Max-Margin, Single-Layer Adaptation of Transferred Image Features

Praveen Kulkarni[1], Joaquin Zepeda[1], Frédéric Jurie[2], Louis Chevallier[1]
[1] Technicolor, 975 avenue des Champs Blancs, CS 17616,
35576 Cesson Sévigné, France, first.last@technicolor.com
[2] University of Caen Basse-Normandie, CNRS UMR 6072,
ENSICAEN, France, frederic.jurie@unicaen.fr

1. Introduction

Convolutional Neural Networks (CNNs) learned on the ImageNet dataset have been shown to be excellent feature extractors that, combined with linear SVM classifiers, yield outstanding results when transferred to target datasets (*e.g.*, Pascal VOC, MIT Indoor 67 and Caltech) not used during the CNN learning process [**?**]. Given the large number of free parameters in CNN models (tens of millions), learning CNNs directly on these smaller target datasets is a difficult task. Yet recent work [**?**, **?**] has established that it is possible to adapt the transferred CNN parameters to the smaller target dataset to further improve results.

The approach we present herein addresses this scenario by learning a single adaptation layer jointly with linear classifiers under a max-margin objective. Unlike existing adaptation schemes, our learning process is very fast, taking in the order of a few minutes when running on a single core CPU, and does not rely on expensive dataset augmentation methods. We further show that it is possible to obtain very good results with features as little as size 20.

Oquab *et al.* [**?**] have proposed a related CNN adaptation method that consists of re-learning, on the Pascal VOC dataset, the last two layers of the transferred architecture. The learned adaptation layers operate on patches from the original images, with the patch ovelap with the object bounding box determining each patch's label at training time. The approach is hence limited by the availability of expensive bounding box annotations, and in this work we show that comparable results can be obtained without relying on bounding box annotations.

Chatfield *et al.* [**?**] consider an objective based on a hinge-loss that is similar to the max-margin objective we propose herein. Their adaptation approach consists of continuing the learning process began on ImageNet on the new target dataset, but with a slower learning rate. The transition layer between the last convolutional and first fully connected layer, in particular, is updated in the process, a costly procedure (in terms of learning time and required hardware) given the large size of this layer (tens of millions of coefficients).

2. Proposed approach

Our approach consists of jointly optimizing the linear classifiers $\mathbf{w}_1 \dots \mathbf{w}_K$ of the K target classes, along with the model parameters $\mathbf{M} \in \mathbb{R}^{r \times D}, \mathbf{b} \in \mathbb{R}^D$ using

$$\underset{\substack{\mathbf{M},\mathbf{b};\\ \mathbf{w}_1,\dots,\mathbf{w}_K}}{\operatorname{argmin}} \sum_{k=1}^{K} |\mathbf{w}_k|^2 + \frac{C_1}{N} \sum_{i=1}^{N} \ell\left(y_{i,k}\, \mathbf{h}\left(\mathbf{M}\mathbf{x}_i + \mathbf{b}\right)^{\mathrm{T}} \mathbf{w}_k\right) \tag{1}$$

where $\mathbf{h}$ and ℓ represents the Rectified Linear Unit (ReLU) and hinge loss operator, respectively, $\mathbf{x}_i \in \mathbb{R}^D$ are the CNN feature representations of the image and $y_{i,k} \in \{-1, 1\}$ are labels indicating the absence/membership of image i in class k. For notational simplicity, we disregard the classifiers' bias terms.

Block-coordinate SGD. We use block-coordinate Stochastic Gradient Descent (SGD), sequentially updating $\mathbf{w}_1, \dots, \mathbf{w}_K$, $\mathbf{M}$ and $\mathbf{b}$ on the same batch of b images, correspondingly using b SGD steps per block of coordinates before randomly drawing a new batch (without repetition in each epoch). We initialized $\mathbf{M}$ using r randomly selected training features. Bias term $\mathbf{b}$ and $\mathbf{w}_1, \dots, \mathbf{w}_K$ are initialized to zero.

Early stopping. To avoid over-fitting of the model to the training data, we use an early stopping criterion guided by the performance over the validation set.

Adaptive learning rate. We use an adaptive learning rate for $\mathbf{M}$ that starts at 1 and decays by half to an empirical minimum of 10^{-2}. This rate is updated every 200 images using cross-validation by running SGD on 50 training images chosen from the next batch. At the end of the process, the final learning rate

Method	Train time	Dim	# params.	mAP
PRE1000C [?]	≈ 1 day	-	∼8.5M	77.73
CNN S TUNE-RNK[?]	-	4K	∼100M	82.42
Ours	120s	20	2759	76.24
Ours	190s	70	9550	77.58

Table 1. Comparison of our proposed method with two existing CNN adaptation schemes.

is used to do a single training run over the validation images.

3. Results

We evaluate our method using Pascal VOC 2007 as a target dataset, using the standard mean Average Precision (mAP) performance measure. This dataset consists of 4192 test and 5011 training images. We hold out 811 training images, choosing them uniformly over all classes, and use them as a validation set. Each image is represented using the VGG-M (128-dimensional) CNN model [?].

In Fig. 1 we evaluate the impact on test set mAP of varying the numer of rows r of $\mathbf{M}$. We consider two cases: joint optimization of $\mathbf{M}$ and the classifiers, as per (1), and optimization of only the classifiers, keeping $\mathbf{M}$ fixed to its initialization value. We can observe that our performance is constant for all output feature sizes r. The adaptation layer provides a large advantage for all r values of as much as 17 points in mAP for $r = 20$, and 1.2 points for $r = 128$.

In Table 1 we compare our method with two state-of-the-art CNN adaptation methods [?, ?]. Our result of 77.58 is comparable to the 77.73 mAP of [?], yet our model has $4e3\times$ less free parameters and takes only a few minutes to learn on a single core CPU ([?] takes close to one day on a GPU). The results of [?] are better than ours, but their CNN model is even larger than [?], with a comparable increase in learning time and required processing power.

In Table 2 we show the advantage of using an adaptive learning rate versus a fixed learning rate by displaying the test set mAP after a fixed number of iterations. In Fig. 2, we illustrate our early stopping approach by plotting the test set, training set and validation set mAP for a sample run.

In Fig. 3 we evaluate the effect of batch size in our block-coordinate SGD optimization process, noting that, for a fixed number of epochs, larger batch sizes (up to the training set size) result in better performance.

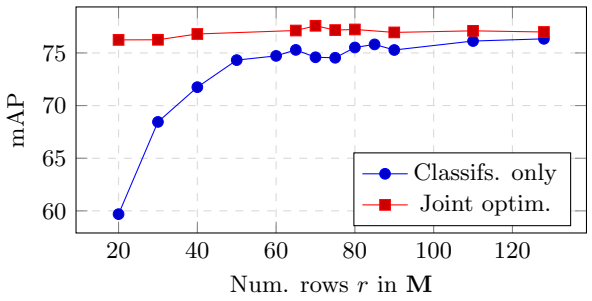

Figure 1. Effect of varying the number of rows r in $\mathbf{M}$ on performance. Here we also compare the results of Joint Optim vs Classifs.

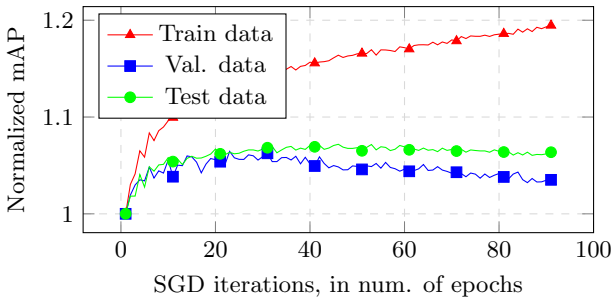

Figure 2. Illustration of cross-validation strategy: the optimal parameters are chosen based on the best performance on the validation set, which occurs at approximately 20 epochs.

	Fixed					Adaptive
Learn rate	10^{-5}	10^{-2}	0.09	0.5	1.0	-
mAP	74.66	75.60	76.81	74.96	73.43	77.25

Table 2. Test set mAP when using fixed learning rate and adaptive learning rate.

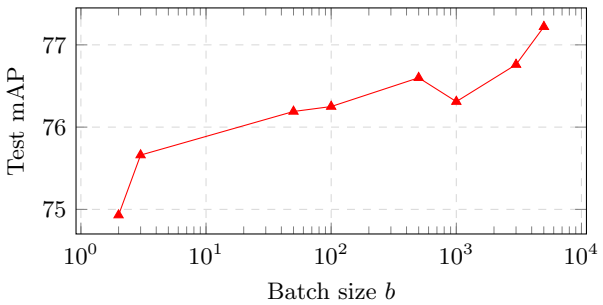

Figure 3. Performance as a function of batch size for 20 epochs of training.