Noisy but Plenty of Data: what makes Deep Learning work (or not)

Anonymous BTAS 2016 submission

Abstract

Recent studies show that improvement of CNN performances rely on increasingly deep network architecture. Large training databases are then needed to learn all parameters of such networks. Due to the cost of hand-labeling tasks, a shift toward semi-automatic labeling is considered, which means partially incorrect labels in the training database. We consider here the problem of noise handling when learning classifiers from automatically labeled data. To tackle with this issue, we make this study on a binary classification problem: determine the presence of glasses on a face. In this context, we will have to handle the presence of image dependent noise (initial labels are obtained by an existing inaccurate classifier). Different experiments are made to show the impact of label noise on classification performances. We first study the influence of the dataset parametrization (size, class balance...) on clean and noisy data. Due to the important degradation of performances in the second case, we propose methods to handle noisy labels during preprocessing and/or while learning. Discussion over the best approaches are given to suggest guidelines for further work.

1. Introduction

With the rise of deep learning and the improved ability to build huge datasets extracted from multiple sources, Convolutional Neural Networks (CNN) classifiers can be designed and trained to perform numerous tasks. CNN have shown to be efficient on complex pattern recognition tasks, notably image classification [7]. But as the datasets grow bigger, it becomes harder to have ground-truth labels associated with the training data for every new problem. When hundred of thousands of samples are collected in a semi-automated fashion, hand-labeling them is a task too time-consuming to be accomplished at a reasonable cost. Automatic annotation may be used to obtain labels in a faster way: this labeling can rely on multiple techniques (image processing, simpler machine learning models...) with variable degrees of accuracy. These methods introduce labeling errors in the training set, ie. a source of noise that the classifier will have to deal with during its learning phase. Such noise results from multiple sources, among others, random errors, bias in the annotator and quality of the samples [12], and can decrease the performances of the classifier [2]. Therefore, handling this noise in a supervised or semi-supervised way is a currently active field of research. Most of the studied methods can be separated in two groups: preprocessing of the noise through filtering of the dataset, and accounting for the noise in learning process itself [3].

Preprocessing methods range from simple thresholding that reproduce an observed noise repartition to more complex setups where specific assumptions on the sources of noise are made [4]. Other techniques focus on training a first classifier using a small hand-labeled subset as a reference, designed to predict the level of noise of a given sample. The full dataset is then relabeled using this classifier before training a standard CNN.

In the second group, methods were introduced which implement new activation layers designed to handle noisy samples, as: uncertain neural network for uncertain data classification (UNN) [5], surrogate loss functions that distinguish between samples taken from a hand-labeled subset and samples with noisy annotations [9] and the use of multiple labels generated in a preprocessing pass where new labels are built to denote the probabilities that a sample falls within a class of noise (e.g. noise free, pure random noise or confusing noise), these probabilities can then be combined to tune another classifier during the training phase [12]. The introduction of a confusion matrix, either designed from observations on the dataset or learned during the training phase of the classifier [11], follows a similar logic: the confusion matrix is fixed during a first step of the training, and is unconstrained during a second learning step.

Each of these methods makes different assumptions on the noise distribution: the noisy labels are either assumed independent of the true label [8] or dependent on the true label but conditionally independent with the image [11] which is unrealistic and ill-suited to confusing labels related to some challenging samples.

To avoid any noise modeling, more general methods of performance enhancements from the machine learning field can be transposed to our subject. Bootstrapping was espe-

cially studied, where a combination of the initial training labels and the current classifiers predictions are combined to generate new labels for the next training iteration [10].

These approaches are going to be studied in this article. In the second section of this paper, we lay out our general experimental setting and the chosen baseline model. In section 3, we analyze some dataset statistics that might interfere with the model performance before describing preprocessing methods to limit the noise level. Then, we will focus on the confusion matrix technique [11] in Section 4 before introducing some bootstrapping schemes in Section 5. The last section introduces an alternative approach based on a CNN noise prediction to correct noisy labels, before concluding over the different experiments led in this paper.

2. Learning and evaluation framework

2.1. Convolutional Neural Network model

Our model is inspired by the CNN architecture described in [14], with 3 blocks of convolution, ReLU and max-pooling layers followed by 2 fully-connected layers whose output for the sample x_n, $(fc_{n,k})_{1\leq k\leq K=2}$ is fed to a softmax loss i.e. we minimize the multinomial logistic loss over the whole training set $\mathcal{X} = \{x_n, y_n\}_{1\leq n\leq N}$:

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

The regularization is assured with dropout on the fully-connected layers. Input data are 96×96 pixels images in color aligned on eyes position. The architecture of the chosen network is summarized in Table 1.

Layer name	type	Filter size stride	output size $C \times H \times W$
Conv1-ReLU1	convolution	3x3/1	32 96 96
Pool1	Max pooling	2x2/2	32 48 48
Conv2-ReLU2	convolution	3x3/1	64 48 48
Pool2	Max pooling	2x2/2	64 24 24
Conv3-ReLU3	convolution	3x3/1	128 24 24
Pool3	Max pooling	2x2/2	128 12 12
FC1	fully-connected		512
drop1-ReLU4	Dropout (50%)		
FC2	fully-connected		2
drop2	Dropout (50%)		
loss	Softmax loss		1

Table 1: Network configuration.

The learning hyperparameters are fixed for the all experiments (cf. Appendix). In fact, our goal is to tackle with the label noise, and not to fine-tune the CNN hyperparameters. We settled for an ideal number of iterations of 10k after different experiments (more led to overfitting). We use the Caffe framework [6] to train and test our neural networks using a Nvidia Tesla K40 GPU.

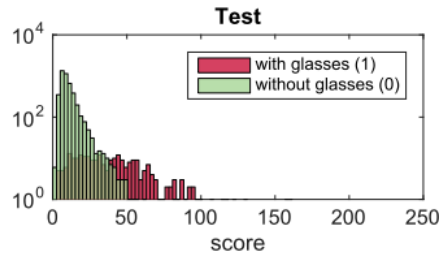


Figure 1: Old detector: score distribution on `Test` base.

2.2. Dataset

We experiment with the CASIA WebFace database [13] containing 380,000 face images of about 10,000 subjects semi-automatically collected from the internet. We hand-labeled 4661 images from the CASIA database for validation (denoted `Test`) and another 4152 images for clean training (denoted `Tr1`), the rest (denoted `CASIA`) is labeled with a glasses detector based on edges information, especially the presence of the bridge of the glasses between the two eyes. The scores of this detector (called *old* detector thereafter) range from 0 to 255 and their distribution given the true label of an input image on manually annotated base `Test` is shown in Figure 1.

We enriched our training set with 4151 images manually labeled (denoted `Tr2`) that have higher image quality and less compression artifacts. These images are part of the available images of [14]. The ratio of positive labels is higher in this second database: 19%, against 5% for `Tr1`, but in both cases, the datasets are skewed and unbalanced. We denote with `Tr3` the fusion of `Tr2` with a balanced subset of `Tr1` (1070 images) to maintain the positives ratio of `Tr2`. The datasets statistics are shown in table 2.

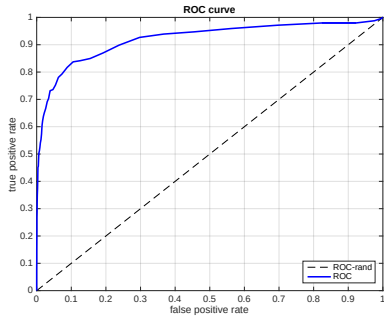
Set	#images	#with glasses	ratio
`CASIA`	381949	unknown	unknown
`Test`	4661	246	5.3%
`Tr1`	4152	214	5%
`Tr2`	4151	789	19%
`Tr3`	5221	1003	19.21%

Table 2: Datasets statistics.

2.3. Evaluation criteria

We use the following indicators to assess the performances of our experiments.

- **Accuracy (Acc):** it corresponds to the number of correct predictions over the number of tested samples.
- **Receiver Operating Characteristic (ROC):** it expresses the true positive (TP) rate as a function of the false positive (FP) rate. The **Area Under the ROC Curve (AUC)** is equal to the probability that the classifier will find the positive image among a random im-



(a) ROC curve

Figure 2: Old classifier: Acc=0.96, *AUC*=0.92, AP=0.679.

age labeled 1 and a random image labeled 0.

- **Precision-Recall (PR):** it expresses the precision (number of TP defined as positive by the classifier divided by the total number of images labeled as positive by the classifier) as a function of recall (number of TP defined as positive by the classifier, over the total number of TP). The **Average precision (AP)** is the area under the PR curve.

We evaluate the performance of the old classifier using these indicators. By thresholding the scores obtained on the `Test` set at some point t, we can assess the classifier performance. A ratio of 5% of glasses has be measured n the `Tr1` database, we therefore set a threshold on the old classifier scores to get a ratio of 5% of positive labels. Performances of this detecteor are given in Figure 2. We will refer to them as `perfOld` in the following sections.

3. Training set statistics and preprocessing

3.1. Training set size and label balance

In this first part, we experiment with varying the training set size and label balance to determine the CNN behaviour in two different modes: with clean labels and with noisy labels defined from the old classifier.

All the models below are trained with the softmax default loss described in section 2.1. For the clean datasets, we start with `Tr3` as learning database and decrease the size iteratively in order to build nested training sets. We also shows the performance for different ratios of positive labels. On the noisy CASIA base, the old classifier scores are binarized following a simple model: a threshold t is fixed and all images with a score lower than t are labeled 0, the others are labeled 1. Figures 3 and 4 show the impact of these parameters on performances. We note that the training set size is more impactful when the labels are clean, in fact:

Clean labels:

- The *AUC* strongly increases with the dataset size at fixed positives ratio: for $r = 20\%$: see curve (2) vs. curve (1) and $r = 5\%$: (4) vs. (3).

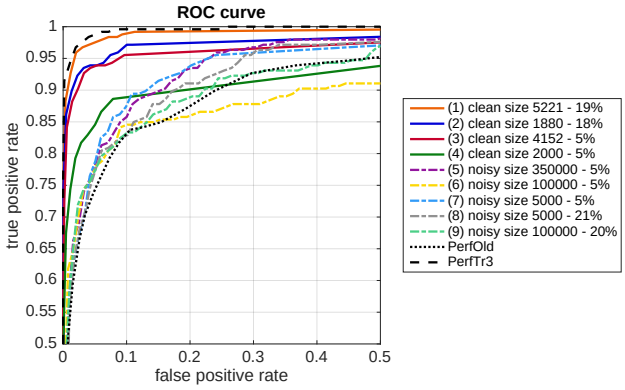


Figure 3: ROC curves with different training database sizes, modes (clean/noisy) and ratios r.

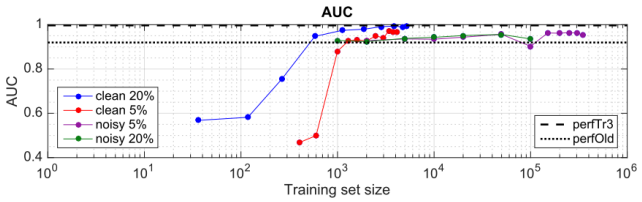


Figure 4: *AUC* performances with different sizes and different r: clean vs noisy.

- The *AUC* increases with the ratio of the observed positive samples at a fixed size: size $\approx$ 2000: (4) vs. (2).

Noisy labels:

- The training set size seems to have little to no effect on the performance: $r = 5\%$: (7) vs. (6) then (6) vs. (5) and $r = 20\%$: (8) vs. (9).
- the positives ratio is ineffective too with a fixed size: size = 2000: (8) vs. (7)

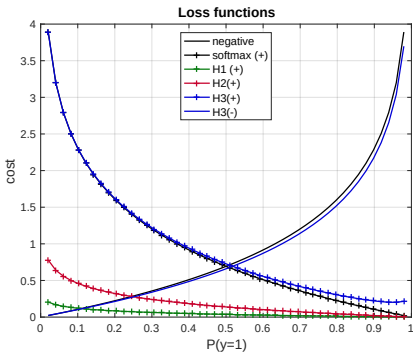
Tests on clean databases show the importance of having a well balanced representation of the different labels. This parameter seems to affect more the training step than the database size, at least without noise.

3.2. Class representation balance

Due to the difference between the positive and negative label proportion, we train our baseline model with a modified loss function which inherits from the multinomial logistic loss:

$$\tilde{\mathcal{L}} = -\frac{1}{N}\sum_i \left(H_{y_n,0}.\log \hat{p}_{n,0} + H_{y_n,1}.\log \hat{p}_{n,1}\right)$$

as suggested in [9].To achieve proportional misclassification costs, we set: $H = \begin{pmatrix} c_{01} & c_{00} \\ c_{11} & c_{10} \end{pmatrix}$, where $c_{i,j}$ is the cost of predicting i when the ground truth is j. On Figure 5, we can see the effect of different values of H on the loss functions for negative and positive samples.



$$H1 = \begin{pmatrix} 1 & 0 \\ 0 & 0.05 \end{pmatrix} \quad H2 = \begin{pmatrix} 0.95 & 0 \\ 0 & 0.2 \end{pmatrix} \quad H3 = \begin{pmatrix} 0.95 & 0 \\ 0.05 & 1 \end{pmatrix}$$

Figure 5: Loss function per cost matrix.

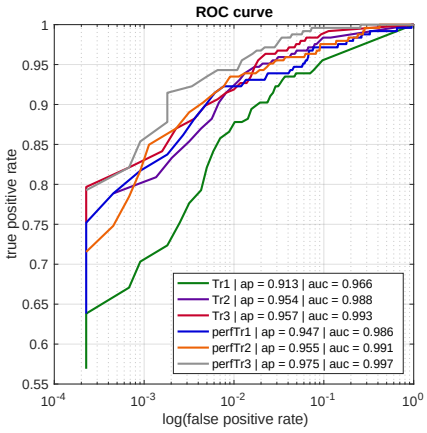


Figure 6: ROC curves on each training set with a modified loss (perfTrX) vs. common loss (TrX).

The best performances on `Tr1`, `Tr2` and `Tr3` were achieved with the following parameters:

$$H_{Tr2} = H_{Tr3} = \begin{pmatrix} 0.99 & 0.01 \\ 0.2 & 0.8 \end{pmatrix} \quad H_{Tr1} = \begin{pmatrix} 0.95 & 0 \\ 0.05 & 1 \end{pmatrix}$$

Improvements obtained with these matrices are shown in Figure 6. We will refer to the model trained on `Tr3` with H_{Tr3} (resp. `Tr1` with H_{Tr1} and `Tr2` with H_{Tr2}) as `perfTr3` (resp.`perfTr1` and `perfTr2`).

3.3. Varying the threshold

The basic threshold applied above is a simplifying model, as it will mislabel the pictures that were outliers for the old classifier, introducing the same type of error during the learning phase. We propose here different ways to handle these errors.

As shown in Table 3, we select thresholds based on the percentage of labels 1 that we want to have in our training set. We observe in Figure 7 that the network performances degrade if the threshold is set high, as the numbers of pictures in each class are unbalanced. The best threshold seems

Threshold	Corresp. score	#with glasses	*AUC*
2.5%	41.5	9549	0.934
5%	26.7	19098	0.938
7.5%	21.2	28647	0.954
10%	18.4	38195	0.944

Table 3: Repartition and *AUC* for each threshold.

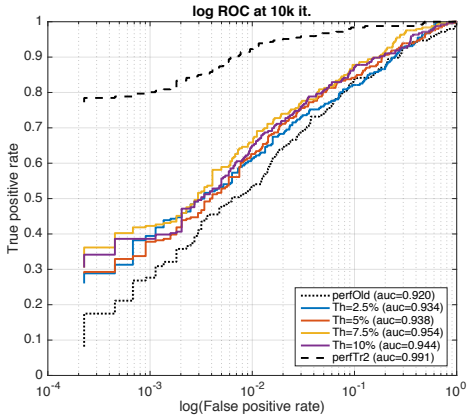


Figure 7: Impact of the binarization threshold (ROC-log).

to lie a bit above the empirical threshold of 5%, corresponding to the repartition observed in the hand-labeled datasets. At 7.5%, we allow for images with glasses that were given a lower score by the old classifier to be correctly identified by the CNN. This is also an indication that a step of data augmentation in order to balance the two classes could improve the performance. A classical approach to balance the dataset would be to duplicate some of the positive class images (and possibly add some noise on the duplicate versions to diversify the dataset).

3.4. Introducing a margin

We furthermore introduce a notion of rejection margin around the threshold, where we discard all images that have a score s such that $|s - t| \leq m$. The rationale is that for these pictures, the score from the old classifier is unsure and should thus be ignored. We select a binarisation threshold of 5% in order to stay close to the observed distribution of the true labels. When the margin gets larger, the size of the training set decreases, an effect that has consequences on the performances if this size becomes too small: once again the two classes are unbalanced, and the outliers mis-labeled by the old classifier have more influence on the results.

Table 4 shows that as the margin increases, the general accuracy decreases. But depending on the operating point (Figure 8), it is possible to get better results with a margin defined by $m = 15$, where many uncertain pictures have been rejected, but at the same time the dataset remains of a sensible size.

m	#images	#with glasses	r	Accuracy at 10k it.	*AUC*
0	381949	19086	4.9%	0.9722	0.952
5	368956	14565	3.9%	0.9713	0.953
10	347032	11551	3.3%	0.9711	0.949
15	286729	9470	3.3%	0.9680	0.962
20	107152	7851	7,3%	0.8759	0.941
25	6630	6587	98%	0.0878	0.699

Table 4: Repartition of labels and results by margin.

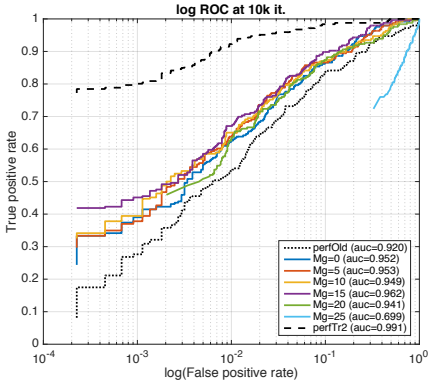


Figure 8: Impact of different margins (ROC curves).

The methods studied in the two previous experiments were mainly focused on addressing the issue of noise through preprocessing of data, either by using a small hand-labeled subset, or by binarizing the scores of the old classifier based on the statistics of the `CASIA` dataset. Meanwhile, the learning phase in itself was similar to a standard CNN training. In the next tests, we try to take the noise into account directly during the training phase, allowing for a better handling of those uncertain labels.

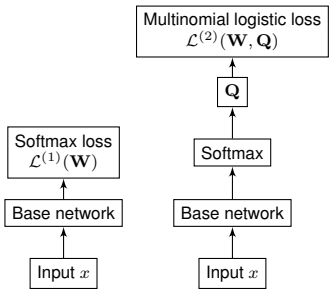
4. Training on noisy labels with a confusion matrix

Sukhbaatar et al. [11] introduced an extra noise layer into the model which maps the network outputs ($\mathbb{P}(y=1)$ and $\mathbb{P}(y=0)$) to the noisy label distribution. We implemented an adapted version of their method in *Caffe* that 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.

4.1. Noise modeling

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

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



(a) Phase 1 (b) Phase 2

Figure 9: Model outline.

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:

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

4.2. White noise - Case study

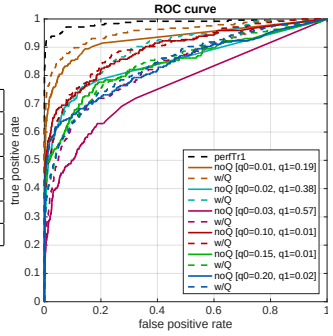
We introduce synthetic noise on the labels of `Tr1` with different levels q_i on each class then train the baseline model (Figure 9a) denoted noQ, as well as the model with the extra noise layer (Figure 9b) denoted withQ on the noisy data. We can either learn $\mathbf{Q}$ in the second phase or simply infuse the true matrix evaluated on a clean subset of the training set. We tested both strategies:

Learning Q: The learnt confusion matrix gets closer to the true matrix for the class 1, while often converging to $(1, 0)^T$ for class 0 given the skewed distribution of the labels.Thus, the accuracy is weakly affected by the introduction of learnt $\mathbf{Q}$.

Evaluating Q on a subset: the introduction of the extra noise layer yields promising results as it can cope with high noise levels especially on the positive class (up to 57%), but with high noise levels on the majority class, the confusion matrix fails to handle the noise (Figure 10).

q_0	q_1	noQ	withQ
0.01	0.19	0.93	0.96
0.02	0.38	0.85	0.90
0.03	0.57	0.75	0.83
0.1	0.01	0.90	0.90
0.15	0.01	0.84	0.85
0.2	0.02	0.84	0.84

(a) *AUC* without/with Q matrix



(b) ROC without/with Q matrix.

Figure 10: Impact of the confusion matrix.

$$Q_1 = \begin{pmatrix} 0.98 & 0.02 \\ 0.08 & 0.92 \end{pmatrix}$$
$$Q_2 = \begin{pmatrix} 0.95 & 0.05 \\ 0.21 & 0.79 \end{pmatrix}$$
$$Q_3 = \begin{pmatrix} 0.997 & 0.003 \\ 0.40 & 0.60 \end{pmatrix}$$
$$Q_4 = \begin{pmatrix} 0.995 & 0.005 \\ 0.54 & 0.46 \end{pmatrix}$$

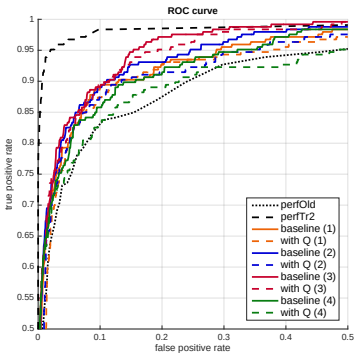


Figure 11: Image dependent noise: effect of Q-matrix.

4.3. Image dependent noise

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. As the existing classifier is based on gradient information strongly correlated with the glasses frame, the independent noise assumption is not valid.

To challenge the confusion matrix method, we evaluate it with different choices of data binarization for the training base `Casia`: given (t_0, t_1), an image is labeled 0 (resp. 1) if the score is smaller than t_0 (resp. larger than t_1), as show in Table 5. We estimate the $\mathbf{Q}$ matrix on the annotated subset `Tr1` (on images for which the score does not belong to $[t_0, t_1]$) for each choice of binarization (Figure 11) and use the full training set `CASIA`. Although the estimated

Test	t_0	t_1	#images	#with glasses	r
1	10	30	250341	15827	6.32%
2	20	20	381916	32132	8.41%
3	30	30	381942	15827	4.14%
4	40	40	381947	10079	2.64%

Table 5: Binarizations used to evaluate the confusion matrix method.

matrices are quite similar to the white noise matrices (low q_0 and large q_1), in this context of image dependent noise, the method fails to improve the classification performances (Figure 11). The conclusion is that the white noise hypothesis is far from what is real on our training dataset. The error of the old classifier is highly dependent on the considered image, which implies to use a more clever way to model the noise in order to tackle with it.

5. Bootstrapping methods

5.1. Bootstrapping a balanced training set

Another way to improve the CNN classification with noisy data is to use bootstrapping, as it has been proposed in [10]. But instead of relabeling the images in a dynamic way at each iteration, we suggest here to learn a new model at each iteration as follows:

1. the CNN is trained on a subset of `CASIA`
2. the resulting network is then used to output new scores for the pictures of the training set.
3. the training set is re-labeled by averaging the previous scores ans the current model probability (Figure 12).

We work on a subset of 100k pictures of `CASIA`, balanced at 20% of positive labels at start, as this balancing has been shown to provide a better baseline in previous experiments.

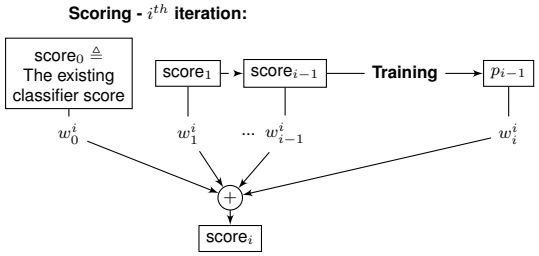


Figure 12: Relabeling strategy.

At iteration i : $[\forall j \in (0, ..., i),\ w_j^i = \frac{1}{i+1}]$: we consider all previous scores and the current CNN score to relabel the samples to stabilize the relabeling. Figure 13 shows that the performances are fluctuating but are improved with respect to the initial performances. As shown in Figure 14, some

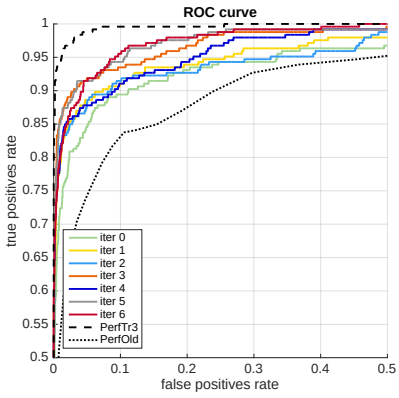
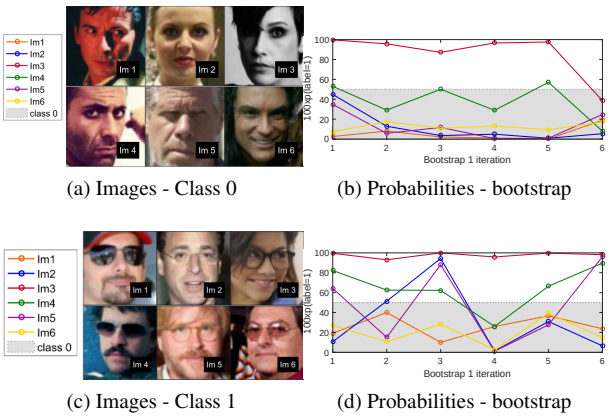


Figure 13: Bootstrap performances starting from the initial classifier annotation.

samples from the training set are difficult to label and multiple bootstrap iterations do not reveal the true label: the bootstrap scores are either fluctuating around the decision threshold or completely outside the desired domain.

5.2. Bootstrapping starting from a clean CNN

Instead of relying on the scores of the old classifier for the initial labeling, we propose here to use our best classifier (`PerfTr3`) to perform the initial annotation over *Casia*,



(a) Images - Class 0 (b) Probabilities - bootstrap

(c) Images - Class 1 (d) Probabilities - bootstrap

Figure 14: Bootstrap - samples classification over iterations.

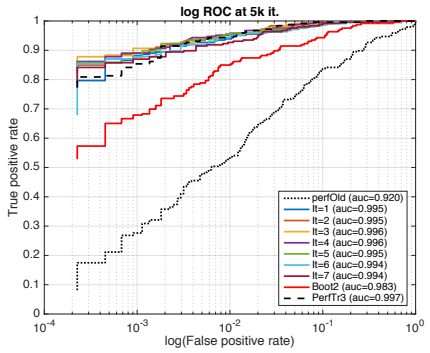
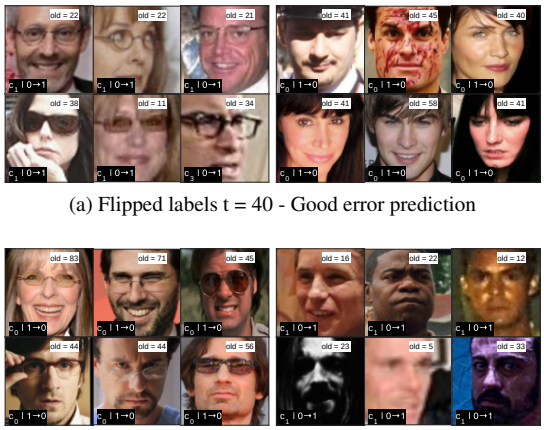


Figure 15: Bootstrap performances starting from the clean CNN classifier.

before bootstraping with the Bootstrap method described above. As this initial labeling is much better than the old one, the performances obtained are improved with respect to the previous section. Overall, the *AUC* remains close to the initial performance of `PerfTr3`. Even if the gain in performance is small in absolute, compared to the advance obtained when starting the bootstrap with noisy labels (curve *Boot2*), we still get a significant relative increase in performances compared to the `PerfTr3` results. On Figure 15, we see that for small false positive rates, the improvement brought by the bootstrap over the simple `PerfTR3` CNN is significative (a relative gain around 30%).

6. Noise prediction

As we know that our training database presents label mistakes, one preliminary step to the learning process can be to predict the possible errors in our data. This has been introduced in [1] and can be used to filter or correct the database. Instead of considering directly our binary classification problem, we distinguish an intermediate problem with 4 different classes given the ground truth label of an image and the old classifier score. We train a CNN that takes the image as an input, and output a label $i \in (0, 1, 2, 3)$ that expresses if this picture would be a true positive, false positive, true negative or false negative when using the old classifier on it. During the supervised training phase, both the ground truth and the old classifier results are used to generate the labels i (Table 6).



(a) Flipped labels t = 40 - Good error prediction

(b) Flipped labels t = 40 - Wrong error prediction

Figure 16: Labels flipping predicted by the 4-class CNN.

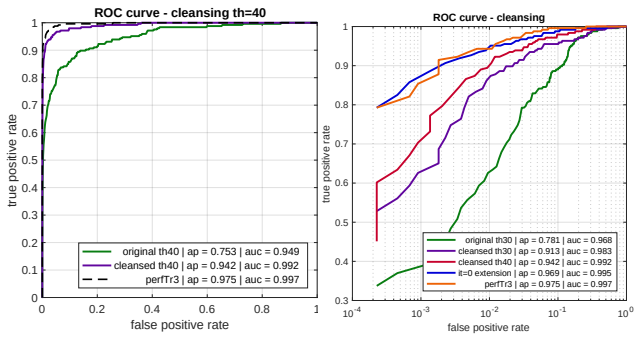
The CNN classifier, trained on a hand labeled dataset (`Tr1`), will help uncover the noisy labels in order to train a robust model on the cleansed labels. We tested this method with $t = 30$ and $t = 40$ but limit the detailed results for the second case, where the dataset has the following properties:

class	label	# training	# test
0	noisy=true=0	3919	4394
1	noisy=0 true=1	116	134
2	noisy=1 true=0	19	21
3	noisy=true=1	98	112

Table 6: Datasets statistics when thresholding at t=40.

Figure 16 illustrates some examples of classification, reflecting the predicted error of the old classifier. The model is unable to expose all false positives, which is quite expected seeing the dataset skewness, and most samples of class 2 end up in class 0. In the misclassified images (in the sense where the corrected label is different from the ground truth label) we notice that the same issues encountered before persist, namely the difficulty to detect rimless glasses or handle unusual poses.

On the bright side, we reach an improved accuracy with the model learned from cleansed labels as shown by the following experiment. After rectifying the labels, we run the baseline on the full training set `CASIA`. Although thresholding at $t = 40$ gave poor performances compared to other thresholds (c.f. Section 3.3) it's more prone to noise



(a) t=40 (b) comparison log-scale

Figure 17: Baseline model learnt on cleansed vs. noisy labels. Curve 'it=0 extension' is the first bootstrap iteration when labeling the full training set with `perfTr3`. It can be seen as cleaning the labels with a binary classifier trained on clean labels (`Tr3`).

prediction with more false negatives that the model has proven to deal with successfully and less false positives harder to unmask (Figure 17). On the left, we see the improvment obtained by relabeling the initial labels given the noise prediction classifier, increasing from $AUC = 0.949$ to $AUC = 0.992$.

Instead of learning the suggested 4-class classifier, another option could be to learn two binary classifiers, one for images labeled as positive, and another for images labeled as negative, as we benefit of the old classifier result as input. These two classifiers may be simpler to train and lead to even better label correction. This experiment has not been tested yet but will be explored in the future.

7. Conclusion

In this paper, we studied multiple methods to handle noisy labels in training databases, from pre-processing the data to modifying the CNN learning phase. Overall, the main solution performance-wise remains to use a hand-labeled small subset: one of our best results was obtained by manually labeling less than 2% of the CASIA dataset. Using a tremendous amount of data but with noisy labels could not compete with a small dataset with clean labels. However, classical performance enhancement techniques such as bootstrapping have shown to be very efficient on both clean and noisy datasets, improving accuracy and precision of the network trained on clean data. Models that take into account (either in the classifier or via another CNN) the dependence between the noise and the content of the pictures in more refined ways are still limited to cases constrained by heavy assumptions on the noise type and distribution, but remain promising; literature on this topic will surely be expanded in the coming years, as research progresses. When focusing on usability of the studied methods, aside from the manual hand-labeling option reinforced by bootstrapping techniques, the experiment of label error prediction by CNN appears as extremely promising. This overview of methods have been applied on a simple problem (binary classification) but with a image-dependent noisy database which is complicated to handle. Extension on multi-class or regression would be interesting to deepen, starting with simpler noise modeling, as we showed that some methods handle it efficiently.

Appendix: Hyperparameters of the CNN

base_lr: 0.01	power: 0.75	momentum: 0.95
lr_policy: "inv"	gamma: 0.0001	weight_decay: 0.0

References

[1] C. E. Brodley and M. A. Friedl. Identifying Mislabeled Training Data. *CoRR*, 2011. 7

[2] D. Flatow and D. Penner. On the Robustness of ConvNets to Training on Noisy Labels. Technical report, Stanford University, 2015. 1

[3] B. Frénay and A. Kabán. A Comprehensive Introduction to Label Noise. In *European Symposium on Artificial Neural Networks*, 2014. 1

[4] B. Frénay and M. Verleysen. Classification in the Presence of Label Noise: A Survey. *IEEE Trans. Neural Netw. Learning Syst.*, 25(5):845–869, 2014. 1

[5] J. Ge, Y. Xia, and C. Nadungodage. *Conference on Advances in Knowledge Discovery and Data Mining*, chapter UNN: A Neural Network for Uncertain Data Classification, pages 449–460. Springer Berlin Heidelberg, 2010. 1

[6] 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 *ACM Multimedia*, pages 675–678, 2014. 2

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*. 2012. 1

[8] J. Larsen, L. Nonboe, M. Hintz-Madsen, and L. K. Hansen. Design of Robust Neural Network Classifiers. In *Int. Conf. on Acoustics, Speech and Signal Processing*, 1998. 1

[9] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *NIPS*. 2013. 1, 3

[10] S. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich. Training Deep Neural Networks on Noisy Labels with Bootstrapping. *CoRR*, 2014. 2, 6

[11] S. Sukhbaatar and R. Fergus. Learning from Noisy Labels with Deep Neural Networks. *CoRR*, 2014. 1, 2, 5

[12] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from Massive Noisy Labeled Data for Image Classification. In *CVPR*, June 2015. 1

[13] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning Face Representation from Scratch. *CoRR*, 2014. 2

[14] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial Landmark Detection by Deep Multi-task Learning. In *ECCV*, pages 94–108, 2014. 2