Learning the Structure of Deep Architectures Using ℓ_1 Regularization

Praveen Kulkarni[1]
Praveen.Kulkarni@technicolor.com

Joaquin Zepeda[1]
Joaquin.Zepeda@technicolor.com

Frederic Jurie[2]
frederic.jurie@unicaen.fr

Patrick Pérez[1]
Patrick.Perez@technicolor.com

Louis Chevallier[1]
Louis.Chevallier@technicolor.com

[1] 975 avenue des Champs Blancs,
CS 17616, 35576 Cesson Sévigné, France
http://www.technicolor.com

[2] University of Caen Basse-Normandie,
CNRS UMR 6072, ENSICAEN, France http://www.unicaen.fr

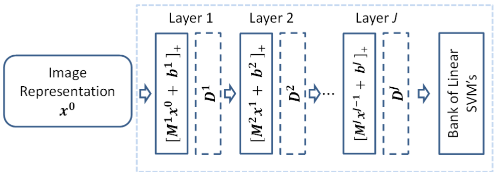

Figure 1: **Proposed deep processing pipeline**. Given an image representation, e.g., the output of the convolutional part of a pre-trained state-of-art DNN, J fully connected layers, each involving a diagonal matrix that controls its effective dimensions, are jointly learned with final linear SVM classifiers.

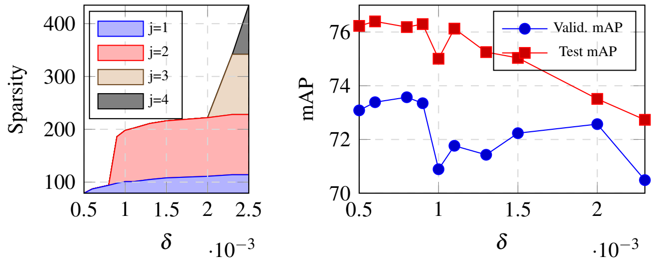

Figure 2: Effect of the penalty weight δ on *(left)* the number of zero diagonal entries of $\mathbf{D}^j$, $j = 1,\ldots,4$, and *(right)* on the classification performance as measured by mAP. The zero diagonal entries are presented as stacked plots so that the vertical displacement of any shaded regions corresponds to the number of zero diagonal entries of $\mathbf{D}^j$ for the corresponding layer.

Our proposed approach is illustrated in Fig. 1. The architecture we consider consists of a sequence of fully-connected layers, with a diagonal matrix between them. We present a method that automatically selects the size of the weight matrices inside fully-connected layers indexed by $j = 1,\ldots,J$. Our approach relies on a regularization penalty term consisting of the ℓ_1 norm of the diagonal entries of diagonal matrices inserted between the fully-connected layers. Using such a penalty term forces the diagonal matrices to be sparse, accordingly selecting the effective number of rows and columns in the weights matrices of adjacent layers. We present a simple algorithm to solve the proposed formulation and demonstrate it experimentally on a standard image classification benchmark.

We can express the architecture in Fig. 1 as a concatenation of units of the following form:

$$f^j(\mathbf{x}) = \mathbf{D}^j \left[\mathbf{M}^j\mathbf{x} + \mathbf{b}^j\right]_+, \tag{1}$$

where $[\mathbf{z}]_+ = [\max(0, z_i)]_i$ is the commonly used Rectified Linear Unit (ReLU) non-linearity. Each layer is defined by diagonal matrix $\mathbf{D}^j$ and matrix $\mathbf{M}^j$. A deep architecture can be derived from (1) using the standard stacking approach. Letting $\circ$ denote the composition operator such that $f \circ g(\mathbf{x}) = f(g(\mathbf{x}))$, this can be denoted as

$$f^J \circ \ldots \circ f^1(\mathbf{x}), \tag{2}$$

where, in this case, the vector $\mathbf{x}$ denotes the representation of the image at the input of the architecture. The image representation can consist of a direct re-ordering of the RGB values in the image [1, 2], or it can be a feature derived from the image [3], which is the approach we follow in the present work.

Besides the variables $(\mathbf{M}^j, \mathbf{D}^j, \mathbf{b}^j)_{j=1}^J$ in (1), one needs to learn the vectors $\{\mathbf{w}^k\}_{k=1}^K$ that define the SVM classifiers for the K classes. We will learn these variables from an annotated training set comprised of N training images $\mathbf{x}_i, i = 1,\ldots,N$, each with K labels $y_i^k \in \{-1,1\}, k = 1,\ldots,K$ indicating whether image i belongs to class k or not. Given such a training set, our approach consist of minimizing the following objective over all the variables $\{(\mathbf{M}^j, \mathbf{b}^j, \mathbf{D}^j)\}_{j=1}^J$ and all the classifiers $\{\mathbf{w}^k\}_{k=1}^K$:

$$\frac{1}{K}\sum_{k=1}^{K}\left(\|\mathbf{w}^k\|_2^2 + \frac{C}{N}\sum_{i=1}^{N} l\left(y_i^k \left(f_J \circ \ldots \circ f_1(\mathbf{x}_i)\right)^\top \mathbf{w}^k\right)\right) + \delta\sum_{j=1}^{J}\|\mathbf{D}^j\|_* + \mu\sum_{j=1}^{J}\|\mathbf{M}^j\|_F^2. \tag{3}$$

In the above expression, we have used *(i)* $l(x)$ to denote the hinge loss, given by $\max(0, 1-x)$; *(ii)* $\|\mathbf{D}\|_*$ to denote the trace norm, given by $\sum_i |D_{ii}|$ for the case of diagonal $\mathbf{D}$; and *(iii)* $\|\mathbf{M}\|_F^2$ to denote the squared Frobenius norm $\sum_{ij} M_{ij}^2$.

To illustrate the motivation behind this learning objective, we note first that the terms inside the summation over k in (3) are recognizable as an SVM objective for class k, where the scalar C is the SVM regularization parameter. The feature vectors used within this SVM objective are given by $f_J \circ \ldots \circ f_1(\mathbf{x}_i)$, which depends on $\{(\mathbf{M}^j, \mathbf{b}^j, \mathbf{D}^j)\}_{j=1}^J$. Hence we are learning the classifiers jointly with the feature extractor used to represent the input images.

The two regularization terms comprised of summations over j in (3) have two important purpose. First is to keep the SVM terms from decreasing indefinitely. A second important purpose is to automatically select the shapes of the weights matrices $\mathbf{M}^j$ and $\mathbf{b}^j$ and ℓ_1 norms applied to diagonal matrices such as $\mathbf{D}^j$ are sparsity inducing norms.

In order to minimize (3), we will employ a block-coordinate SGD approach. We evaluate our method on Pascal VOC 2007 dataset and compare it against various state-of-the-art algorithms. Our learning algorithm is governed by three important terms: the penalty weights μ and δ and and the number of training epochs T. The number of training epochs is determined using the validation set. Further, we evaluate the performance of our method as a function of the number of layers J in the architecture.

In Fig. 2, we plot both the sparsity for all layers and the corresponding test and validation mAPs when varying the penalty weight δ. Note that increasing δ drastically increases the number of zero diagonal entries in the architecture while only slightly affecting the classification performance.

[1] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the Devil in the Details: Delving Deep into Convolutional Nets. In *British Machine Vision Conference*, 2014. URL `http://arxiv.org/abs/1405.3531`.

[2] Alex Krizhevsky, I. Sutskever, and Geoffrey Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Neural Information Processing Systems*, pages 1–9, 2012.

[3] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. *Computer Vision and Pattern Recognition*, 2014.