Object recognition & Computer vision
Project report - MVA 2015/2016
Training Convolutional Networks with Noisy Labels

Maha ELBAYAD
ENS Cachan
maha.elbayad@student.ecp.fr

Abstract

The performance of Convolutional neural networks depends on the amount of labeled samples and their presumable quality, and since hand labelling is impractical, a shift toward semi-automatic labelling is inevitable which means more inaccuarte, subjective and sometimes misleading data that must be handled differently. For these reasons the method proposed in [4] handles the labels noise in an intuitive yet performant manner.

1. Introduction:

The main objective of the studied paper is to assess the impact of the noise level on the classification error. The method outline consists of two phases: (1) training a convolutional neural network on the noisy data, (2) finetune the network with an additional constrained linear layer that would mimic a confusion matrix to retrieve a prediction closer to the true labels. Although the article deals with two types of noise, namely the **the label flip** and **the outlier**, we would focus for this project solely on the label flip.

2. The method:

2.1. Noise modelling:

Given a trining set $\mathcal{X} = \{x_n, y_n\}_{1 \leq n \leq N}$ where y_n is the true label of the entry x_n. $y_n \in \{1...K\}$ in the contest of a K-multiclass classification. The additional noise layer has optimally for weights the confusion matrix $\mathbf{Q}$ defined as:

$$\mathbf{Q} = (q_{ij})_{1 \leq i,j \leq K},\ \forall i,j \in \{1,..K\}\ q_{ij} = \mathbb{P}(\tilde{y} = i | y = j)$$

Where $\tilde{y}$ is the noisy label. In other words q_{ij} is the probability of the class j being mistaken for class i

By the sum rule we can evaluate the noisy label as:

$$\mathbb{P}(\tilde{y} = i | x) = \sum_j q_{ij} \mathbb{P}(y = j | x) \tag{E1}$$

2.2. The network architecture

For the base network, prior to learning the confusion matrix, we implement the architecture proposed in [3] with response normalization and pooling: the CNN contains 3 convolution layers and a single fully-connected layer whose output for the sample x_n, $(fc_{n,k})_{1 \leq k \leq K}$ is fed to a softmax loss:

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{N} \sum_{i=1}^{N} \log(\hat{p}_{n,\tilde{y}_n}), \quad \hat{p}_{n,k} = \frac{\exp(fc_{nk})}{\sum_k \exp(fc_{nk})} \tag{E2}$$

We denote with $\mathbf{W}$ the weights of this base network wich in addition to the matrix $\mathbf{Q}$ form the parameters of our model.

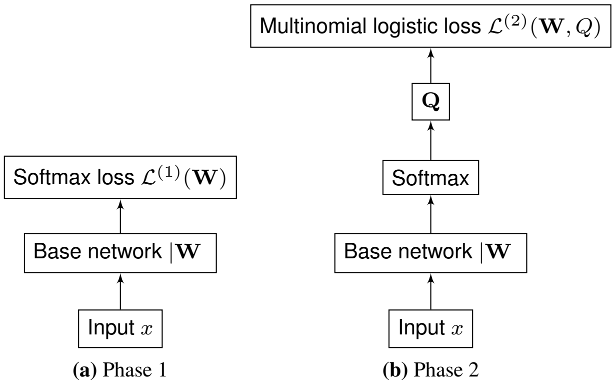
Multinomial logistic loss $\mathcal{L}^{(2)}(\mathbf{W}, Q)$
Q
Softmax loss $\mathcal{L}^{(1)}(\mathbf{W})$
Softmax
Base network $|\mathbf{W}$
Base network $|\mathbf{W}$
Input x
Input x
(a) Phase 1 **(b)** Phase 2

Figure 1: Model outline

Figure (1) illustrates the outline of the model where for the second phase we finetune the parameter $\mathbf{W}$ in addition to the extra linear layer with weights $\mathbf{Q}$. As $\mathbf{Q}$ expects probabilities for its input we insert a softmax layer to process the fully-connected layer's output. For similar reasons we use the multinomial logistic loss as the outputs are already a probability distribution.

$$\mathcal{L}^{(2)}(\mathbf{W}, Q) = -\frac{1}{N}\sum_{n} \log \hat{p}(\tilde{y} = y_n | x_n, \mathbf{W}, \mathbf{Q}) \quad \text{(E3)}$$

2.3. The confusion layer

The article proves that minimizing the loss of (E3) with an additional regularizer forces the predicted distribution $\mathbf{Q}^*$ from the combined model to be as close as possible to the true confusion matrix $\mathbf{Q}$. To do so we use a weight decay on $\mathbf{Q}$ which will diffuse it.

In addition to the weight decay, and as we want $\mathbf{Q}$ to be optimally a stochastic matrix along its second dimension (each column sums to 1) we will scale the matrix $\mathbf{Q}$ at each each forward propagation.

3. Experiments

4. The environment

For our experiments we use Caffe, the deep learning framework by the BVLC [1], to implement our confusion layer which is a tweaked version of the `innerProduct` layer. The training of the baseline networks has been performed on a GPU node of the Mesocentre de calcul de Centrale Paris, while the finetuning is done on CPU.

4.1. CIFAR10

We compare the performance of the model with and without the confusion layer on a manually corrupted dataset at variant noise levels. We're using CIFAR10 [2] a subset of 80 million Tiny Images dataset of natural images labeled into 10 object categories, 50k images are allocated to the training set and another 10k for the test set.

We train the first part of the model in 70000 iterations each with a batch of 100 images (i.e 140 epochs) with a learning rate varying as shown in table (1)

Model	Epochs	lr	weight decay
Baseline	1-120	1e-3	1 on fc
	121-130	1e-4	..
	131-140	1e-5	..
Confusion	140-145	1e-4	.05 on Q

Table 1: Learning rate policy

While training the network, we keep track of the accuracy of the baseline network and the accuracy of the noise model prior to applying the confusion layer as it's supposed to absorb the noise and force the fully connected layer to output adapted scores. We note that the reduction of the learning rate by a factor of 10 twice allows the network to further minimize the loss and boost its accuracy. The second jump of accuracy, albeit less noticeable, at the beginning of Q's finetuning proves that the layer is absorbing the noise as intended.

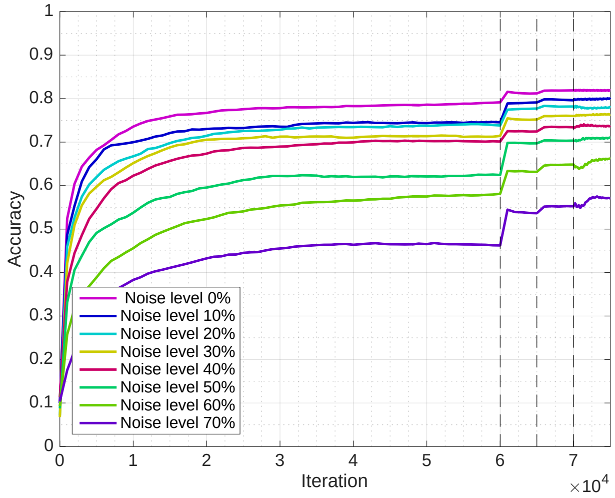

Figure 2: Accuracy during training at variant noise levels (CIFAR10)

The figure (3) shows the difference between ground truth confusion matrix $\mathbf{Q}$ and the learnt $\mathbf{Q}^*$. Empirically this method has more impact at high noise level although the accuracy overall drops, but compared to the baseline model we get around +2 points of accuracy.

4.2. Casia WebFace Database

To test the method on realistic label noise we'll use a semi-automatically labeled subset of the Casia Webface database [5] to perform a binary classification of facial images (96 × 96) with (1) or without (0) glasses.

If the model described above doesn't assume symmetric label noise (noise independent of the true label) it still does assume that the noise is independent of the input x. And with a minimal number of classes (binary classification) the number of additional parameters in Q is at its minimum. In these settings we will test the performance of the noise layer. In addition to that, having an unbalanced dataset with only 5% of the images in the positive class (1) we will introduce an adapted noise with asymmetric level as well: a noise level referred to as $\alpha(1)$ corresponds to a flip from 0 to 1 with probability α and a flip from 1 to 0 with probability $19 \times \alpha$ to maintain the distribution of the labels (5% of class 1) and $\alpha(2)$ corresponds to a flip from 0 to 1 with probability α and a flip from 1 to 0 with probability $\frac{\alpha}{10}$.

Our data is split into 4152 images for the training and 4661 for the test set.

The baseline network has 3 blocks of {Conv, Pool, ReLU}

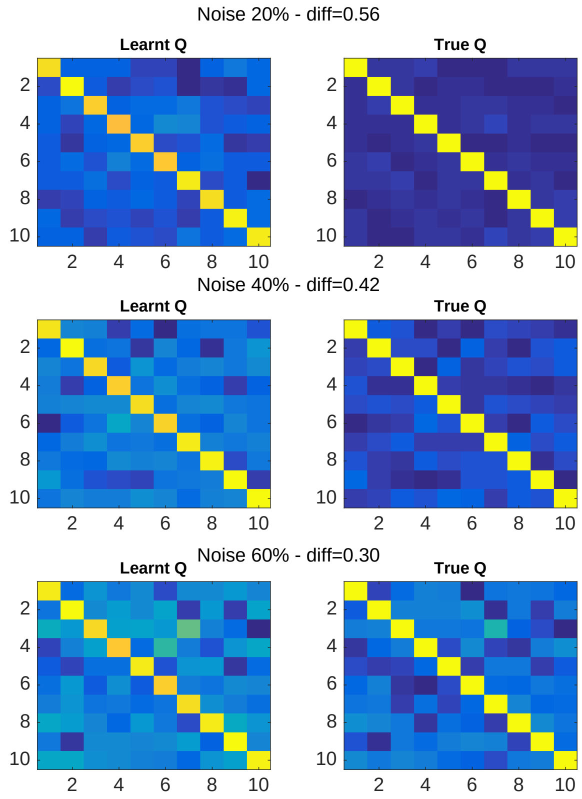

Figure 3: Confusion matrices: the learnt and the truly used to generate the noisy labels - the bright colors correspond to a value close to 1 - the diff value is for the N_2 distance between the two matrices (CIFAR10)

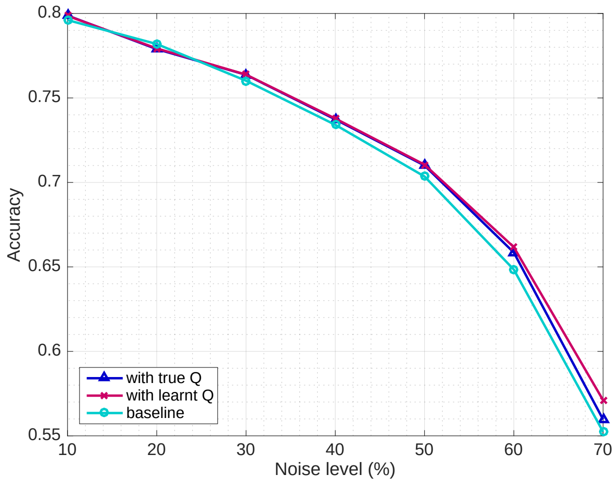

Figure 4: Test accuracy when trained on 50k images with variant noise level (CIFAR10)

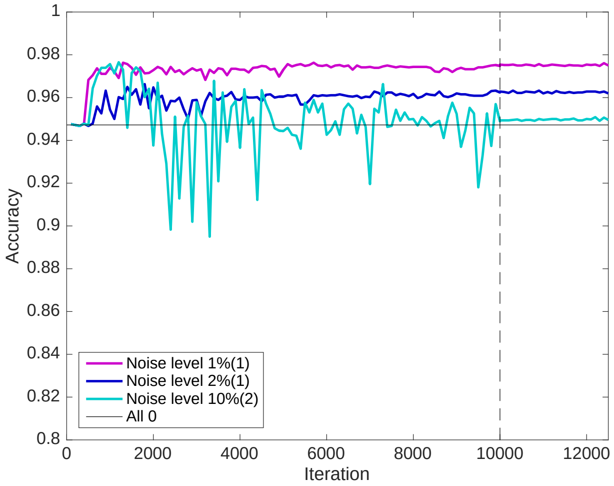

Figure 5: Accuracy during training at variant noise levels (CASIA)

and 2 fully-connected layers that lead to a softmax layer. The learning rate is decreasing throughout the 10000 training iterations (≈ 154 epochs):

$$lr(iter) = \frac{0.01}{(1 + iter \times 10^{-4})^{0.75}}, \; iter \leq 10^4$$

The $\mathbf{Q}$ matrix tuning takes 2500 iterations ($\approx$ 38 epochs) with a fixed learning rate of 10^{-5} and a weight decay of 0.05.

The learnt confusion matrix gets close to the true matrix on its second column - class 1, while often converging to $(1, 0)^T$ in its first column - class 0 given the skewed distribution of the labels. Thus, the accuracy is weakly affected by the introduction of the layer.

Learnt Q		True Q	
1.00	0.20	0.99	0.19
0.00	0.80	0.01	0.81

(a) Noise level 0.01(1)

Learnt Q		True Q	
1.00	0.21	0.98	0.38
0.00	0.79	0.02	0.62

(b) Noise level 0.02(1)

Learnt Q		True Q	
1.00	0.06	0.90	0.01
0.00	0.94	0.10	0.99

(c) Noise level 0.1(2)

Learnt Q		True Q	
1.00	0.00	0.80	0.02
0.00	1.00	0.20	0.98

(d) Noise level 0.2(2)

Table 2: Confusion matrices: the learnt and the truly used to generate the noisy labels

5. Discussions

Our implementation of the confusion layer in the way described in the article and with the columns normalization mentioned above does generally succeed to learn the confusion matrix, yet the layer in itself doesn't boost the performance of the network, especially in the case of natural noise. Nonetheless the learnt $\mathbf{Q}^*$ can be used to bootstrap the training set or simply for the small gain of accuracy given that the finetuning of $\mathbf{Q}$ doesn't require much computation.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[2] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus. Training convolutional networks with noisy labels, 2015. proc, ICLR.

[5] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.