Fast Video Classification via Adaptive Cascading of Deep Models

Quickly go through what we have now

Specialization:

1. Reduce length of output in fully connected layers to reduce number of parameters dramatically.
2. Reduce number of convolutional layers
3. Store specialized models on disk and re-use it when we confront the similar locality.

Distillation:

Perhaps always use it to boost accuracy.

Quickly go through what we have now

Probability Layer:

Use the following formula to make use of pre-knowledge, run-time knowledge, and subtle difference in probability which cannot be captured by specialization.

$$P(X = i) = a \cdot softmax(X_i) + b \cdot P_{underlying}(X = i)$$

Main Results

1. Show the existence of temporal locality in daily life. Simplify state-of-the-art architectures and re-train these architectures on specialized datasets.
2. Present an algorithm in optimization to detect locality and choose specialized models

Overall:

Realize end-to-end classification speedups of 2.4-7.8x / 2.6-11.2x (on GPU/CPU) relative to a state-of-the-art convolutional neural network, at competitive accuracy.

Compare with state-of-the-art architectures

1. VGGFace for face detection
2. VGG-16 for scene recognition
3. GoogLeNet for object recognition

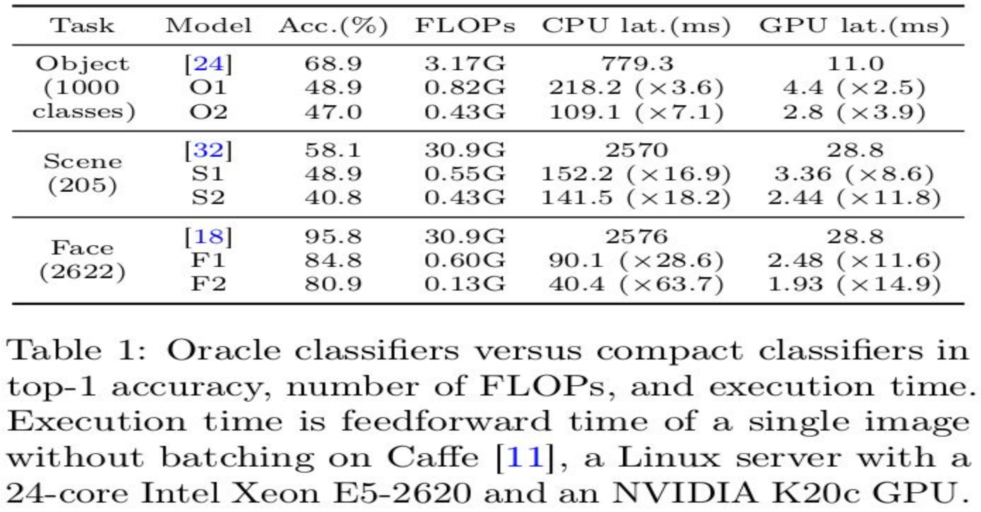
Result 1a: Show class-skew in day-to-day video

Datasets: From Youtube, Shen chose a set of 30 videos of length 3 minutes to 20 minutes from 5 classes of daily activities. Then use VGGFace to classify sampled frames and record what appears in the frame.

Result: For 10s segments, typically roughly 100 frames, 5 objects comprised 90% of all objects in a segment 60% of the time.

Result 1b: Simplify architectures

Method: manually removing layers, decreasing kernel sizes, increasing kernel strides, and reducing the size of fully connected layers.



Task	Model	Acc.(%)	FLOPs	CPU lat.(ms)	GPU lat.(ms)
Object (1000 classes)	[24]	68.9	3.17G	779.3	11.0
	O1	48.9	0.82G	218.2 (×3.6)	4.4 (×2.5)
	O2	47.0	0.43G	109.1 (×7.1)	2.8 (×3.9)
Scene (205)	[32]	58.1	30.9G	2570	28.8
	S1	48.9	0.55G	152.2 (×16.9)	3.36 (×8.6)
	S2	40.8	0.43G	141.5 (×18.2)	2.44 (×11.8)
Face (2622)	[18]	95.8	30.9G	2576	28.8
	F1	84.8	0.60G	90.1 (×28.6)	2.48 (×11.6)
	F2	80.9	0.13G	40.4 (×63.7)	1.93 (×14.9)

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 [11], a Linux server with a 24-core Intel Xeon E5-2620 and an NVIDIA K20c GPU.

Result 1c: Specialization

Dataset: Manually generated specialized datasets, 10 classes and the “other” class.

When will specialize:

Object: 60% skew

Scene: 70% skew

Face: 50 % skew

Result 1c: Specialization

Train speed: 4s for F1 and F2, 14s for O1/O2

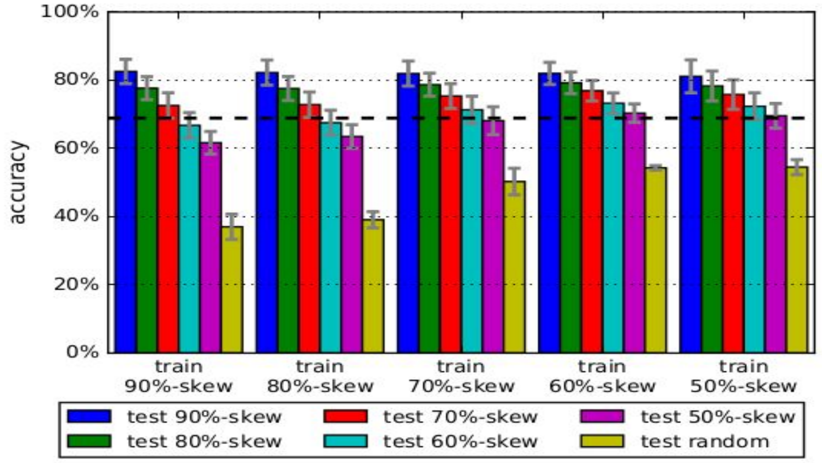
Train method:

1. In the beginning, train all model architectures on the fully, unskewed datasets of their oracles. At test time, when the skew is available, only train the top (fully connected and softmax) layers of the compact model.
2. As a pre-processing step, run all inputs in the training dataset through the lower features.

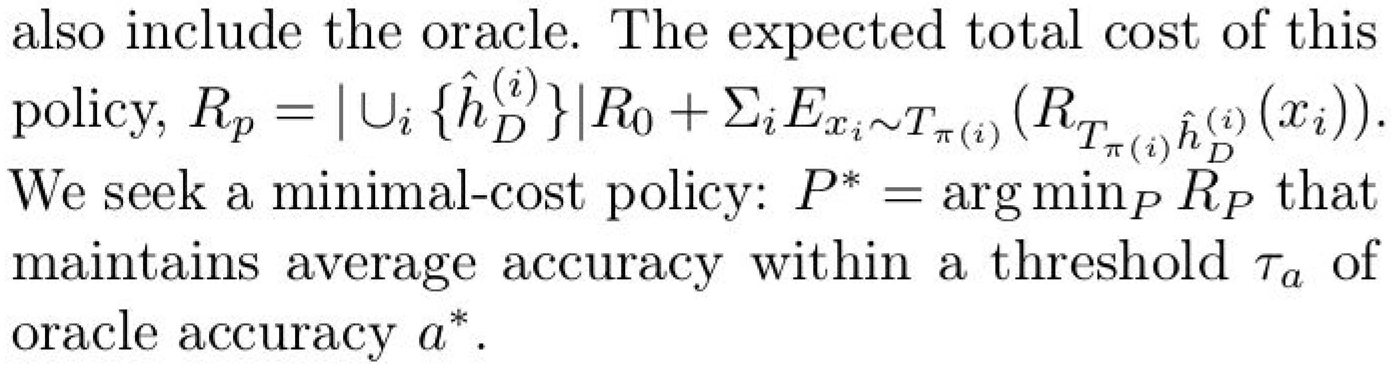
Result 1c: Specialization

Result of O1, which is a simplified model from GoogLeNet

(a)



Result 2: Optimization model



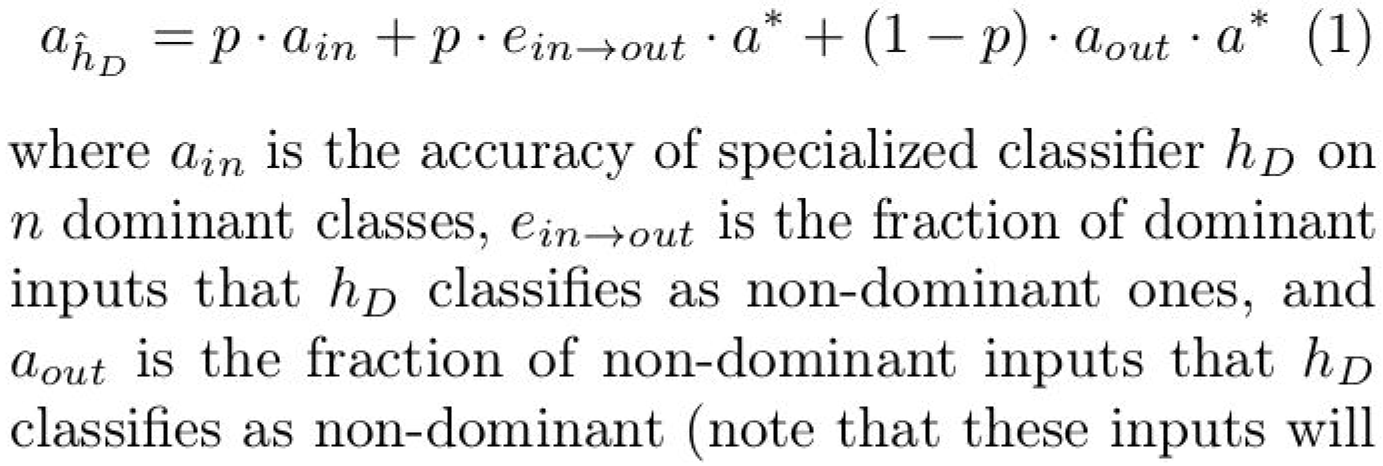
also include the oracle. The expected total cost of this policy, $R_p = |\cup_i \{\hat{h}_D^{(i)}\}|R_0 + \Sigma_i E_{x_i \sim T_{\pi(i)}}(R_{T_{\pi(i)}\hat{h}_D^{(i)}}(x_i))$. We seek a minimal-cost policy: $P^* = \arg\min_P R_P$ that maintains average accuracy within a threshold τ_a of oracle accuracy a^*.

Result 2: Optimization model

Shen brought up an empirical algorithm which summarizes intuitive ideas, such as we should not re-target frequently.

The main goal of the algorithm is to reduce latency while keep accuracy. For keeping accuracy, Shen will pre-calculate accuracy according to models and the size of the dominant class. In the run time, Shen will use the following formula to estimate the accuracy of specialized model. (See next page)

Result 2: Optimization model



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

where a_{in} is the accuracy of specialized classifier h_D on n dominant classes, $e_{in \to out}$ 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

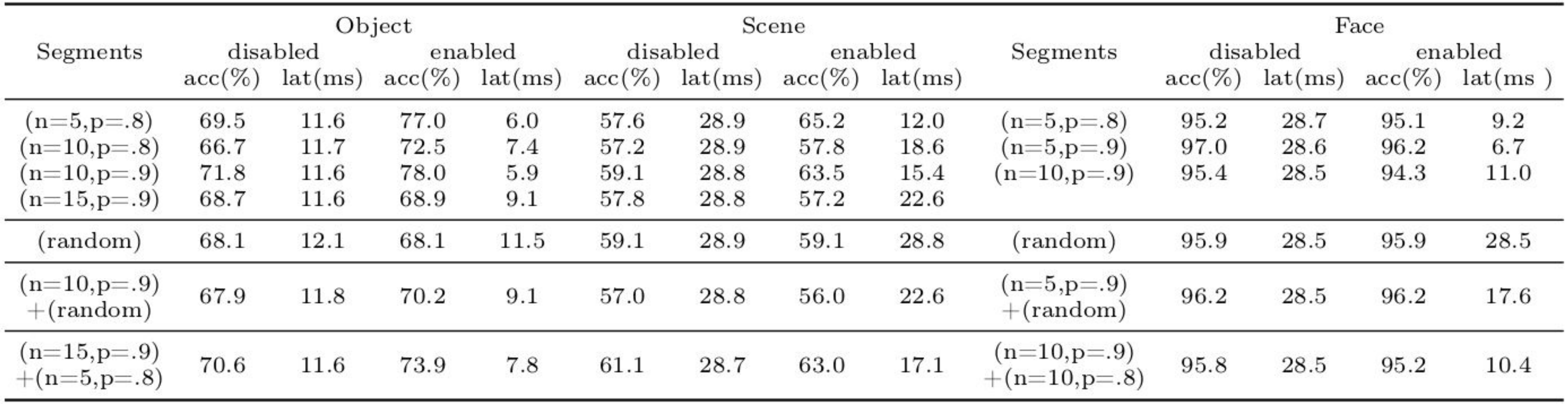
Experiments: Synthetic Experiments

Dataset:

1. Collect specialized dataset from testing dataset.
2. Each image appears for ⅙ seconds.

Experiments: Synthetic Experiments

Results:



Segments	Object				Scene				Segments	Face			
	disabled		enabled		disabled		enabled			disabled		enabled	
	acc(%)	lat(ms)	acc(%)	lat(ms)	acc(%)	lat(ms)	acc(%)	lat(ms)		acc(%)	lat(ms)	acc(%)	lat(ms)
(n=5,p=.8)	69.5	11.6	77.0	6.0	57.6	28.9	65.2	12.0	(n=5,p=.8)	95.2	28.7	95.1	9.2
(n=10,p=.8)	66.7	11.7	72.5	7.4	57.2	28.9	57.8	18.6	(n=5,p=.9)	97.0	28.6	96.2	6.7
(n=10,p=.9)	71.8	11.6	78.0	5.9	59.1	28.8	63.5	15.4	(n=10,p=.9)	95.4	28.5	94.3	11.0
(n=15,p=.9)	68.7	11.6	68.9	9.1	57.8	28.8	57.2	22.6					
(random)	68.1	12.1	68.1	11.5	59.1	28.9	59.1	28.8	(random)	95.9	28.5	95.9	28.5
(n=10,p=.9) +(random)	67.9	11.8	70.2	9.1	57.0	28.8	56.0	22.6	(n=5,p=.9) +(random)	96.2	28.5	96.2	17.6
(n=15,p=.9) +(n=5,p=.8)	70.6	11.6	73.9	7.8	61.1	28.7	63.0	17.1	(n=10,p=.9) +(n=10,p=.8)	95.8	28.5	95.2	10.4

Experiments: Video Experiments

Process:

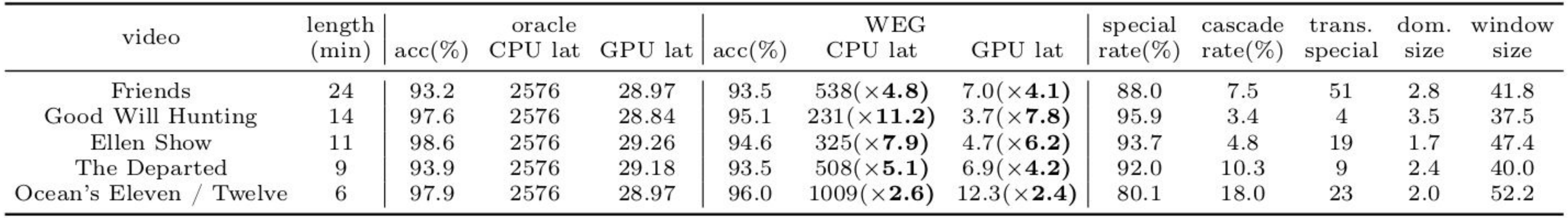
Selected videos from Youtube. Extract all faces from the videos to disk using the Viola Jones detector. Then Shen runned their own algorithm on these faces and measured the total execution time. Dividing by the number of faces gave the average numbers shown here.

A few keys:

1. Number of classes in the video is 2 or 3.
2. Skew is 80%, 88%, 92%, 93%, and 95%.

Experiments: Video Experiments

Result:



video	length (min)	oracle			WEG			special	cascade	trans.	dom.	window
		acc(%)	CPU lat	GPU lat	acc(%)	CPU lat	GPU lat	rate(%)	rate(%)	special	size	size
Friends	24	93.2	2576	28.97	93.5	538(×**4.8**)	7.0(×**4.1**)	88.0	7.5	51	2.8	41.8
Good Will Hunting	14	97.6	2576	28.84	95.1	231(×**11.2**)	3.7(×**7.8**)	95.9	3.4	4	3.5	37.5
Ellen Show	11	98.6	2576	29.26	94.6	325(×**7.9**)	4.7(×**6.2**)	93.7	4.8	19	1.7	47.4
The Departed	9	93.9	2576	29.18	93.5	508(×**5.1**)	6.9(×**4.2**)	92.0	10.3	9	2.4	40.0
Ocean's Eleven / Twelve	6	97.9	2576	28.97	96.0	1009(×**2.6**)	12.3(×**2.4**)	80.1	18.0	23	2.0	52.2

Why is this paper good?

They have a combination of real life implementation (Youtube) and theory analysis (optimization model).

Without theory analysis, it would be too simple.

Without real life implementation, we would compare it with the optimization papers, and Shen’s paper might not be that good.

Compared with ours

1. Probability layer: Shen still used n, p, to detect locality. Probability layer is more general than n,p.
2. Distillation: We will use distillation, which might give a better result.
3. We will use repeated patterns. Thus we have more freedom in retraining specialized models.
4. Shen only considered time/computation, and did not consider memory.
5. To make our paper not simple, we should make an end-to-end framework.
6. Re-train: Shen only re-trains fully connected layers. Perhaps we can have better idea using distillation?
7. Perhaps we should also get some optimization model.