Fast Video Classification via Adaptive Cascading of Deep Models

Haichen Shen, Seungyeop Han*, Matthai Philipose†, Arvind Krishnamurthy
University of Washington, Rubrik Inc.*, Microsoft Research†
{haichen, arvind}@cs.washington.edu, seungyeop.han@rubrik.com, matthaip@microsoft.com

Abstract

Recent advances have enabled “oracle” classifiers that can classify across many classes and input distributions with high accuracy without retraining. However, these classifiers are relatively heavyweight, so that applying them to classify video is costly. We show that day-to-day video exhibits highly skewed class distributions over the short term, and that these distributions can be classified by much simpler models. We formulate the problem of detecting the short-term skews online and exploiting models based on it as a new sequential decision making problem dubbed the Online Bandit Problem, and present a new algorithm to solve it. When applied to recognizing faces in TV shows and movies, we realize end-to-end classification speedups of 2.5-8.5×/2.8-12.7× (on GPU/CPU) relative to a state-of-the-art convolutional neural network, at competitive accuracy.

1. Introduction

Consider recognizing entities such as objects, people, scenes and activities in every frame of video footage of day-to-day life. Such footage may come, for instance, from the media, wearable cameras, movies, or surveillance cameras. In principle, these entities could be drawn from thousands of classes: many of us encounter hundreds to thousands of distinct people, objects, scenes and activities through our life. Over short intervals such as minutes, however, we tend to encounter a very small subset of classes of entities. For instance, a wearable camera may see the same set of objects from our desk at work for an hour, a movie may focus only on cooking-related activities through a five-minute kitchen sequence, and media footage of an event may focus on only those celebrities participating in the event. In this paper, we characterize and exploit such short-term class skew to significantly reduce the latency of classifying video using Convolutional Neural Networks (CNNs).

Since the seminal work of Viola and Jones [23] on face detection, one of the best-known techniques to speed up classification has been to structure the classifier as a cascade (or tree [26]) of simple classifiers such that “easy” examples lead to early exits and are therefore classified faster. Cascaded classifiers require that training and test data are strongly (and identically) biased toward a small number of easy to detect classes. In the (binary) face detection task, for example, the class “not a face” is both (i) by far more common than “face”, and (ii) often quite easy to classify via a small number of comparisons of inexpensive Haar-style features. In fact, traditional cascades are most applicable in detection tasks [4], where the background is both much more common and easier to classify than the foreground.

Distinct from the (two-class) detection setting in traditional cascading, recent advances in convolutional neural networks (CNNs) [16, 21, 29] have opened up the possibility of using a single, pre-trained “oracle” classifier to recognize thousands of classes such as people, objects and scenes. When training such oracle classifiers, such as GoogLeNet [21] of VGGFace [16], a small number of classes do not usually dominate the training set: for broad applicability, the classifier is trained assuming that all classes are more or less equally likely. Even if such a skew toward such small classes existed, there is no a priori reason that these dominant classes are fast to classify. It may seem therefore that cascading is not a promising optimization for improving the speed of entity recognition in video via CNNs.

We demonstrate, however, that for many recognition tasks, day-to-day video often exhibits significant short-term skews in class distribution. We present measurements on a diverse set of videos that show, for instance, that in over 90% of 1-minute windows, at least 90% of objects interacted with by humans belong to a set of 25 or fewer objects. The underlying ImageNet-based recognizer, on the other hand, can recognize up to 1000 objects. We show that similar skews hold for faces and scenes in videos.

Even if such skew exists, to our knowledge, it has not been shown that distributions skewed toward small sets of classes can be classified accurately by simpler
CNNs than uniformly distributed ones. We therefore also demonstrate that when class distribution is highly skewed, “specialized” CNNs trained to classify inputs from this distribution can be much more compact than the oracle classifier. For instance, we present a CNN that executes 60× fewer FLOPs than the state-of-the-art VGGFace [16] model, but has comparable accuracy when over 50% of faces come from the same 10 or fewer people. We present similar order-of-magnitude faster specialized CNNs for object and scene recognition.

Given the ability to produce fast, accurate versions of CNNs specialized for particular test-time skews, we seek to estimate the (possibly non-stationary) skew at test-time, produce a specialized model if appropriate, exploit the model as long as the skew lasts, detect when the skew disappears and then revert to the oracle model. As with standard “bandit”-style sequential decision-making problems, the challenge is in balancing exploration (i.e., using the expensive oracle to estimate the skew) with exploitation (i.e., using a model specialized to the current best available estimate of the skew). We formalize this problem as the Oracle Bandit Problem and propose a new exploration/exploitation-based algorithm we dub Windowed-ε-Greedy (WEG) to address it.

Using a combination of synthetic data and real-world videos, we empirically validate the WEG algorithm. In particular, we show that WEG can reduce the end-to-end classification overhead of face recognition on TV episodes and movies by 2.5-8.5× relative to unspecialized classification using the VGGFace classifier on a GPU (2.8-12.7× on a CPU). We show via synthetic data that similar gains are to be had on object and scene recognition as well. We provide a detailed analysis of WEG’s functioning, including an accounting of how much its key features contribute. To our knowledge our system is the first to use test-time sequential class skews in video to produce faster classifiers.

2. Related work

There is a long line of work on cost-sensitive classification, the epitome of which is perhaps the cascaded classification work of Viola and Jones [23]. The essence of this line of work [25,27] is to treat classification as a sequential process that may exit early if it is confident in its inference, typically by learning sequences that have low cost in expectation over training data. Recent work [15] has even proposed cascading CNNs as we do. All these techniques assume that testing data is i.i.d. (i.e., not sequential), that all training happens before any testing, and rely on skews in training data to capture cost structure. As such, they are not equipped to exploit short-term class skews in test data.

Traditional sequential models such as probabilistic models [3, 17, 24] and Recurrent Neural Networks (RNNs) [5, 11] are aimed at classifying instances that are not independent of each other. Given labeled sequences as training data, these techniques learn more accurate classifiers than those that treat sequence elements as independent. However, to our knowledge, none of these approaches produces classifiers that yield less expensive classification in response to favorable inputs, as we do.

Similar to adaptive cascading, online learning methods [9, 12, 22] customize models at test time. For training, they use labeled data from a sequential stream that typically contains both labeled and unlabeled data. As with adaptive cascading, the test-time cost of incrementally training the model in these systems needs to be low. A fundamental difference in our work is that we make no assumption that our input stream is partly labeled. Instead, we assume the availability of a large, resource-hungry model that we seek to “compress” into a resource-light cascade stage.

Estimating distributions in sequential data and exploiting it is the focus of the multi-armed bandit (MAB) community [1, 13]. The Oracle Bandit Problem (OBP) we define differs from the classic MAB setting in that in MAB the set of arms over which exploration and exploitation happen are the same, whereas in OBP only the oracle “arm” allows exploration whereas specialized models allow exploitation. Capturing the connection between these arms is the heart of the OBP formulation. Our Windowed-ε-Greedy algorithm is strongly informed by the use of windows in [6] to handle non-stationarities and the well-known [20] ε-greedy scheme to balance exploration and exploitation.

Finally, much recent work has focused on reducing the resource consumption of (convolutional) neural networks [2, 7, 8, 18]. These techniques are oblivious to test-time data skew and are complementary to specialization. We expect that even more pared-down versions of these optimized models will provide good accuracy when specialized at test-time.

3. Class skew in day-to-day video

Specialization depends on skew (or bias) in the temporal distribution of classes presented to the classifier. In this section, we analyze the skew in videos of day-to-day life culled from YouTube. We assembled a set of 30 videos of length 3 minutes to 20 minutes from five classes of daily activities: socializing, home repair, biking around urban areas, cooking, and home tours. We expect this kind of footage to come from a variety of sources such as movies, amateur productions of the kind that dominate YouTube and wearable videos.

We sample one in three frames uniformly from these videos and apply state-of-the-art face (derived from
In order to exploit skews in the input, we cascade the expensive but comprehensive oracle model with a (hopefully much) less expensive “compact” model. This cascaded classifier is designed so that if its input belongs to the frequent classes in the incoming distribution it will return early with the classification result of compact model, else it will invoke the oracle model. Thus if the skew dictates that $n$ frequent classes, or dominant classes, comprise percentage $p$ of the input, or skew, model execution will cost the overhead of just executing compact model roughly $p\%$ of the time, and the overhead of executing compact model and oracle sequentially the rest of the time. When $p$ is large, the lower cost compact model will be incurred with high probability.

To be more concrete, we use state of the art convolutional neural networks (CNNs) for oracles. In particular, we use the GoogLeNet [21] as our oracle model, for object recognition; the VGG Net 16-layer version for scene recognition [29]; and the VGGFace network [16] for face recognition. The compact models are also CNNs. For these, we use architectures derived from the corresponding oracles by systematically (but manually) removing layers, decreasing kernel sizes, increasing kernel strides, and reducing the size of fully-connected layers. The

| Task   | Model       | Acc. (%) | FLOPs | CPU lat.(ms) | GPU lat.(ms) |
|--------|-------------|----------|-------|--------------|--------------|
| Object | O1          | 68.9     | 3.17G | 779.3        | 11.0         |
| classes | O2         | 48.9     | 0.82G | 218.2        | 6.13 (×2.5)  |
| Scene  | S1          | 58.1     | 30.9G | 2570         | 28.8         |
|        | S2          | 48.9     | 0.55G | 152.2        | 3.36 (×8.6)  |
| Face   | F1          | 95.8     | 30.9G | 2576         | 28.8         |
|        | F2          | 84.8     | 0.60G | 90.1 (×28.6) | 2.48 (×11.6) |

Table 1: Oracle classifiers versus compact classifiers in top-1 accuracy, number of FLOPs, and execution time. Execution time is feedforward time of a single image without batching on Caffe [10], a Linux server with a 24-core Intel Xeon E5-2620 and an NVIDIA K20c GPU.

and (d); e.g. the cyan line in (d) dominates that in (b). We expect that if we ran a hand-detector and only recognized objects in the hand (analogously to recognizing detected faces), the skew would be much sharper.

Specialized models must exploit skews such as these to deliver appreciable speedups over the oracle. Typically, they should be generated in much less than a minute, handle varying amounts of skew gracefully, and deliver substantial speedups when inputs belong to subsets of 20 classes or fewer out of a possible several hundred in the oracle.

4. Specializing Models

Figure 1 shows the results for object recognition and scene recognition. We partition the sequence of frames into segments of length $\tau$ and show one plot per segment length. Each line in the plot corresponds to percentage skew $s \in \{60, 70, 80, 90\}$. Each line in the plots shows the cumulative distribution representing the fraction of all segments where $n$ labels comprised more than $s$ percent of all labels in the segment. For instance, for 10-second segments (Figure 1(a)), typically roughly 100 frames, 5 objects comprised 90% of all objects in a segment 60% of the time (cyan line), whereas they comprise 60% of objects 90% of the time (dark blue).

In practice, detecting skews and training models to exploit them within 10 seconds is often challenging. As figures (b) and (c) show, the skew is less pronounced albeit still very significant for longer segments. For instance, in 90% of 3-minute segments, the top 15 objects comprise 90% of objects seen. The trend is similar with faces and scenes, with the skew significantly more pronounced, as is apparent from comparing figures (b) and (d); e.g. the cyan line in (d) dominates that in (b).

Figure 1: Temporal skew of classes in day-to-day video.

[16], scene [29] and object recognizers [19] to every sampled frame. Note that these “oracle” recognizers can recognize up to 2622 faces, 205 scenes and 1000 objects respectively. For face recognition, we record the top-scoring label for each face detected, and for the others, we record only the top-scoring class on each frame. For object recognition in particular, this substantially undercounts objects in the scene; our count (and specialization) applies to applications that identify some instances, in 90% of 3-minute segments, the top 15 objects comprise 90% of objects seen. The trend is similar albeit still very significant for longer segments. For instance, in 90% of 3-minute segments, the top 15 objects comprise 60% of objects 90% of the time (dark blue).

In practice, detecting skews and training models to exploit them within 10 seconds is often challenging. As figures (b) and (c) show, the skew is less pronounced albeit still very significant for longer segments. For instance, in 90% of 3-minute segments, the top 15 objects comprise 90% of objects seen. The trend is similar with faces and scenes, with the skew significantly more pronounced, as is apparent from comparing figures (b) and (d), comprise percentage skew $s \in \{60, 70, 80, 90\}$. Each line in the plots shows the cumulative distribution representing the fraction of all segments where $n$ labels comprised more than $s$ percent of all labels in the segment. For instance, for 10-second segments (Figure 1(a)), typically roughly 100 frames, 5 objects comprised 90% of all objects in a segment 60% of the time (cyan line), whereas they comprise 60% of objects 90% of the time (dark blue).

In practice, detecting skews and training models to exploit them within 10 seconds is often challenging. As figures (b) and (c) show, the skew is less pronounced albeit still very significant for longer segments. For instance, in 90% of 3-minute segments, the top 15 objects comprise 90% of objects seen. The trend is similar with faces and scenes, with the skew significantly more pronounced, as is apparent from comparing figures (b) and (d); e.g. the cyan line in (d) dominates that in (b).

4. Specializing Models

In order to exploit skews in the input, we cascade the expensive but comprehensive oracle model with a (hopefully much) less expensive “compact” model. This cascaded classifier is designed so that if its input belongs to the frequent classes in the incoming distribution it will return early with the classification result of compact model, else it will invoke the oracle model. Thus if the skew dictates that $n$ frequent classes, or dominant classes, comprise percentage $p$ of the input, or skew, model execution will cost the overhead of just executing compact model roughly $p\%$ of the time, and the overhead of executing compact model and oracle sequentially the rest of the time. When $p$ is large, the lower cost compact model will be incurred with high probability.

To be more concrete, we use state of the art convolutional neural networks (CNNs) for oracles. In particular, we use the GoogLeNet [21] as our oracle model, for object recognition; the VGG Net 16-layer version for scene recognition [29]; and the VGGFace network [16] for face recognition. The compact models are also CNNs. For these, we use architectures derived from the corresponding oracles by systematically (but manually) removing layers, decreasing kernel sizes, increasing kernel strides, and reducing the size of fully-connected layers. The
end results are architectures (O[1|2] for objects, S[1|2] for scenes and F[1|2] for faces) that use noticeably less resources (Table 1), but also yield significantly lower average accuracy when trained and validated on unskewed data, i.e., the standard training and validation sets. For instance, O1 requires roughly $4 \times$ fewer FLOPs to execute than VGGFace, but achieves roughly 70% of its accuracy.

However, in our approach, we train these compact models to classify skewed distributions observed during execution, denoted by specialized classifier, and their performance on skewed distributions is the critical measure. In particular, to generate a specialized model, we create a new training dataset with the data from the $n$ dominant classes of the original data, and a randomly chosen subset from the remaining classes with label “other” such that the dominant classes comprise $p$ percent of the new data set. We train the compact architecture with this new dataset.

Figure 2 shows how compact models trained on skewed data and cascaded with their oracles perform on validation data of different skews. Figure 2(a) analyzes the case where $n = 10$, for various combinations of training and validation skews for model O1. Recall from Table 1 that O1 delivers only 70% of its accuracy on unskewed inputs. However, when training and testing is on skewed inputs, the numbers are much more favorable. When O1 is trained on $p=90\%$ skewed data with $n=10$ dominant classes, it delivers over 84% accuracy on average (the left-most dark-blue bar). This is significantly higher than the oracle’s average of 68.9% (top-1 accuracy), denoted by the horizontal black line. We also observed from Figure 2(a) that when O1 is trained on 60% skewed data, the cascaded classifier maintains high accuracy across a wide range of testing skews from 90% to 50%. Therefore, in what follows, we use 60% skew as fixed training skew to specialize object compact models in the rest of paper (similarly 70% fixed skew for scene and 50% for face). Figure 2(b) shows that, where $n$ is varied for O1, the cascaded classifier degrades very gracefully with $n$. Finally, Figure 3, which reports similar measurements on compact models S[1|2] and F[1|2] shows that these trends carry over to scene and face recognition.

Finally, we note that since skews are only evident at test-time, specialized models must be trained extremely fast (ideally a few seconds at most). We use two techniques to accomplish this. First, before we begin processing any inputs, we train all model architectures on the full, unskewed datasets of their oracles. At test time, when the skew $n,p$ and the identity of dominant classes is available, we only retrain the top (fully connected and softmax) layers of the compact model. The lower layers, being “feature calculation” layers do not need to change with skew. Second, as a pre-processing step, we run all inputs in the training dataset through the lower feature-calculation layers, so that when re-training the top layers at test time, we can
 avoid doing so. This combination of techniques allows us to re-train the specialized model in roughly 4s for F1 and F2 and 14s for O1/O2, many orders of magnitude faster than fully re-training these models.

5. Sequential Model Specialization

5.1. The Oracle Bandit Problem (OBP)

Let \( x_1, x_2, \ldots, x_t, \ldots \in X = \mathbb{R}^n \) be a stream of images to be classified. Let \( y_1, y_2, \ldots, y_t, \ldots \in Y = [1, \ldots, k] \) be the corresponding classification results. Let \( \pi : \mathbb{I}^+ \to \mathbb{I}^+ \) be a partition over the values. Associate the distribution \( T_j \) with partition \( j \), so that each pair \((x_t, y_t)\) is sampled from \( T_{\pi(i)} \). Intuitively, \( \pi \) partitions, or segments, \( x_t, x_{t+1}, \ldots \) into a sequence of “epochs”, where elements from each epoch \( j \) are drawn independently from the corresponding stationary distribution \( T_j \). Thus, for the overall series, samples are drawn from an abruptly-changing, piece-wise stationary distribution. At test time, neither results \( y_t \) nor partitions \( \pi \) are known.

Let \( h^* : X \to Y \) be a classifier, designated the “oracle” classifier, trained on distribution \( T^* \). Intuitively \( T^* \) is a mixture of all distributions comprising the oracle’s input stream: \( T^* = \sum_j T_j \). Let \( R(h^*) \) be the cost (e.g., some combination of execution cycles and accuracy), assumed invariant across \( x \), needed to execute \( h^* \) on any \( x \in X \). Executing \( h^* \) incurs a cost of \( R(h^*) \) with probability 1. At test time, on each input \( x_t \), we can always consult \( h^* \) at cost \( R(h^*) \) to get a label \( y_t \), with some (high) accuracy \( a^* \).

Let the dominant classes \( D_{T_p} \subseteq Y \) of distribution \( T \) be the distinct set of classes in the top \( p \) fraction of \( T \)’s cumulative distribution. For each set of dominant classes \( D_{T_p} \), the corresponding specialized classifier \( h_{D_{T_p}} \) is trained on a dataset that draws fraction \( p \) of \( y \)’s examples from classes in \( D_{T_p} \) and the rest from \( Y - D_{T_p} \) with a common label “other”. On input \( x \in X \), \( h_{D_{T_p}} \) is converted into a cascaded classifier \( \hat{h}_{D_{T_p}}(x) \in D_{T_p} \), return \( y \), otherwise return \( h^*(x) \).

Given examples drawn from some distribution \( T' \) (which may be the same as \( T \)), each classifier \( \hat{h}_{D_{T_p}} \) has a cost distribution \( R_{T'}(\hat{h}_{D_{T_p}}) \) attached to it. Let \( \mu(R) \) be the mean of distribution \( R \). Also suppose \( \|T', T\| \) is the directional similarity (e.g., symmetric Kullbach-Leibler divergence) between \( T' \) and \( T \). We require that \( R \) satisfies the following distributional monotonicity (DM) property. For any two distributions \( T' \) and \( T'' \), if \( \|T', T\| = \|T'', T\| \), \( \mu(R_{T'}(h_{D_{T_p}})) \leq \mu(R_{T''}(h_{D_{T_p}})) \) for any \( h_{D_{T_p}} \). Further, we require that the accuracy of \( h_{D_{T_p}} \) satisfy the DM property. Finally, the first time \( \hat{h}_{D_{T_p}} \) is used, we charge a flat (and usually high) “training” cost for \( R_0 \).

Intuitively, the DM property ensures that the more similar the current distribution of data is to the distribution that a classifier was specialized on, the less need for the classifier to “cascade” to the oracle (giving a lower average cost of classification), and the more accurate the classifier as a whole.

Now consider a policy \( P \) that, for each incoming image \( x_t \), selects a classifier \( \hat{h}_{D_{T_p}} \) (for some set choice of \( D_{T_p} \)), and applies it to \( x_t \). The classifier selected could also include the oracle. The expected total cost of this policy \( R_P = \sum_i \{\hat{h}_{D_{T_p}}\} R_0 + \Sigma_i \mu(R_{T_{\pi(i)}}(\hat{h}_{D_{T_p}})) \). We seek a policy \( P^* \) that minimizes this cost: \( P^* = \arg \min_P R_P \). Note that since the cost is typically a combination of accuracy and resource use (e.g., latency), reducing cost can optimize along both these fronts.

5.2. The Windowed \( \varepsilon \)-Greedy (WEG) Algorithm

The DM property is at the heart of minimizing cost \( R_T(\hat{h}_{D_{T_p}}) \). Essentially, if we can empirically estimate the current distribution \( T' \) and ensure that the classifier \( \hat{h}_{D_{T_p}} \) currently in use is based on distribution \( T \) that is not “too different from” \( T' \), then classification costs will be low. Note that the (high) one-time cost \( R_0 \) of specialization forbids us from training a new \( \hat{h}_{D_{T_p}} \) every time \( T' \) changes.

The above suggests that a suitable policy might alternate between exploring to estimate the current distribution and exploiting a specialized classifier \( \hat{h}_D \) when available. Unlike standard “bandit” settings [14], where the actions for exploration and exploitation belong to the same set, so that exploration would involve trying out random other specialized classifiers, in our “oracle bandit” setting other random specialized classifiers are likely to have both low accuracy and high execution cost. We therefore always explore by consulting oracle \( h^* \), which will give a good empirical estimate of the current distribution albeit at high cost.

Another way in which the OBP differs from the standard bandit setting is that the underlying distribution is (piece-wise) non-stationary. When empirically estimating the current distribution therefore, we need a notion of the current epoch (or “piece” of distribution). Borrowing from the literature on non-stationary bandits [6], we therefore maintain a window \( S_t \) of samples initialized when we determine a new epoch \( t \) has started.

Algorithm 1 details our solution, the Windowed \( \varepsilon \)-Greedy (WEG) algorithm. WEG uses heuristics to switch between exploration and exploitation and doesn’t guarantee to find the optimal policy. While processing the incoming stream \( x_t \), WEG alternates between three phases. In the window initialization phase, it uses oracle \( h^* \) to accumulate a minimum number \( w_{min} \) of samples of the current epoch \( j \). If these samples are distributionally close enough to those from the previous epoch,
we infer the previous epoch is continuing and merge the corresponding sample sets (Line 6). In the next (unspecialized) classification phase, WEG estimates whether a cascaded model $h_D$ based on the dominant classes $D$ in the current epoch will be more accurate than $h^*$ (Line 11). If so, it trains $h_D$ and transitions to specialized classification; if not, it uses the oracle.

The specialized classification phase simply applies the current cascaded model $h_D$ to inputs (Line 18) until it determines that the distribution it was trained on (as represented by $D$) does not adequately match the current distribution. This determination is non-trivial because in the specialization phase, we wish to avoid consulting the oracle in order to reduce costs. However, the oracle is the only unbiased source of samples from the current distribution. We therefore use two proxy measures of increasing expense (Line 23).

First, we use the fraction of times $\hat{h}_D$ avoided using the oracle in the last $w$ classifications as empirical skew $\hat{p}$ and estimate the true current skew $p^*$ that triggered entry to the phase. However, this is often too conservative.

As we observed in Section 4, cascaded classifiers can maintain high accuracy across a wide range of skews. We therefore re-estimate accuracy based on the current skew $p$, thus avoiding a premature exit from specialized classification and subsequent excess specialization. Second, our accuracy estimate is prone to failure in the case where $h_D$ routinely confuses elements of the dominant class $D$. To handle this case, we have no option but to consult $h^*$. We do so with a small probability $\epsilon$ (Line 19) and exit if the oracle confirms that dominant-class confusion is too frequent.

Because both oracle and specialized classifier make mistakes in classification, the empirical skew $\hat{p}$ observed at runtime can differ from the true skew $p$ of the distribution. Thus, we adopt a simple model to estimate the true skew from empirical skew. Suppose $N$ is total number of classes that the oracle is trained to classify, $n$ is dominant classes size $|D|$, and $a^*$ is accuracy of oracle. Assuming the confusion matrix is uniform, empirical skew $\hat{p}$ when using oracle classifier is:

$$\hat{p} = p \cdot a^* + (1 - a^*) \frac{n - 1}{N - 1} + (1 - p)(1 - a^*) \frac{n}{N - 1}$$

(1)

The first term handles the case that the input belongs to $D$ and is correctly classified. The second term handles the case that the input belongs to $D$ but is confused with the other classes in $D$. The third term assumes the input does not belong to $D$ but is confused with classes in $D$. Given that we know $\hat{p}$, we can derive the true skew $p$ by solving Equation 1. Similarly we can also estimate true skew $p$ when using cascaded classifier.

Another key technique in Algorithm 1 is to estimate the accuracy of a cascaded classifier. Suppose we already know the true skew $p$, the accuracy of cascaded classifier $\hat{h}_D$ can be estimated by:

$$a_{\hat{h}_D} = p \cdot a_{in} + p \cdot e_{in→out} \cdot a^* + (1 - p) \cdot a_{out} \cdot a^*$$

(2)

where $a_{in}$ is the accuracy of specialized classifier $h_D$ on $n$ dominant classes, $e_{in→out}$ is the fraction of dominant inputs that $h_D$ classifies as non-dominant ones, and $a_{out}$ is the fraction of non-dominant inputs that $h_D$ classifies as non-dominant (note that these inputs will be cascaded to the oracle). We have observed that these parameters $a_{in}$, $e_{in→out}$, $a_{out}$ of specialized classifier $h_D$ are mainly affected only by the size of the dominant class $D$, not the identity of elements in it. Thus, we pre-compute these parameters for a fixed set of values of $n$ (averaging over 10 samples of $D$ for each $n$), and use linear interpolation for other $n$s at test time.
Table 2: Average accuracy and GPU latency of recognition over segments. For the segment column, each parenthesis indicates a segment of 5 minutes with the number of dominant classes and the skew.

| Segments | Object | Scene | Face |
|----------|--------|-------|------|
|          | oracle | WEG   | oracle | WEG   | oracle | WEG   |
| (n=5, p=.8) | 69.5   | 11.6  | 74.9  | 6.3   | 57.6   | 28.9  |
| (n=10, p=.8) | 66.7   | 11.6  | 70.9  | 7.8   | 57.2   | 28.9  |
| (n=15, p=.9) | 68.7   | 11.6  | 70.0  | 8.0   | 57.8   | 28.9  |

| Random |          |       |       |
|--------|----------|-------|-------|
| (n=5, p=.8) | 68.1   | 11.9  | 68.1  | 11.8  |
| (n=10, p=.9) | 67.9   | 11.7  | 71.0  | 9.4   |
| (n=15, p=.9) | 70.6   | 11.6  | 74.3  | 7.3   |

Table 3: Accuracy and average processing latency per frame on videos with oracle vs. WEG (latencies are shown in ms). For additional insight, the last 5 columns show key statistics from WEG usage.

6. Evaluation

We implemented the WEG algorithm with a classification runtime based on Caffe [10]. The system can be fed with videos to produce classification results by recognizing frames. Our goal was to measure both how well the large specialized model speedups of Table 1 translated to speedups in diverse settings and on long, real videos. Further we wished to characterize the extent to which elements of our design contributed to these speedups.

6.1. Synthetic experiments

First, we evaluate our system with synthetically generate data in order to study diverse settings. For this experiment, we generate a time-series of images picked from standard large validation sets of CNNs we use. Each test set comprises of one or two segments where a segment is defined by the number of dominant classes, the skew, and the duration in minutes. For each segment, we assume that images appear at a fixed interval (1/6 seconds) and that each image is picked from the testing set based on the skew of the segment. For an example of a segment with 5 dominant classes and 90% skew, we pre-select 5 classes as dominant classes and pick an image with 90% probability from the dominant classes and an image with 10% probability from the other classes at each time of image arrival over 5 minutes duration. Images in a class are picked in a uniform random way. We also generate traces with two consecutive segments with different configurations to study the effect of moving from one context to the other.

The following points are worth noting. (i) (Row 1 and it’s sub-rows) WEG is able to detect and exploit skews over 5-minute intervals and get significant speedups over the oracle while preserving accuracy. For the single segment cases, the GPU latency speedup per-image was 1.5× to 2.0×, 1.5× to 2.3×, and 3.3× to 5.9×, for object, scene, and face, respectively. However, due to WEG’s overhead these numbers are noticeably lower than the raw speedups of specialized models (Table 1). When the number of dominant classes increase, the specialist latency increases because it alternates between exploration and exploitation to recognize more dominant classes (compare row 1/2 or 3/4 for object and scene recognition, and row 2/3 for face recognition). The latency also increases when the skew of dominant classes decreases because specialist cascades more times to oracle model when using the cascaded classifier (compare row 2/3 for object and scene recognition, and row 2/3 for face recognition). (ii) (Row 2) WEG is quite stable in handling random inputs, essentially resorting to the oracle so that accuracy and latency are unchanged. (iii) (Rows 3 and 4) WEG is able to detect abrupt input distribution changes as the accuracy remains comparable.
We investigated this and found that the specialized com-
WEG on these faces and measured the total execution.
We now turn to evaluating WEG on real videos. We
2.8
12.7× (CPU) and 2.5×-8.5× (GPU).
To understand the speedup, we summarize the statis-
tics of WEG execution in Table 3. “Special rate” indi-
cates the percentage of time that specializer exploits
the cascaded classifier to reduce the latency, while cas-
cade rate reveals the percentage of time that a cascaded
classifier cascades to the oracle classifier, thus hurting
performance. Higher special rate and lower cascade
rate yield more speedup. The cascade rate of “Ocean’s
Eleven” is significantly higher than that of other videos.
We investigated this and found that the specialized com-
 pact CNN repeatedly made mistakes on one person in
the video, which led to a high cascade rate. “Trans. spe-
cial” counts the number of times WEG needed to switch
between specialized and unspecialized classification to
handle the distribution changes and insufficient explo-
ration. The average dominant classes sizes (“dom. size”) show that the real videos are skewed to fewer dominant
classes than the configurations used in the synthetic
experiment. This explains why our system achieved
higher speedup on real videos than on synthetic data.
Overall, the statistics show that the dataset exercise
WEG features such as skew estimation, cascading and
specialization.
To understand better the utility of WEG’s features,
we performed an ablation study: (a) We disable the
adaptive window exploration (Line 5-6 in Algorithm 1),
and use a fixed window size of 30 and 60. (b) We use the
skew of dominant classes in the input distribution as the
training skew for specializing compact CNNs instead
of using the fixed (50%) training skew suggested in
Section 4. (c) We apply a simple (but natural) criterion
to exit from the specialized classification phase: WEG
now exits when the current skew is lower than the skew
when it entered into specialized classification phase
instead of using the estimated accuracy as soft indicator.
Figure 4 shows the comparison between these vari-
ants and our specialist in accuracy and CPU / GPU
speedups when recognizing faces on Friends video. In
the figure we show the absolute differences in accuracy
and relative differences in CPU / GPU speedup. (a)
Fixed window size (30 and 60) variants achieve similar
accuracy but lower speedup. As table 3 (“window size”
column) shows, the adaptively estimated size for the
window is between 30 and 60. In general, too small
a window fails to capture the full dominant classes,
yielding specializers that exit prematurely. Too large a
window requires more work by the oracle to fill up the
window. (b) Using variable rather than fixed skew for
training achieves more speedup, but suffers from 30%
loss in accuracy. This is because the training skew is
usually very high. As discussed in Section 4, training
on highly skewed data produces models vulnerable to
false positives in “other” classes. (c) The simple exit
variant achieves almost comparable accuracy while the
latency is more than 50% higher than our system. It
demonstrates the value of our accuracy estimate in mod-
eling the accuracy of cascaded classifiers and to prevent
premature exit from the specialized classification phase.
In summary, the key design elements of WEG each have
a role in producing fast and accurate results.

7. Conclusion
We characterize skew in day-to-day video, showed that
skewed distributions need much simpler CNNs, and
developed new very fast specialized CNNs. We for-
malize the “bandit”-style sequential model selection
as Oracle Bandit Problem and provide a new ex-
ploration/exploitation based algorithm, Windowed e-
Greedy (WEG). Our solution speeds up face recognition
on TV episodes and movies by 2.5-8.5× on a GPU (2.8-
12.7× on a CPU) with little loss in accuracy relative to a modern convolutional neural network.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3), May 2002. 2

[2] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014. 2

[3] C. Desai and D. Ramanan. Detecting actions, poses, and objects with relational phraselets. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ECCV’12, 2012. 2

[4] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *PAMI*, 2014. 1

[5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 2

[6] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory (ALT)*, 2011. 2, 5

[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 2

[8] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Proceedings of the Twenty-Third Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1135–1143. 2015. 2

[9] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *2011 International Conference on Computer Vision*, 2011. 2

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia (MM)*, 2014. 3, 7

[11] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[12] B. Kveton and M. Valko. Learning from a single labeled face and a stream of unlabeled data. In *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, 2013. 2

[13] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1), Mar. 1985. 2

[14] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. 5

[15] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 2

[16] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015. 1, 2, 3

[17] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012. 2

[18] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. *arXiv:1603.05279*, Mar. 2016. 2

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 3

[20] R. Sutton and A. Barto. *Reinforcement Learning, an introduction*. MIT Press/Bradford Books, 1998. 2

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 1, 3

[22] M. Valko, B. Kveton, L. Huang, and D. Ting. Online semi-supervised learning on quantized graphs. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 606–614, Corvallis, Oregon, 2010. AUAI Press. 2

[23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 1, 2

[24] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007. 2

[25] Z. Xu, M. Kusner, M. Chen, and K. Q. Weinberger. Cost-sensitive tree of classifiers. In *ICML*, 2013. 2

[26] Z. E. Xu, M. J. Kusner, K. Q. Weinberger, M. Chen, and O. Chapelle. Classifier cascades and trees for minimizing feature evaluation cost. *Journal of Machine Learning Research*, 15:2113–2144, 2014. 1

[27] Q. Yang, C. X. Ling, X. Chai, and R. Pan. Test-cost sensitive classification on data with missing values. *IEEE Trans. Knowl. Data Eng.*, 18(5):626–638, 2006. 2

[28] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016. 3

[29] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *Proceedings of the Twenty-eighth Annual Conference on Neural Information Processing Systems (NIPS)*, 2014. 1, 3