Exemplar-Based Early Event Prediction in Video

Zekun Zhang[1]
zekzhang@cs.stonybrook.edu
Farrukh M. Koraishy[2]
farrukh.koraishy@stonybrookmedicine.edu
Minh Hoai[1]
minhhoai@cs.stonybrook.edu

[1] Department of Computer Science
[2] Division of Nephrology
Department of Medicine

Stony Brook University
New York, 11794, USA

Abstract

Observing a video stream and being able to predict target events of interest before they occur is an important but challenging task due to the stochastic nature of visual events. This task requires a classifier that can separate the precursory signals that lead to the events and the ones that do not. However, a naïve approach for training this classifier would require seeing many examples of the target events before a model with high precision can be obtained. In this paper, we propose a method for early prediction of visual events based on an ensemble of exemplar predictors. Each exemplar predictor is associated with an instance of the target event, being trained to separate the target event from negative samples. The exemplar predictors can be calibrated and integrated to create a stronger predictor. Experiments on several datasets show that the proposed exemplar-based framework outperforms other methods, yielding higher precision given fewer training samples. Our code and datasets can be found at github.com/cvlab-stonybrook/EnEx.

1 Introduction

Being able to monitor a video stream and forecast an event of interest before it happens is profound capability that has numerous applications in a wide range of fields, ranging from entertainment and education to healthcare and security. For example, the ability to predict illicit events at retail stores, such as shoplifting and vandalism, will enable prevention and quick response, minimizing damages and improving security. In this paper, we aim for an algorithmic framework that will turn a video camera into a predictor that continually improves its performance based on delayed self-supervision signals. We focus on scenarios where the camera can observe the same type of scene for an extended period of time and visual events from a target category of interest occur infrequently but repeatedly.

In general, it is difficult to predict visual events due to the stochastic nature of visual data and the limited field of view of the camera. An event can be predicted in advance only if there is some precursory evidence and that evidence can be detected. However, many visual events, such as picking up a phone or opening a door, might occur without any prior visual indication. Even when prior visual indication does exist, it might not be observable or detectable from the perspective of the camera.

The difficulties of predicting visual events can be seen from the training perspective. Let us refer to a video segment that precedes a target event of interest as *pre-positive*, and a video segment that is not followed by a target event as *pre-negative*. For early prediction, we need to distinguish pre-positive video segments from pre-negative ones. To do so, we can collect the previously observed pre-positive and pre-negative video segments and train a classifier to separate these two sets. However, visual data is highly stochastic and visual events can occur without any prior indication. Many pre-positive video segments are similar or even identical to pre-negative segments, and for a naïve approach, much of the training effort would be wasted on the failed attempt to separate the non-separable, and it will take many examples to identify the predictable patterns.

The premise of our work is that not all target events can be predicted, but many can be predicted with a high level of certainty. Instead of aiming to predict everything and ending up with a predictor that is not confident about anything, we propose to focus on identifying and detecting the precursory patterns that will surely lead to a target event. In other words, we should prioritize high precision over high recall. Having a predictor with high precision is very important for many applications, especially when it is much more likely for the target event not to happen and early prediction is only meaningful when the event does happen.

We develop here a non-parametric method for visual event prediction based on an ensemble of exemplar classifiers. Each exemplar classifier is obtained by training a classifier to separate a single pre-positive video segment from many pre-negatives. Each exemplar classifier is like a local distance function that is discriminatively trained, representing a candidate precursory pattern that should be monitored. Our approach does not suffer from the drawbacks of parametric classifiers, which treat all pre-positive segments as a whole, implicitly assuming that pre-positive segments are visually related and separable from the pre-negatives.

Our work is inspired by the success of previous exemplar models used for image classification [59], object segmentation [27], object detection [7, 28], and action recognition [57]. The philosophy of our work is also similar to local learning [2] and local distances [13, 14]. Our work is most similar to the object detection framework of [28], where they also train a set of exemplar SVMs, one for each positive example. Their approach, however, is inefficient for continual learning, which is an issue that we address in this paper. Moreover, we propose a more robust approach for calibrating and combining multiple exemplar classifiers.

Experimental results on multiple visual events show that the proposed method outperforms both parametric baseline predictors and other exemplar based methods in terms of prediction precision. The proposed method is faster to train and more robust to the noisy training data.

2 Related Work

We propose a framework that can directly forecast high-level semantic properties of events, unlike methods for predicting mid-level features [34, 41, 45, 46] or low-level feature maps such as raw video frames [25, 30, 51, 54], optical flow maps [17, 47]), segmentation maps [26], dynamic images [36], and human poses [12, 24, 43, 48, 49, 50]. There are methods for recognizing partial human actions by training frame-based classifiers [10, 29, 38, 58] or segment-based classifiers [20, 21, 35, 37, 49], but these methods are for early recognition not early prediction. More recently, many end-to-end neural network based methods are proposed for early recognition or anticipation [1, 11, 15, 16, 40], but they rely on large offline datasets with high-quality and dense annotation to learn good video representation and classifiers. None of the aforementioned works are designed for self-supervised continual learning, where target events happen infrequently. They do not consider the difficulties of separating the non-separable and they do not aim for a predictor with high precision.

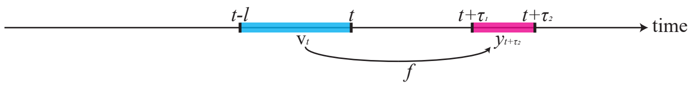


Figure 1: Prediction task: Our goal is to have a predictor f that analyzes information in time $[t-l, t]$ to predict if a target event will occur between $t+\tau_1$ and $t+\tau_2$.

One might assume the sequence of video frames or events as a Markov process and use methods such as Hidden Markov Models [33] for prediction. This is the framework of many existing prediction methods such as [6, 9, 31, 39], but this approach is not suitable for early prediction of high-level visual events for two reasons. First, visual events are complex and do not usually satisfy the Markov property. Second, to model a video with a Markov process, we need to specify a state space, and this approach is only appropriate if the states are what we aim to predict. Similarly, time series forecasting techniques, such as ARMA [53] and ARIMA [3], can be used for forecasting future observation values. These methods are appropriate if we are interested in predicting raw observation values such as the Dow Jones Utilities Index [4]. However, knowing raw observation values is not enough to predict high-level semantic events. Of course, forecasting the future observations may be a good first step for prediction, but this two-stage approach has a disadvantage because the former is usually more difficult than the latter.

3 Framework Description

We envision a self-supervised learning framework that will turn a video camera into a predictor that continually improves its performance based on delayed self-supervised signals. Once an event has been retrospectively detected, the predictor will be improved based on whether the predictor has predicted it correctly. The core of our framework is a predictor that can predict with lead time duration $[\tau_1, \tau_2]$. At time t, the predictor analyzes the video sequence up until t and predicts whether an event of interest will occur between times $t+\tau_1$ and $t+\tau_2$, inclusively. Let $\mathbf{v}_t$ be the feature representation for the video segment in time $[t-l, t]$, where l is the length of the input window. τ_1, τ_2 and l are all problem-specific parameters, and $0 < \tau_1 \leq \tau_2$. Let y_t be the binary variable that indicates whether the event of interest occurs between $t-(\tau_2-\tau_1)$ and t. We assume that y_t will be known at time t, but our goal at time t is to predict $y_{t+\tau_2}$, as illustrated in Figure 1.

Suppose we can keep collecting labeled training examples from our past observations. At time t, the accumulated set of labeled training examples is $\{(\mathbf{v}_i, y_{i+\tau_2}) | i = 1, \ldots, t-\tau_2\}$. In general, there are not many positive examples, and let $\mathcal{P} = \{\mathbf{z}_1, \ldots, \mathbf{z}_p\}$ denote this subset. On the other hand, there are many negative samples, and we do not need to use all of them for training. We will sample two disjoint subsets of negative examples $\mathcal{N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and $\mathcal{N}^h = \{\mathbf{x}_1^h, \ldots, \mathbf{x}_n^h\}$. The former will be used for training the exemplar predictors, while the latter will be used for calibration. The training and calibration of the exemplar predictors will be explained in details below.

In this section, we describe our method for early prediction based on an ensemble of exemplar predictors. Each exemplar predictor $f_{\mathbf{z}}(\cdot)$ is obtained by training a classifier to separate a *single* pre-positive video segment $\mathbf{z}$ from many pre-negatives. We propose a novel formulation to efficiently train multiple exemplar classifiers together, making it suitable for continual learning. We also propose a novel approach for calibrating and combining the exemplar classifiers using rank pooling to create a stronger predictor.

Notation. We will use bold uppercase letters to denote matrices (e.g., $\mathbf{D}$), bold lowercase letters to denote column vectors (e.g., $\mathbf{d}, \alpha$). $\mathbf{d}_j$ represents the j^{th} column of the matrix $\mathbf{D}$, while d_j is the j^{th} element of the column vector $\mathbf{d}$. d_{ij} denotes the scalar in the row i and column j of the matrix $\mathbf{D}$ and the i^{th} element of the column vector $\mathbf{d}_j$. Non-bold letters represent scalar variables. $1_n \in \mathbb{R}^n$ is a column vector of ones. $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is an identity matrix.

3.1 Exemplar predictor for a positive example

Given a particular positive example $\mathbf{z} \in \mathcal{P}$ and the set of $\mathcal{N} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of negative examples, we propose to learn the exemplar predictor based on Kernel Ridge Regression [42]. Specifically, we consider the RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x}-\mathbf{x}'||_2^2)$, where γ is set to the inverse of the average squared distance between elements of $\mathcal{P}$ and $\mathcal{N}$. Let $\phi(\mathbf{x})$ be the implicit feature function that maps a data point $\mathbf{x}$ to the feature space. For the exemplar predictor, we will learn a weight vector $\mathbf{w}$ and a bias term so that: $\mathbf{w}^T\phi(\mathbf{z}) + b = 1$ and $\mathbf{w}^T\phi(\mathbf{x}_i) + b \approx 0$. We can learn $\mathbf{w}$ and b as an optimization: minimize$_{\mathbf{w},b}$ $\lambda||\mathbf{w}||^2 + \sum_{i=1}^n(\mathbf{w}^T\phi(\mathbf{x}_i) + b)^2$, s.t. $\mathbf{w}^T\phi(\mathbf{z}) + b = 1$, where λ is a regularization parameter. From the constraint, we have $b = 1 - \mathbf{w}^T\phi(\mathbf{z})$ and the objective is equivalent to:

$$\underset{\mathbf{w}}{\text{minimize}}\, \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(1 - \mathbf{w}^T(\phi(\mathbf{z}) - \phi(\mathbf{x}_i)))^2. \tag{1}$$

Let $\mathbf{l}_i = \phi(\mathbf{z}) - \phi(\mathbf{x}_i)$ and $\mathbf{L} = [\mathbf{l}_1, \ldots, \mathbf{l}_n]$. The solution of the above optimization problem can be found by setting the derivative to zero: $\mathbf{w} = (\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{L}\mathbf{1}_n$. On the other hand, we have $(\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)\mathbf{L} = \mathbf{L}(\mathbf{L}^T\mathbf{L} + \lambda\mathbf{I}_n)$, then $\mathbf{L}(\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1} = (\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{L}$. So we must have $\mathbf{w} = \mathbf{L}(\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{1}_n$. Thus the weight vector has the form $\mathbf{w} = \mathbf{L}\alpha$, where $\alpha = (\mathbf{L}\mathbf{L}^T + \lambda\mathbf{I}_n)^{-1}\mathbf{1}_n$. In practice, there is no need to explicitly recover $\mathbf{w}$. It is sufficient to solve for α, and the decision value for any data point $\mathbf{q}$ can be calculated:

$$f_{\mathbf{z}}(\mathbf{q}) = \mathbf{w}^T\phi(\mathbf{q}) + b = \sum_{i=1}^{n}\alpha_i(k(\mathbf{z},\mathbf{q}) - k(\mathbf{x}_i,\mathbf{q}) - k(\mathbf{z},\mathbf{z}) + k(\mathbf{x}_i,\mathbf{z})) + 1. \tag{2}$$

3.2 Efficient training of multiple predictors

For a single exemplar predictor, we need to find α using: $\alpha = \mathbf{C}^{-1}\mathbf{1}_n$, where $\mathbf{C} = \mathbf{L}^T\mathbf{L} + \lambda\mathbf{I}$. Let $\mathbf{K}$ be the $n\times n$ kernel matrix with $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. Let $\mathbf{k}_\mathbf{z} = [k(\mathbf{x}_1,\mathbf{z}), \ldots, k(\mathbf{x}_n,\mathbf{z})]^T$. We have $\mathbf{C} = \mathbf{L}^T\mathbf{L} + \lambda\mathbf{I}_n = \mathbf{K} - \mathbf{k}_\mathbf{z}\mathbf{1}_n^T - \mathbf{1}_n^T\mathbf{k}_\mathbf{z} + k(\mathbf{z},\mathbf{z}) + \lambda\mathbf{I}_n$. Since we use an RBF kernel, $k(\mathbf{z},\mathbf{z}) = 1$. Let $\mathbf{A} = \mathbf{K} + 1 + \lambda\mathbf{I}_n$ and $\mathbf{B} = \mathbf{A} - \mathbf{k}_\mathbf{z}\mathbf{1}_n^T$. We have $\mathbf{C} = \mathbf{B} - \mathbf{1}_n^T\mathbf{k}_\mathbf{z}$. Using Sherman-Morrison [18] we have:

$$\alpha = \mathbf{C}^{-1}\mathbf{1}_n = \left(\mathbf{B}^{-1} + \frac{\mathbf{B}^{-1}\mathbf{1}_n\mathbf{k}_\mathbf{z}^T\mathbf{B}^{-1}}{1 - \mathbf{k}_\mathbf{z}^T\mathbf{B}^{-1}\mathbf{1}_n}\right)\mathbf{1}_n = \frac{\mathbf{B}^{-1}\mathbf{1}_n}{1 - \mathbf{k}_\mathbf{z}^T\mathbf{B}^{-1}\mathbf{1}_n} = \frac{\mathbf{h}_\mathbf{z}}{1 - \mathbf{k}_\mathbf{z}^T\mathbf{h}_\mathbf{z}}, \tag{3}$$

where $\mathbf{h}_\mathbf{z} = \mathbf{B}^{-1}\mathbf{1}_n$. Again, using Sherman-Morrison:

$$\mathbf{h}_\mathbf{z} = \mathbf{B}^{-1}\mathbf{1}_n = \left(\mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\mathbf{k}_\mathbf{z}\mathbf{1}_n^T\mathbf{A}^{-1}}{1 - \mathbf{1}_n^T\mathbf{A}^{-1}\mathbf{k}_\mathbf{z}}\right)\mathbf{1}_n = \mathbf{u}^* + \frac{\mathbf{A}^{-1}\mathbf{k}_\mathbf{z}\mathbf{1}_n^T\mathbf{u}^*}{1 - \mathbf{k}_\mathbf{z}^T\mathbf{u}^*} = \mathbf{u}^* + \frac{\mathbf{1}_n^T\mathbf{u}^*}{1 - \mathbf{k}_\mathbf{z}^T\mathbf{u}^*}\mathbf{u}_\mathbf{z}, \tag{4}$$

where $\mathbf{u}^* = \mathbf{A}^{-1}\mathbf{1}_n, \mathbf{u}_\mathbf{z} = \mathbf{A}^{-1}\mathbf{k}_\mathbf{z}$. Since $\mathbf{A}$ and $\mathbf{u}^*$ do not depend on the exemplar instance $\mathbf{z}$, the above suggests an efficient method to train multiple exemplar predictors. This method requires computing $\mathbf{A}^{-1}$ only once, so the total complexity for p exemplars is the complexity of computing $\mathbf{A}^{-1}$ and the complexity of computing $\mathbf{u}_\mathbf{z}$ and $\mathbf{h}_\mathbf{z}$ p times. The total complexity is $O(n^3 + pn^2)$. Here we assume the complexity for computing $\mathbf{A}^{-1}$ is $O(n^3)$ but there are algorithms with lower complexity of $O(n^{2.3727})$. In practice, for numerical stability, there is no need to solve for $\mathbf{A}^{-1}$. We can find $\mathbf{u}^*$ and $\mathbf{u}_\mathbf{z}$ for $\mathbf{z} = \mathbf{z}_1, \ldots, \mathbf{z}_p$ using the backslash operation: $[\mathbf{u}^*, \mathbf{u}_{\mathbf{z}_1}, \ldots, \mathbf{u}_{\mathbf{z}_p}] = [\mathbf{1}_n, \mathbf{k}_{\mathbf{z}_1}, \ldots, \mathbf{k}_{\mathbf{z}_p}]\backslash\mathbf{A}$.

3.3 Calibrating exemplar predictors

The exemplar classifiers are efficiently trained together, but the classifiers are independent. To compare and combine them, we will need to calibrate their prediction scores. One approach is to use Platt scaling [32], but it is unsuitable for our scenarios due to the imbalance between positive and negative classes and the need to prioritize high precision over recall. Here, we propose to use a non-parametric method based on precision values instead.

Consider a particular exemplar $\mathbf{z}$ with the corresponding decision function $f_{\mathbf{z}}(\cdot)$ given in Equation (2), we will first learn a function g that maps the raw output of f to the corresponding precision value, which is calculated based on the set of other positive examples other than $\mathbf{z}$ (i.e., $\mathcal{P}_{\mathbf{z}} = \mathcal{P} \setminus \{\mathbf{z}\}$) and the set of holdout negatives $\mathcal{N}^h$. In other word, $g(\theta)$ is the precision value of the detector that uses $f_{\mathbf{z}}$ as the decision function and θ the detection threshold, i.e., $g(\theta) = a/(a+b)$, where a and b are the number of positive and negative examples with decision values greater or equal to θ.

We further ensure that the calibration function is monotonic non-decreasing by defining $\hat{g}(\theta) := \sup_{\theta' \leq \theta} g(\theta')$. Hereafter, we will use $f^c_{\mathbf{z}}$ to denote the calibrated decision function for the exemplar $\mathbf{z}$, i.e., $f^c_{\mathbf{z}}(\mathbf{q}) := \hat{g}(f_{\mathbf{z}}(\mathbf{q}))$, for any query point $\mathbf{q}$. $f^c_{\mathbf{z}}(\mathbf{q})$ is essentially the estimated probability for the query point $\mathbf{q}$ to be positive, where the estimation was performed on the holdout data.

3.4 Combining predictors: the EnEx predictor

Given a query data $\mathbf{q}$ and the calibrated scores of the exemplar classifiers, our objective here is to calculate a value to indicate the likelihood for $\mathbf{q}$ being positive. Recall that the calibrated score of an exemplar predictor is the precision value evaluated on a holdout dataset, so the calibrated score is the direct estimate for the probability of the query point to be positive. One approach is to use mean pooling, but the estimated probability will be corrupted by unreliable prediction outputs. This is because each exemplar predictor is trained with a single positive exemplar, so its decision is expected to be reliable only for data points that are sufficiently similar to the exemplar. A more intuitive approach is to use max pooling (as in [28]), assigning the decision value for $\mathbf{q}$ based on the exemplar classifier that has the highest calibrated score for $\mathbf{q}$. Max pooling, however, is not robust [22] given the noisiness of individual exemplar classifiers. In this work, we propose to use Orderly Weighted Averaging [22, 52, 55, 56]. This consists of two steps: (1) rank the calibrated scores of the exemplar predictors from highest to lowest, and (2) use a learned weight vector to combine the scores.

For a query data $\mathbf{q}$, let $\mathbf{s_q}$ denote the vector of calibrated scores $\mathbf{s_q} = [f^c_{\mathbf{z}_1}(\mathbf{q}), \ldots, f^c_{\mathbf{z}_p}(\mathbf{q})]^T$. We first calculate $\mathbf{s_q}$ for $\mathbf{q}$ in $\mathcal{P}$ and $\mathcal{N}^h$. If $\mathbf{q}$ is from $\mathcal{P}$, the entry of $\mathbf{s_q}$ that comes from the exemplar classifier $\mathbf{q}$ cannot be used, because $\mathbf{q}$ was used for training the classifier. We therefore treat it as a missing value and replace it with the average value. We will then sort the entries of each vector $\mathbf{s_q}$ from the highest to the lowest, and learn a classifier to separate $\{sort(\mathbf{s_q}) | \mathbf{q} \in \mathcal{P}\}$ from $\{sort(\mathbf{s_q}) | \mathbf{q} \in \mathcal{N}^h\}$. Following [22], we learn a weight vector $\mathbf{v}$ by optimizing the following linear program:

$$\min_{\mathbf{v},b}. \frac{1}{|\mathcal{P}|} \sum_{\mathbf{q} \in \mathcal{P}} \max(\mu - (\mathbf{v}^T sort(\mathbf{s_q}) + b), 0) + \frac{1}{|\mathcal{N}^h|} \sum_{\mathbf{q} \in \mathcal{N}^h} \max(\mu + (\mathbf{v}^T sort(\mathbf{s_q}) + b), 0),$$

$$\text{s.t. } v_1 \geq v_2 \geq \cdots \geq v_p \geq 0, \text{and } v_1 + v_2 + \cdots + v_p = 1. \tag{5}$$

Here, μ is the margin term of the Hinge loss, and it is set to: $0.5 \times |\mathbb{E}_{\mathbf{q} \in \mathcal{P}} \|\mathbf{s}_q\|_1 - \mathbb{E}_{\mathbf{q} \in \mathcal{N}^h} \|\mathbf{s}_q\| n_1|$. The learned weight vector $\mathbf{v}$ will be then used for combining the ranked calibrated scores. We will refer to the combined exemplar predictor as the **EnEx** predictor.

Table 1: Results of predicting OpenDoor events with prediction of lead times 2*s* and 4*s*

	$\tau_1 = 2$s			$\tau_1 = 4$s		
model	@0.1	@0.3	@1.0	@0.1	@0.3	@1.0
MLP	54.5	54.1	31.1	45.8	41.0	21.3
LogReg	65.9	66.7	38.2	46.3	43.1	22.4
LSSVM	61.3	59.7	39.1	36.6	33.2	22.9
SVM	70.8	65.1	40.8	50.6	41.6	25.5
kNN (k=5)	69.6	66.8	38.1	45.0	32.8	20.4
eSVM	**80.3**	**76.3**	43.1	49.5	41.3	23.1
EnEx	79.2	**76.3**	**44.8**	**57.0**	**47.9**	**27.4**

Each column shows the the average precision values at a particular recall threshold r, $AP@r$. Each number is actually the running average $AP@r$ (averaged over time) for a predictor that is continually updated as a new positive event is retrospectively detected. Non-parametric methods (kNN, eSVM, and EnEx) work relatively well. EnEx outperforms the other methods by a wide margin in several cases, especially when considering the AP for a lower recall threshold.

3.5 Forgetting mechanism based on clustering

The proposed training method is efficient, but can still be time-consuming if thousands of exemplars are kept as more samples are observed. Thus we introduce a forgetting mechanism based on clustering. Suppose the maximum number of exemplar predictors is p_{max}. When the number of observed exemplars exceeds this budget, each exemplar predictor $f_{\mathbf{z}}(\cdot)$ is used on all the training samples ($\mathcal{P}$ and $\mathcal{N}$) to obtain a vector of scores. And this vector is used as the representation of the exemplar $\mathbf{z}$. Then k-medoids is used to cluster the exemplars into p_{max} clusters, and only the medoids are kept as representative exemplars. This method performs clustering on the set of exemplar predictors (i.e., their classification scores) instead of the representations of the exemplars in the feature space.

4 Experiments

We evaluate the proposed framework on its ability to predict different types of target events in video. Specifically, we predict when a phone will be picked up, when a door will be opened, and when certain cooking actions are performed. Our experiments are performed on self-recorded videos and the EpicKitchen dataset [8]. Other than specifically mentioned, the hyper-parameters of the models are kept the same across all the experiments.

4.1 Experiment setting

Replayed simulation from realistic data. We aim to develop a framework that keeps observing a video stream, making predictions, and improving itself. This can be an everlasting process once the framework is deployed. However, for development and evaluation, we speed up the video stream by simulating the sequential arrival of events as follows. Given a particular category of target events, a set of prerecorded videos, and a specific lead time duration $[\tau_1, \tau_2]$, the first step is to identify where target events occurred and extract the video segments leading up to the events within the lead time range. The second step is to sample video segments which were not followed by an event of interest within the lead time range. This creates a collection of T video segments, each with a binary label for whether a target event will be seen within the lead time range after the video segment. We randomly shuffle the video segments and simulate their sequential arrival. At a certain step t, a pair of video segment and label $(\mathbf{v}_t, y_{t+\tau_2})$ is used to update the predictor. We evaluate the performance of this predictor snapshot on the future video segments $(\mathbf{v}_{t+1}, y_{t+1+\tau_2}), \ldots, (\mathbf{v}_T, y_{T+\tau_2})$. Note that the predictor does not gain any information from the evaluation process. For meaningful evaluation results, we will use all unseen video segments for evaluation and only simulate the process up until $t = 0.75T$, so that at least 25% of the data is used for evaluation.

Evaluation metric. Our goal is early prediction of target events, and we are more interested in the moments when the events do occur than the moments when the events do not occur. In general, events of interest do not occur very often, so an unintelligent predictor that never predicts the event will already achieve very high accuracy. So accuracy is not a good performance measure. A better choice for data with class imbalance would be average precision. However, as noted earlier, events might occur without any prior indication, so it is unrealistic to aim for the prediction of every event. Therefore, we propose to use the the average precision of the predictor up until certain recall value only. Let $Pr(r)$ be the precision of the predictor at recall r, the average precision until a particular recall threshold r_t ($0 < r_t \leq 1$) is calculated as $AP@r_t = \int_0^{r_t} Pr(r)dr/r_t$. Note that $AP@1$ is the ordinary average precision. Every recall threshold r_t corresponds to a score threshold s_t for the decision function, and this average precision is equivalent to calculating the average precision only with test samples that have scores higher than s_t. For a properly trained predictor, lower recall threshold should result in higher average precision.

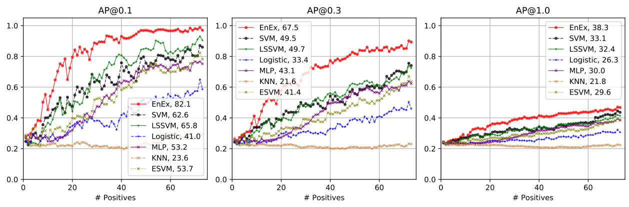
4.2 Predicting OpenDoor events

We consider the task of predicting when a door will be opened given the visual stream of a camera. The camera is mounted inside of a room facing the door. The door can be opened from the inside or outside. If the door is opened from the outside, there will be no prior visual indication. An OpenDoor action starts when the door is separated from the door frame and ends when the door is completely shut. Our goal is to predict the OpenDoor events before they start. We recorded for an extended period of time and observed 27 OpenDoor events. About half of the times the door was opened from the outside. From the remaining part of the video, we sample a negative set of video segments that did not contain an OpenDoor event. Altogether, we have a dataset of 323 video segments, among which 8.36% are positive.

Our task at time t is to predict if a OpenDoor event will occur between time $t+\tau_1$ and $t+\tau_2$. The input feature $\mathbf{v}_t$ is a 512-dim vector, which is obtained by max-pooling the ResNet-34 [19] feature vectors for all the frames in the time window $[t-l, t]$, with $l = 5$s. We experiment with different values for the lead time τ_1 of 2s and 4s. The tolerance duration $\tau_2 - \tau_1$ is always 2s. We compare with four parametric and two non parametric classifiers: multiple layer perceptron (MLP), logistic regression (LogReg), least-squared SVM (LSSVM) [42], SVM [44], k nearest neighbors (kNN, $k = 5$), and exemplar SVM (eSVM) [28]. We use an RBF kernel for SVM, LSSVM, and eSVM. The results averaged over 20 experiment runs, are shown in Table 1. Non-parametric methods work better than parametric ones in this experiment, and EnEx yields the best or close to the best result for different measurements and settings. In the supplementary video, it is clearly shown that the learned model only responds to the cases that the door is opened from the inside.

4.3 Predicting PickupPhone human actions

We consider the task of predicting when a cell phone will be picked up. A camera is mounted on top of a shelf pointing at a desk on which the phone is placed. The phone will be picked



Each curve in each subplot shows how the performance of a classifier changes as it encounters more target events for training. Each curve is the average of 10 experiment runs. The number next to each classifier in the legend box corresponds to the area under the curve.

Figure 2: Performance of different classifiers for predicting PickupPhone actions with lead time $\tau_1 = 0.5s$. Each subplot shows the Average Precision at recall threshold r ($AP@r$).

Table 2: Results on top-20 EpicKitchen actions

	positive		*AP*@0.1					*AP*@1.0				
action	%	#	**EnEx**	eSVM	SVM	LSSVM	LogReg	**EnEx**	eSVM	SVM	LSSVM	LogReg
open-cupboard	11.6	580	**30.2**	20.9	28.3	27.7	25.1	20.0	13.2	18.3	**20.4**	18.8
open-drawer	10.6	441	**32.6**	24.3	31.7	27.5	25.0	**20.4**	13.0	18.4	20.2	18.5
put-plate	8.8	377	**83.2**	32.9	79.3	76.4	51.2	31.5	15.5	33.3	**33.4**	28.6
take-plate	8.3	372	**27.9**	18.1	27.8	24.3	21.6	**16.6**	10.2	14.8	16.4	15.0
close-cupboard	9.3	362	53.0	32.5	**53.4**	46.6	50.1	27.8	14.4	27.3	27.7	**28.6**
take-spoon	7.6	345	**20.4**	14.1	14.3	15.9	13.7	**11.7**	8.9	10.2	11.4	10.8
open-fridge	7.8	344	**28.3**	13.9	19.6	21.5	20.7	**14.1**	9.2	12.6	13.8	13.4
open-tap	14.1	333	**64.4**	22.2	55.6	50.6	37.8	**30.8**	16.8	28.5	28.5	25.2
wash-hand	7.1	330	36.1	7.8	35.7	**37.0**	30.4	17.1	7.2	15.5	**17.2**	15.2
put-spoon	8.4	317	52.3	25.3	**52.8**	48.0	32.8	**21.5**	12.4	20.6	21.1	17.9
take-knife	8.1	306	**22.9**	17.5	19.2	19.3	15.3	**12.3**	9.9	11.1	12.0	11.1
put-knife	7.4	287	**49.2**	15.9	45.2	45.9	31.0	19.7	9.5	17.8	**20.3**	16.7
close-fridge	7.1	280	**68.5**	27.2	67.5	62.9	53.7	31.5	11.8	**33.1**	32.6	31.1
close-tap	10.3	267	60.1	22.3	**67.6**	59.9	48.4	33.0	14.0	**34.1**	33.6	29.8
put-pan	7.5	256	42.9	28.4	**43.0**	33.1	40.0	18.6	11.9	18.0	19.0	**20.0**
close-drawer	8.0	249	**59.9**	23.6	51.6	53.2	33.0	**25.8**	11.7	23.4	**25.8**	21.5
turnoff-tap	7.4	227	**80.5**	28.7	69.6	62.4	44.5	**31.1**	12.7	29.4	28.8	24.8
wash-plate	11.0	218	**74.6**	54.1	70.9	66.6	56.9	**38.6**	23.6	**38.6**	37.9	33.9
put-bowl	8.5	217	**47.1**	37.9	42.9	41.3	36.4	**21.2**	15.0	19.7	20.3	19.2
turnon-tap	7.4	214	**45.5**	15.5	26.3	31.9	25.0	**15.4**	9.4	12.9	14.5	13.2
average	8.8	316	**49.0**	24.2	45.1	42.6	34.6	**22.9**	12.5	21.9	22.7	20.7

% and # are the percentage and number of positive events from the data stream, respectively. Action classes are sorted by #. This table reports the running average AP@r for several types of predictors.

up when a notification pops up on the phone's screen, but it can also be picked up without any screen notification. The field of view is limited, and the phone owner is not visible most of the time, only his hand appears when the phone is being picked up. We recorded for an extended period of time to capture multiple instances of the phone pickup action. Altogether there are 105 pickup actions, roughly half of them occur without a notification on the phone's screen. From the remaining part of the video, we sample a negative set of video segments that did not contain the pickup action. Altogether, we have a dataset of 446 video segments, among which 22.53% are positive.

The 512-dim input feature $\mathbf{v}_t$ is obtained by max-pooling the ResNet-34 feature vectors for all the frames in $l = 2$s window before t. The tolerance duration $\tau_2 - \tau_1$ is always set to 2s. We compare with several types of parametric and non-parametric classifiers, as described in Section 4.2. The results for $\tau_1 = 0.5s$, averaged over 20 experiment runs, are shown in Figure 2. Results for other lead times can be found in the supplementary material. As can been seen from Figure 2, EnEx outperforms other methods by a wide margin. The gap is particularly large when when there are only 20 positive examples of the target event. The other exemplar-based classifier eSVM does not perform as well in this case, possibly due to the use of different ways for calibrating and combining the exemplar classifiers. In the supplementary material, it is illustrated that the EnEx model only responds to the events with a notification on the screen. We also perform some ablation experiments to show the effectiveness of the proposed calibration and combination methods.

4.4 Predicting human actions in kitchens

We consider the task of predicting the action of a person in a kitchen using an ego-centric camera mounted on the head. For this experiment, we use the publicly available EpicKitchen dataset [8]. Videos from this dataset are densely annotated with *verb-noun* action labels (e.g. *open-drawer*). Some sample frames can be found in the supplementary material. We do experiments on the most 20 popular action categories, listed in Table 2. The data is

Table 3: Results of AKI prediction

	Covid-19 patients			All patients		
Model	@0.1	@0.3	@1.0	@0.1	@0.3	@1.0
MLP	56.5	49.9	33.3	47.2	40.1	23.4
LogReg	50.0	43.5	28.8	41.0	36.0	22.0
LSSVM	57.8	50.1	34.0	51.1	42.1	25.9
SVM	61.4	52.9	36.3	56.6	46.7	28.6
kNN (k=5)	41.1	29.5	20.4	31.5	21.3	14.1
ESVM	46.7	43.6	30.5	30.6	29.0	19.9
EnEx	**67.5**	**58.7**	**39.3**	**60.4**	**49.0**	**29.7**

Two data pools are used: patients with Covid-19 and all patients. Each column shows the the average precision values at a particular recall threshold r, $AP@r$. Each number is actually the running average $AP@r$ (averaged over time) for a predictor that is continually updated as a new positive AKI case is observed. EnEx outperforms the other methods, especially when considering the AP for a lower recall threshold.

sampled similarly as in previous experiments. We use the I3D [6] network trained on the Kinetics 400 [23] to extract action feature at each second of the video. $\mathbf{v}_t$ is the temporally max-pooled I3D feature vectors of the $l = 5$s before t. We set $\tau_1 = 1$s, and $\tau_2 - \tau_1 = 2$s. Note that different action instances have different length, so after sampling, the most popular action might not be the one that has the highest positive rate in training samples.

Table 2 reports the running average $AP@0.1$ and $AP@1.0$ scores of different methods, averaged over 20 experiment runs. The results show that the proposed method EnEx predicts visual events with higher precision, especially at low recall threshold. For the AP at the low recall threshold $AP@0.1$, EnEx outperforms other methods on 16 out of 20 action classes, by a wide margin in many cases. For the remaining 4 action classes, the performance gap between the best method and EnEx is smaller than 1%, except for *close-tap*, where SVM obtains much higher $AP@0.1$ than EnEx. Results are more varied for $AP@1.0$. Out of the 20 actions, EnEx is the best in 12 cases (with 2 ties). For actions that EnEx is not the best, it is also not far behind from the best. On the other hand, the other methods do not perform as consistently and robustly. SVM outperforms EnEx for some actions, but on other actions (e.g. *open-fridge*, *turnoff-tap*, *turnon-tap*), its precision is much lower than EnEx. LSSVM and Logistic Regression also suffer from this problem. eSVM does not performance well on most the actions, probably due to its lack of dedicated calibration and combination methods.

4.5 Predicting AKI of hospitalized patients

EnEx is general and can be used for non-visual events as well. We consider the task of predicting if a patient will develop acute kidney injury (AKI) during hospital stay. We collected a dataset of 3455 hospitalized patients from a major US hospital (name withheld for anonymity) during the current Covid-19 pandemic. Among them, 1373 (39.7%) were Covid-19 positive, and 264 (7.6%) developed AKI before their death or hospital discharge. We aimed to build a predictive model based on a set of 95 features including demography, medical history, syndromes, and laboratory tests. We also simulated the sequential arrival of the patients, retrained prediction models whenever a positive AKI case was observed, and re-evaluated the models on the yet-to-arrive patients. Table 3 shows the results of this experiment, averaged over 20 experiment runs. We consider two pools of patients: (i) patients with Covid-19 and (ii) all patients, with or without Covid-19. EnEx outperforms the other models, and the performance between EnEx and the second best model is wider on the pool of Covid-19 patients.

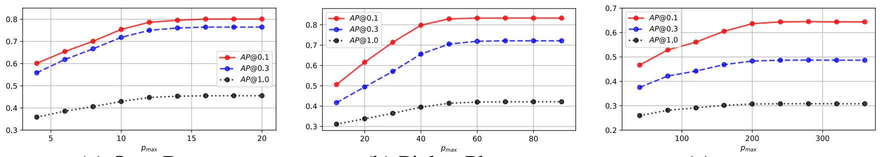
4.6 Comparison with end-to-end methods

We compare the performance of EnEx against deep network models for predicting Pickup-Phone actions with $\tau_1 = 0.5s$, OpenDoor events with $\tau_1 = 2s$, and averaged top-5 EpicKitchen actions with $\tau_1 = 1s$. The compared models are: i) a pretrained ResNet-34 [19] followed by

Table 4: Comparison of EnEx with end-to-end models

	OpenDoor 2s			PickupPhone 0.5s			Epic 1s	
model	@0.1	@0.3	@1.0	@0.1	@0.3	@1.0	@0.1	@1.0
ResNet	57.7	52.9	31.5	68.1	63.1	**38.4**	41.9	22.5
ruLSTM	61.9	51.3	26.9	76.9	52.4	33.2	40.5	**29.2**
EnEx	**79.2**	**76.3**	**44.8**	**82.1**	**67.5**	38.3	**45.4**	23.3

Each column in the table shows *AP@r*. Each number is the average of 10 experiment runs. EnEx outperforms ResNet and ruLSTM in most cases, especially for OpenDoor events where the positive samples are very sparse.



(a) OpenDoor (b) PickupPhone (c) *open-tap*

Figure 3: Performance of different exemplar budgets. Data points are averaged over 20 runs.

temporal max-pooling and fully-connected classifier (similar to the feature extractor used in Section 4.2 and Section 4.3) and ii) ruLSTM [15], the state-of-the-art model for action anticipation. The whole networks of the deep models are trained end-to-end using cross-entropy loss. The same data shuffling and evaluation methods in Section 4.1 are used. Similarly, each model is trained and evaluated for 10 runs with different random data shuffling, and the average performance is reported in Table 4. Despite being much heavier in computation, deep models still fall behind EnEx in most cases, especially for lower recall thresholds and when only very few positive samples are observed. We also observe that the performance of neural models has higher variance across different runs, implying unstable optimization.

4.7 Effect of clustering-based forgetting

We use the same setting of predicting PickupPhone human actions with $\tau_1 = 0.5s$, OpenDoor events with $\tau_1 = 2s$, and *open-tap* actions with $\tau_1 = 1s$. In each run, all the EnEx models share the same split of observed samples, but have different exemplar budgets p_{max}. The largest budget is equivalent to unlimited. We run the experiments for 20 times, and report *AP@r* in Figure 3. The results show that using about half of the observed exemplars can reduce the calculation cost, but still achieve similar prediction performance. When the number of exemplars increases to the point that computation becomes time-consuming, forgetting mechanism can be applied to trade-off between speed and precision.

5 Conclusions

We have described a non-parametric method for event prediction based on an ensemble of exemplar classifiers. Each exemplar classifier is obtained by training a classifier to separate a single pre-positive video segment from many pre-negatives. We have developed an efficient formulation for training multiple exemplar classifiers together, and we have also proposed a novel method for calibrating and combining multiple weak exemplar classifiers to create a stronger ensemble classifier. The ensemble classifier accounts for the non-predictable events, and it does not suffer from the drawbacks of many parametric classifiers. We have evaluated our method on its ability to predict visual evens (PickupPhone, OpenDoor, and cooking actions) as well as non-visual event (Acute Kidney Injury) and showed that our method outperformed the others in many cases.

References

[1] M. Akbarian, Fatemehsadat Saleh, M. Salzmann, Basura Fernando, L. Petersson, and L. Andersson. Encouraging lstms to anticipate actions very early. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 280–289, 2017.

[2] Leon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural Computation*, 4 (6):888–900, 1992.

[3] George Box and G Jenkins. Some recent advances in forecasting and control. *Applied Statistics*, 17(2):91–109, 1968.

[4] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2 edition, 2002.

[5] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4724–4733, 2017. doi: 10.1109/CVPR.2017.502. URL https://doi.org/10.1109/CVPR.2017.502.

[6] S. Y. Chen. Kalman filter for robot vision: A survey. *IEEE Transactions On Industrial Electronics*, 59(11):4409–4420, 2012.

[7] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[9] J Davis and A Tyagi. Minimal-latency human action recognition using reliable-inference. *Image and Vision Computing*, 24(5):455–472, 2006.

[10] C. Ellis, S. Masood, M. F. Tappen, J. J. LaViola, and R. Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *To appear in International Journal of Computer Vision,*, 2013.

[11] Y. A. Farha, A. Richard, and J. Gall. When will you do what? - anticipating temporal occurrences of activities. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5343–5352, 2018. doi: 10.1109/CVPR.2018.00560.

[12] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the International Conference on Computer Vision*, 2015.

[13] Andrea Frome, Yoram Singer, and Jitendra Malik. Image retrieval and classification using local distance functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[14] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[15] Antonino Furnari and Giovanni Maria Farinella. What would you expect? anticipating egocentric actions with rolling-unrolling lstms and modality attention. In *International Conference on Computer Vision (ICCV)*, 2019.

[16] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

[17] Ruohan Gao, Bo Xiong, and Kristen Grauman. Im2flow: Motion hallucination from static images for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[18] Gene H. Golub and Charles F. Van Loan. Matrix computations. 1996.

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

[20] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[21] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *International Journal of Computer Vision*, 107(2):191–202, 2014.

[22] Minh Hoai and Andrew Zisserman. Improving human action recognition using score distribution and ranking. In *Proceedings of the Asian Conference on Computer Vision*, 2014.

[23] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. URL http://arxiv.org/abs/1705.06950.

[24] Hema S Koppula and Ashutosh Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *Proceedings of the International Conference on Machine Learning*, 2013.

[25] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. In *Proceedings of International Conference on Learning and Representation*, 2018.

[26] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *Proceedings of International Conference on Learning and Representation*, 2017.

[27] Tomasz Malisiewicz and Alexei A. Efros. Recognition by association via learning per-exemplar distances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[28] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the International Conference on Computer Vision*, 2011.

[29] S.Z. Masood, C. Ellis, A. Nagaraja, and M.F. Tappen. Measuring and reducing observational latency when recognizing actions. In *Proceedings of the International Conference on Computer Vision*, 2011.

[30] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *Proceedings of International Conference on Learning and Representation*, 2016.

[31] Sebastian Nowozin and Jamie Shotton. Action points: A representation for low-latency online human action recognition. Technical Report MSR-TR-2012-68, Microsoft Research Cambridge, 2012.

[32] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 1999.

[33] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[34] MarcAurelio Ranzato, Arthur Szlam, Joan Bruna, Michael Mathieu, Ronan Collobert, and Sumit Chopra. Video (language) modeling: a baseline for generative models of natural videos. arXiv:1412.6604, 2015.

[35] Michalis Raptis and Leonid Sigal. Poselet key-framing: A model for human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[36] Fahimeh Rezazadegan, Sareh Shirazi, and Larry S. Davis. A real-time action prediction framework by encoding temporal evolution. arXiv:1709.07894, 2017.

[37] M.S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the International Conference on Computer Vision*, 2011.

[38] K. Schindler and Luc Van Gool. Action snippets: How many frames does human action recognition require? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[39] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research*, 22(2):99–116, 2003.

[40] Yuge Shi, Basura Fernando, and Richard Hartley. Action anticipation with rbf kernelized feature mapping rnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[41] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *Proceedings of the International Conference on Machine Learning*, 2015.

[42] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. DeMoor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

[43] Sam Toyer, Anoop Cherian, Tengda Han, and Stephen Gould. Human pose forecasting via deep markov models. In *International Conference on Digital Image Computing: Techniques and Applications*, 2017.

[44] Vladimir Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

[45] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[46] J. Walker, A. Gupta, and M. Hebert. Patch to the future. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[47] Jacob Walker, Abhinav Gupta, and Martial Hebert. Dense optical flow prediction from a static image. In *Proceedings of the International Conference on Computer Vision*, 2015.

[48] Boyu Wang and Minh Hoai. Back to the beginning: Starting point detection for early recognition of ongoing human actions. In *Computer Vision and Image Understanding*, volume 175, pages 24–31, 2018.

[49] Boyu Wang and Minh Hoai. Predicting body movement and recognizing actions: an integrated framework for mutual benefits. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, 2018.

[50] Boyu Wang, Lihan Huang, and Minh Hoai. Active vision for early recognition of human actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[51] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Advances in Neural Information Processing Systems*. 2017.

[52] Zijun Wei and Minh Hoai. Region ranking SVMs for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[53] Peter Whittle. *Hypothesis Testing in Time Series Analysis*. Almquist and Wicksell, 1951.

[54] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.

[55] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[56] Ronald R. Yager and Dimitar P. Filev. Induced ordered weighted averaging operators. *IEEE Transactions on Systems, Man and Cybernetics*, 29(2):141–150, 1999.

[57] Bangpeng Yao and Li Fei-Fei. Action recognition with exemplar based 2.5D graph matching. In *Proceedings of the European Conference on Computer Vision*, 2012.

[58] Mihai Zanfir, Marius Leordeanu, and Cristian Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *Proceedings of the International Conference on Computer Vision*, 2013.

[59] Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.