 2, 5, 6

[24] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: A Large Video Database for Human Motion Recognition. In *ICCV*, 2011. 5

[25] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A Hierarchical Representation for Future Action Prediction. In *ECCV*, 2014. 1

[26] Zhenzhong Lan, Yi Zhu, Alexander G Hauptmann, and Shawn Newsam. Deep Local Video Feature for Action Recognition. In *CVPRW*, 2017. 2

[27] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. Deep Kalman Filtering Network for Video Compression Artifact Reduction. In *ECCV*, 2018. 2, 4

[28] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning Activity Progression in LSTMs for Activity Detection and Early Detection. In *CVPR*, 2016. 7

[29] Michaël Mathieu, Camille Couprie, and Yann LeCun. Deep Multi-Scale Video Prediction Beyond Mean Square Error. *arXiv preprint arXiv:1511.05440*, 2015. 4, 5

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 6

[31] Anurag Ranjan and Michael J Black. Optical Flow Estimation Using a Spatial Pyramid Network. In *CVPR*, 2017. 2

[32] Fitsum A. Reda, Guilin Liu, Kevin J. Shih, Robert Kirby, Jon Barker, David Tarjan, Andrew Tao, and Bryan Catanzaro. SDCNet: Video Prediction Using Spatially-Displaced Convolution. In *ECCV*, 2018. 3

[33] Michael S Ryoo. Human Activity Prediction: Early Recognition of Ongoing Activities from Streaming Videos. In *ICCV*, 2011. 1, 2, 6

[34] Michael S Ryoo and Jake K Aggarwal. Spatio-Temporal Relationship Match: Video Structure Comparison for Recognition of Complex Human Activities. In *ICCV*, 2009. 2

[35] Mohammad Sadegh Aliakbarian, Fatemeh Sadat Saleh, Mathieu Salzmann, Basura Fernando, Lars Petersson, and Lars Andersson. Encouraging LSTMs to Anticipate Actions Very Early. In *ICCV*, 2017. 2, 6, 7

[36] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NIPS*, 2015. 7

[37] Yuge Shi, Basura Fernando, and Richard Hartley. Action Anticipation with RBF Kernelized Feature Mapping RNN. In *ECCV*, 2018. 2, 6, 7

[38] Gurkirt Singh, Suman Saha, and Fabio Cuzzolin. Predicting Action Tubes. In *ECCVW*, 2018. 2

[39] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip HS Torr, and Fabio Cuzzolin. Online Real-Time Multiple Spatiotemporal Action Localisation and Prediction. In *ICCV*, 2017. 2

[40] Khurram Soomro, Haroon Idrees, and Mubarak Shah. Predicting the Where and What of Actors and Actions through Online Action Localization. In *CVPR*, 2016. 7

[41] Khurram Soomro, Haroon Idrees, and Mubarak Shah. Online Localization and Prediction of Actions and Interactions. *IEEE Trans. PAMI*, 41(2):459–472, 2018. 7

[42] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[43] Frederick Tung and Greg Mori. Deep Neural Network Compression by In-Parallel Pruning-Quantization. *IEEE Trans. PAMI*, 2019. 2

[44] Gül Varol, Ivan Laptev, and Cordelia Schmid. Long-Term Temporal Convolutions for Action Recognition. *IEEE Trans. PAMI*, 40(6):1510–1517, 2018. 2

[45] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing Motion and Content for Natural Video Sequence Prediction. In *ICLR*, 2017. 4

[46] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating Visual Representations from Unlabeled Video. In *CVPR*, 2016. 2, 3

[47] Michelle W. Voss. Understanding the Mind of the Elite Athlete. *Scientific American*, 2010. 1

[48] Limin Wang, Yuanjun Xiong, Zhenchang Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *ECCV*, 2016. 2, 3, 5, 6

[49] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed Video Action Recognition. In *CVPR*, 2018. 2

[50] Matthew D Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *ECCV*, 2014. 2

[51] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep Feature Flow for Video Recognition. In *CVPR*, 2017. 2