Glasses detection using CNN
ECP 3rd year project

Maha Elbayad
maha.elbayad@student.ecp.fr

Simon Rodriguez
simon.rodriguez@student.ecp.fr

Supervisor:
Catherine Herold, Morpho

ECP Supervisor:
Nikos Paragios, CentraleSupelec

Abstract

The performance of Convolutional neural networks depends on the amount of labeled samples and their presumable quality, i.e. having correct labels that are suited to the tackled problem. Since hand labeling is impractical on large databases, a shift toward semi-automatic labeling is inevitable, which means less accurate and sometimes misleading data that must be handled differently. We consider here the problem of binary classification with respect to the presence or not of glasses on a face. In this context, we will have to handle the presence of image dependent noise (labels are obtained by an existing imprecise classifier) and we attempt to assess the noise level present in the training set during the preprocessing and/or while learning.

1 Introduction

With the rise of deep learning and the improved ability to build huge datasets extracted from multiple sources, CNN classifiers can be designed and trained to perform complex tasks. Their accuracy and precision improve upon what could be achieved with conventional machine learning methods. Convolutional neural networks have shown to be surprisingly efficient on complex pattern recognition tasks, notably image classification [1]. But as the datasets grow bigger, it becomes harder to have ground-truth labels associated with the training data. 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 is 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[2], and can disturb or decrease the performances of the classifier [3]. Handling this noise in a supervised or semi-supervised setting is a currently active field of research due to the reasons presented above. Most of the methods studied can be separated in two groups: preprocessing of the noise through filtering of the dataset, and accounting for the noise in the structure of the classifier itself [4].

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 [5]. 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 dataset is 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, for example: uncertain neural network for uncertain data classification called UNN[6],

surrogate losses functions that distinguish between samples taken from a hand-labeled subset and samples with noisy annotations [7] 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 [2]. The introduction of a confusion matrix, either designed from observations on the dataset or learned during the training phase of the classifier [8], 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 [9] or dependent on the true label but conditionally independent with the image[8] 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 especially 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].

In the first two sections of this report we lay out our general experimental setting and the chosen baseline model. In section 4 we list our evaluation criteria and analyze some dataset statistics that might interfere with the model performance before describing the methods we investigated in the course of this project: first with two major preprocessing methods, limiting the noise level through dataset preprocessing (section 4.3) and noise cleansing with a classifier trained on a small clean dataset (section 4.6). Then, from the second group of methods we focused on the confusion matrix technique[8](section 4.4) before introducing some bootstrapping schemes (section 4.5).

2 Setting

2.1 Caffe

We use the Caffe framework [11] developed by the Berkeley Vision and Learning Center to train and test our neural networks. The framework relies on a pure C++/CUDA architecture, allowing for good performances when run on GPU. Bindings with Python and Matlab are provided but we generally used the direct command-line interface, as the main learning phase took place on the Ecole CentraleSupelec computing center, on a node equipped with two 12 cores Intel Xeon E52695v2 (2.4Ghz) and a Nvidia Tesla K40 GPU. Caffe relies on a text file based definition for networks and configuration, allowing us to quickly define multiple tests.

2.2 Dataset

We experiment with the CASIA WebFace database [12] containing 380,000 face images of about 10,000 subjects semi-automatically collected from the internet, mainly the IMDB website. We hand-labelled 4661 images from the CASIA database for validation (denoted `Test`) and another 4152 images for clean training (denoted `Tr1`) the rest (denoted `CASIA`) is labelled with an glasses detector based on edges information, especially the presence of the bridge of the glasses between the two eyes. The scores of the detector range from 0 to 255 and their distribution given the true label of an input image on manually annotated databases is shown in figure 1. (Those databases being confidential, we cannot display examples of pathological cases where images are mislabelled.)

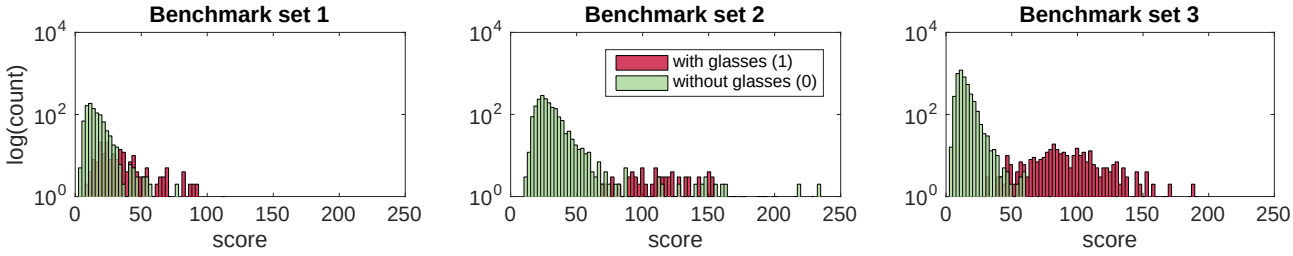


Figure 1: Scores distribution on benchmark datasets

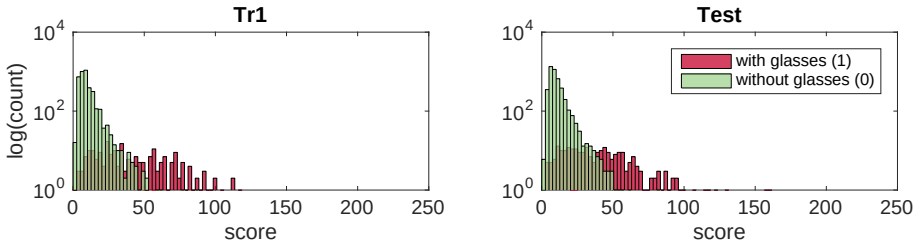
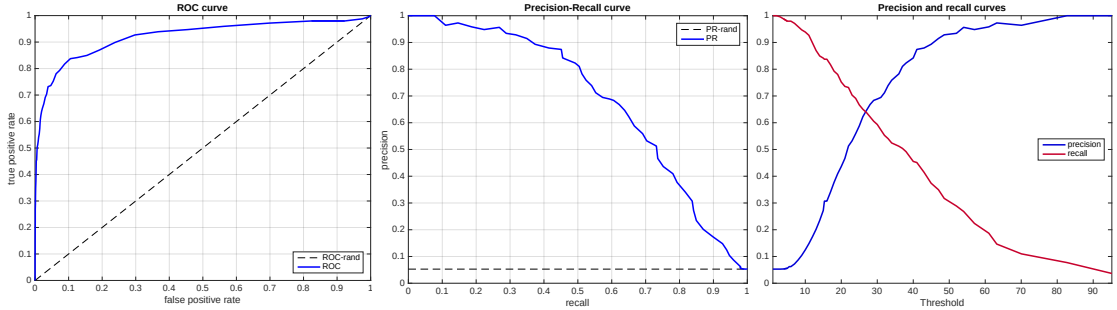


Figure 2: Scores distribution on subsets Tr1 and Test 1

We can evaluate the performance of the old classifier, using indicators that will be described in further details in section 4.1. If we threshold the scores obtained on the test set via the old classifier at some point t we can assess the classifier performance. When varying the threshold t we notice that we can balance the trade-off between precision and recall at a threshold of 26 (figure 3c). At this exact threshold, 5% of the images from `CASIA` would be considered positive (with glasses). This ratio of 5% is also obtained on randomly sampled subsets from `CASIA` that we hand-labeled which means that our dataset is initially skewed and unbalanced.

In the following sections we will refer to the performance of the existing classifier as `perfOld`.



(a) ROC curve (b) Precision-Recall curve (c) Precision & recall curves

Figure 3: Performance of the existing classifier

We enriched our training set with 4151 images[1] manually labelled (denoted `Tr2`) that have higher image quality and less compression artifacts with a positives ratio of 20%. We denote with `Tr3` the fusion of `Tr2` with a balanced subset of `Tr1` (1070 images) to maintain the 20% positives ratio. The datasets statistics are shown in table 1, namely the dataset size, the number of its positive samples (with glasses) and the positives ratio, that is the percentage of the positive samples (with glasses) to the whole dataset, denoted by r.

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

Table 1: Datasets statistics

3 The baseline model architecture

Our model is inspired by the Alexnet architecture and the CNN architecture described in [13] and refined in [14], with 3 blocks of convolution, ReLU and max-pooling layers followed by 2 fully-connected layers

[1] courtesy of the multimedia lab in Hong Kong (`http://mmlab.ie.cuhk.edu.hk/projects/TCDCN.html`)

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 $\mathcal{X} = \{x_n, y_n\}_{1\leq n\leq N}$:

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

We denote with **W** the weights of the network. The regularization is assured with dropout on the fully-connected layers. The hyperparameters of the architecture are summarized in table 2.

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

Table 2: Baseline model hyperparameters : N=Number/batch size, C=Channels, W=Width, H=Height

The general learning hyperparameters (learning rate, weight decay) were defined at the beginning of the project with the supervising team at Safran Morpho. In fact, the goal of the project was never to fine-tune a CNN or explore the space of its hyperparameters, thus those values were fixed once and for all (see 8.1).

4 Experiments & Results

During this study, we tried to follow two directions in our research and experimentations: the first one was to get the best classification possible, even if this implies to hand-label a subset of the initial dataset. This will be addressed in the first experiment, where we will show that this methodology can indeed lead to the best performances. The second axis aimed to take into account the fact that some labels are noisy. Even if the performances obtained on a noisy dataset are initially lower, a wider range of methods (as described in the first section) can be applied to improve the scores and bring interesting results.

4.1 Evaluation criteria

We start by presenting the main indicators we will rely on to assess the performances of our experiments. All our evaluators are using the `Test` dataset, and are based on the comparison between the ground truth labels and the scores output by the CNN on this set. For all the following definitions we consider the notations in the table to the right.

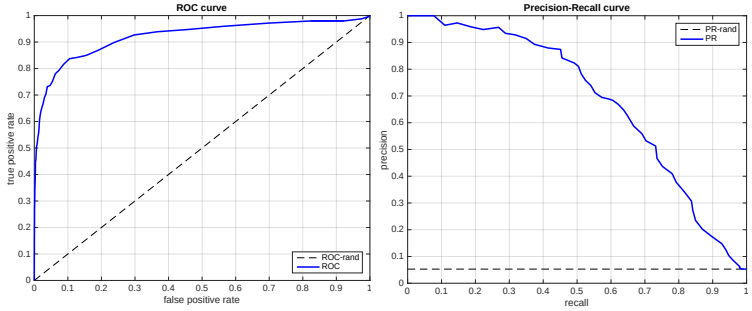
		prediction	
		1	0
ground truth	1	tp	fn
	0	fp	tn

- **Accuracy:** The accuracy is the simplest indicator, defined as the ratio of the number of correct predictions over the number of samples tested. It only makes the distinction between erroneous and correct predictions, and is thus a very global indicator, that furthermore relies on the necessity to define a threshold on the scores given by the classifier ($accuracy = \frac{tp+tn}{tp+tn+fn+fp}$).
- **Confusion matrix:** The confusion matrix enhance the concept of accuracy by discerning four cases: the true positives (tp), the true negatives (tn), the false positives (fp) and the false negatives (fn). It allows one to better grasp the behavior of the classifier ; depending on the desired

application, a classifier that makes more false positives than false negatives can for instance be preferred.

- **Receiver Operating Characteristic:** The ROC curve expresses the true positive rate, also called the sensitivity, equal to $\frac{tp}{tp+fn}$, as a function of the false positive rate, also called the fall-out, i.e. $\frac{fp}{fp+tn}$. Each point of the curve is obtained by setting a threshold on the scores of the classifier (i.e. the output of the softmax layer of the CNN) and computing the corresponding ratios.
  Any ROC curve will pass through the points (0,0) and (1,1) as they corresponds to the points where all samples are negative/positive respectively. A random classifier will have the identity line as a ROC curve, while a perfect classifier ROC is composed of the lines from (0,0) to (0,1) and from (0,1) to(1,1).
  By choosing a false positive rate corresponding to the behavior of the desired classifier, one can assess how well it performs at this given rate.
- **Area Under the Curve:** The area under the ROC curve is equal to the probability that the classifier will predict a higher label for a randomly chosen image labelled 1 than for another randomly chosen image labelled 0.
- **Precision-Recall:** The PR curve expresses the precision (the number of ground truth positive correctly defined as positive by the classifier, divided by the total number of images labelled as positive by the classifier) as a function of recall (the number of ground truth positive correctly defined as positive by the classifier, over the total number of images that really are labelled positives). A higher precision implies that the classifier gives positive labels in a relevant way (not too many false positives), while a high recall means that the classifier detects positives more efficiently (not too many false negatives).
  Using the notations of the table above, we have that precision $= \frac{tp}{tp+fp}$ and recall $= \frac{tp}{tp+fn}$.
- **Average precision:** This is the area under the PR curve. It gives a general overview of the PR behavior of the classifier.

In the figure below the performances curves of the original classifier are shown, along with the numerical evaluators described above.



(a) ROC curve (b) Precision-Recall curve

Figure 4: Performances of the existing classifier:
Accuracy=0.962, AUC=0.920, Average Precision=0.679

4.2 The effects of the training set statistics

4.2.1 The positive samples ratio

We train our model on the sets `Tr1`, `Tr2` and `Tr3` respectively and evaluate each model on the `Test` set. `Tr1` share the same distribution as the test set ($r \approx 5\%$) while `Tr2` have more positive samples ($r \approx 20\%$) and so does `Tr3` with more samples to train on. The results in figure 7 show that the CNN underperforms when the training data is skewed, i.e. the two classes are unbalanced (the performance of `Tr2` compared

to `Tr1`), as it tends to promote the more frequent class. The fact that `Tr1` and `Test` are drawn from the same sample may favor training with `Tr1` but balancing the training set by means of under-sampling [15] or optimizing a class-sensitive cost [7][16][17] yields better results.

We approached the cost sensitivity by training our baseline model with a different loss function built on the cost matrix $\mathbf{C}$, where:

$$\mathbf{C} = \begin{pmatrix} c_{00} & c_{01} \equiv c_{FN} \\ c_{10} \equiv c_{FP} & c_{11} \end{pmatrix}$$

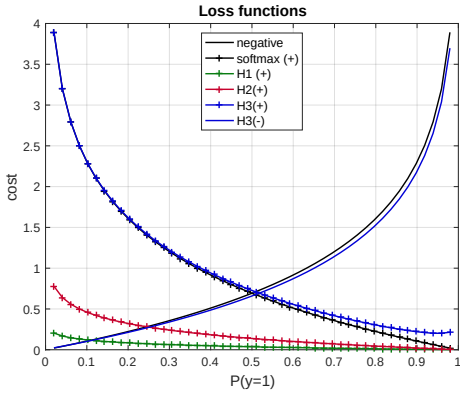
$c_{i,j} \equiv$ the cost of predicting i when the ground truth is j.
To implement this new loss function in *Caffe* we adapt the already existing `InfoGainLoss` layer which inherits from the multinomial logistic loss (1):

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

To achieve proportional misclassification costs to the ones defined in $\mathbf{C}$ we set:

$$H = \begin{pmatrix} c_{01} & c_{00} \\ c_{11} & c_{10} \end{pmatrix}$$

On figure 5, c_{01} is the cost of a false negative, the image of $\mathbb{P}(y = 1) = 0$ on the positive curve (to the extreme left) and c_{10} is the cost of a false positive, the image of $\mathbb{P}(y = 1) = 1$ on the negative curve (to the extreme right).
Intuitively, we would want for the misclassification costs c_{10} and c_{01} to satisfy $c_{10}\mathbb{P}(y = 0) : c_{01}\mathbb{P}(y = 1)$ and for the correct classifications to have a zero cost. $H1$, and to a lower extent $H2$, were defined with this objective in mind. $H3$ takes the opposite direction with a false negative cost slightly lower than the false positive cost.

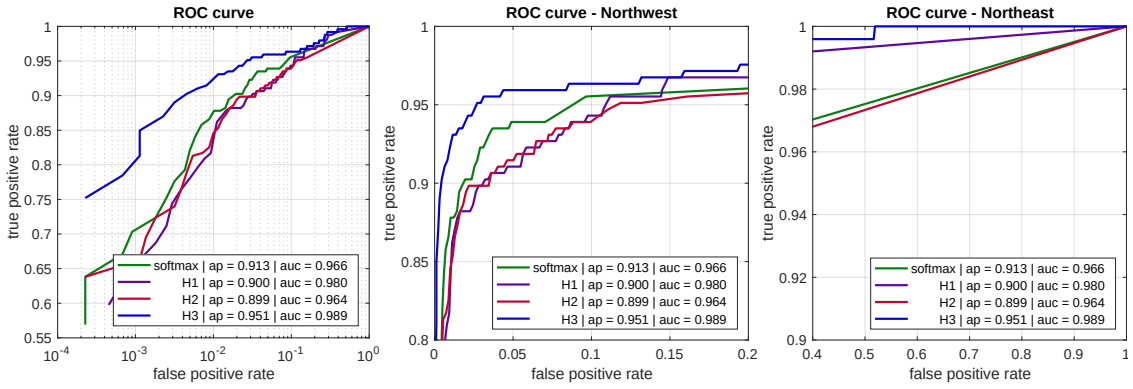


$$H1 = \begin{pmatrix} 1 & 0 \\ 0 & 0.05 \end{pmatrix}$$

$$H2 = \begin{pmatrix} 0.95 & 0 \\ 0 & 0.2 \end{pmatrix}$$

$$H3 = \begin{pmatrix} 0.95 & 0 \\ 0.05 & 1 \end{pmatrix}$$

Figure 5: Loss function per cost matrix



(a) Log ROC (b) ROC zoom-in (c) ROC zoom-in

Figure 6: Roc curves with the 3 different losses compared to the default softmax loss - `Tr1`

The model with $H1$-loss performs marginally better than the softmax model with a lower precision yet on the ROC curve it achieves higher true positives rate (northeast of the ROC figure 6c) i.e. low false negatives rate which is coherent with the higher false negative cost assigned (1 vs. 0.05). $H2$ is slightly equivalent to the softmax model. $H3$ on the other hand performs surprisingly well, one possible interpretation is that most classifiers assign a zero cost to correctly classifying a sample ($c_{00} = c_{11} = 0$), however, with $H3$ the classifier is encouraged to doubt the class of all samples with a lower cost at $p(y = 1) = 0.95$. Such costs have a regularization effect on the model.

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

$$H_{Tr2} = H_{Tr3} = \begin{pmatrix} 0.99 & 0.01 \\ 0.2 & 0.8 \end{pmatrix}$$

$$H_{Tr1} = \begin{pmatrix} 0.95 & 0 \\ 0.05 & 1 \end{pmatrix}$$

We will refer to the model trained on `Tr3` with H_{Tr3} as `perfTr3` and we may refer as well to the best performance achieved on `Tr1` as `perfTr1` and that on `Tr2` as `perfTr2`.

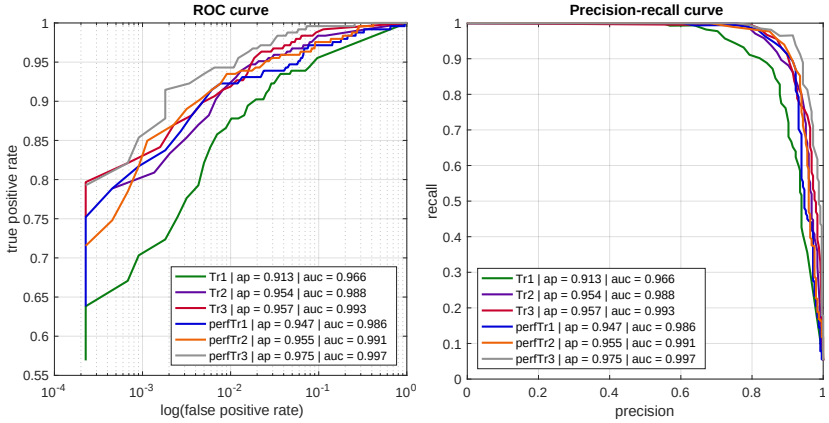


Figure 7: Best performance on each training set

4.2.2 The training set size

In this section we experiment with varying the training set size to determine the optimum size necessary to achieve better performance in two different modes: with clean labels and with noisy labels induced from the existing classifier.

All the models below are trained with the softmax default loss described in section 3. For the clean datasets, we start with `Tr3` and we keep decreasing the dataset size iteratively in order to build nested training sets.

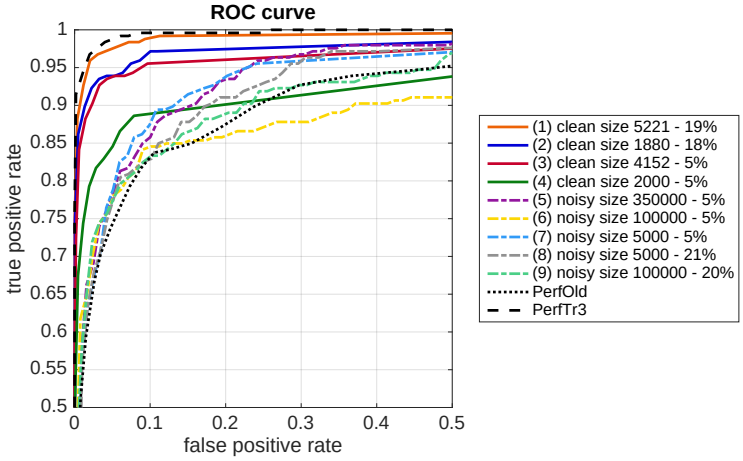


Figure 8: A selection of roc curves with different sizes, modes (clean/ noisy) and ratios r

We note that the training set size is more impactful when the labels are clean, in fact:
Clean labels:

- The average precision and the area under the ROC curve (AUC) strongly increase with the dataset size at fixed positives ratio : $r = 20\%$: (2) $\rightarrow$ (1) and $r = 5\%$: (4) $\rightarrow$ (3).
- The average precision and the area under the ROC curve (AUC) increase with the ratio of the observed positive samples at a fixed size : size ≈ 2000: (4) $\rightarrow$ (2).

Noisy labels:

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

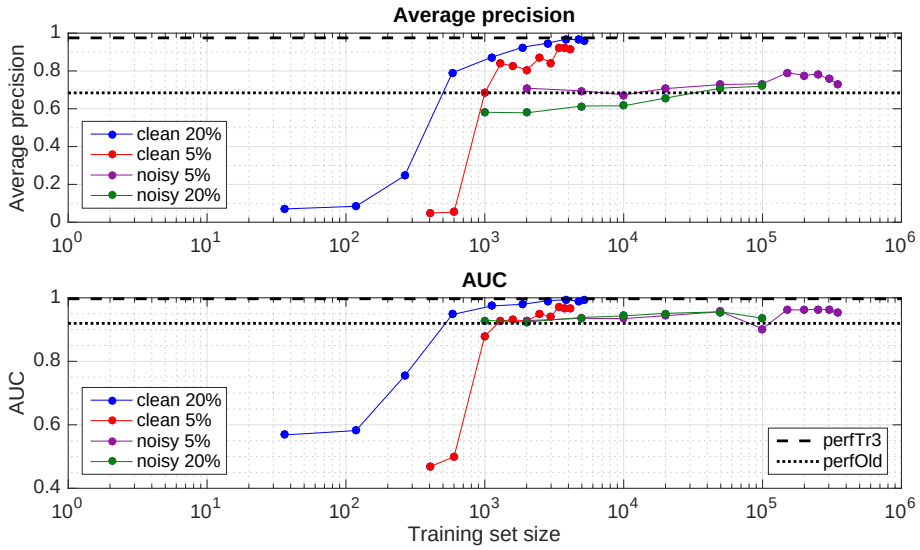


Figure 9: Performance with different sizes and different r: clean vs noisy

4.3 Limiting errors through dataset preprocessing

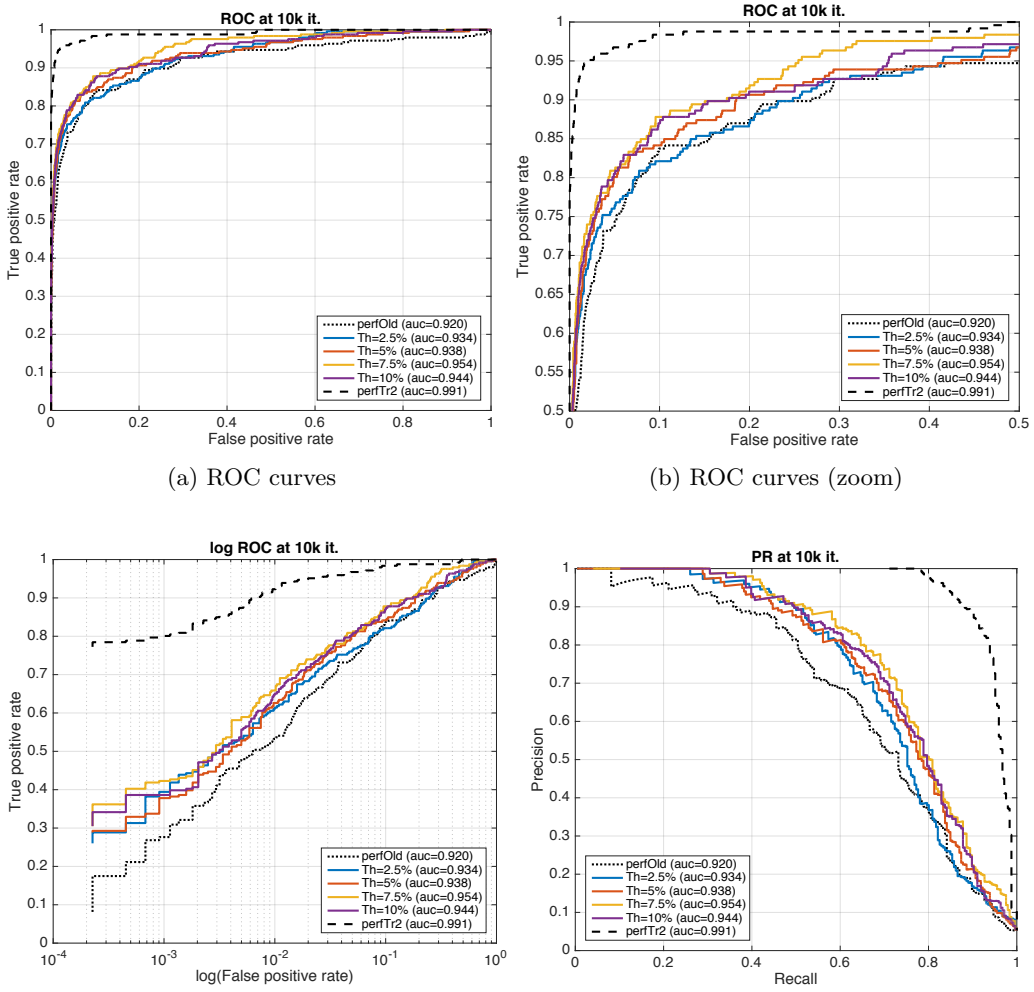
We now experiment with the whole `CASIA` base (minus the `Test` set), to study how the scores from the old classifier can be binarized. We adopt a simple model, where a threshold t is fixed : all images with a score lower than t are labelled 0, all images with a score higher than t are labelled 1. The binarization implies a loss of granularity compared to the raw scores of the old classifier, but is needed due to the structure of the CNN. A basic threshold applied as described 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; for those images, we expect that the picture itself will contain enough information for the CNN to rectify the mistakes.

4.3.1 Varying the threshold

We select thresholds based on the percentage of labels 1 that we want to have in our training set, of total size approx. 381000.

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.



(a) ROC curves

(b) ROC curves (zoom)

(c) ROC log curves

(d) Precision-Recall curves

Figure 10: Impact of the binarization threshold on the classification performances

We observe in figure 10 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 to lie a bit above the empirical threshold of 5% (score=26.7), corresponding to the repartition observed in the hand-labelled 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. The question of the selection of the duplicated pictures based on the noisy labels remains open.

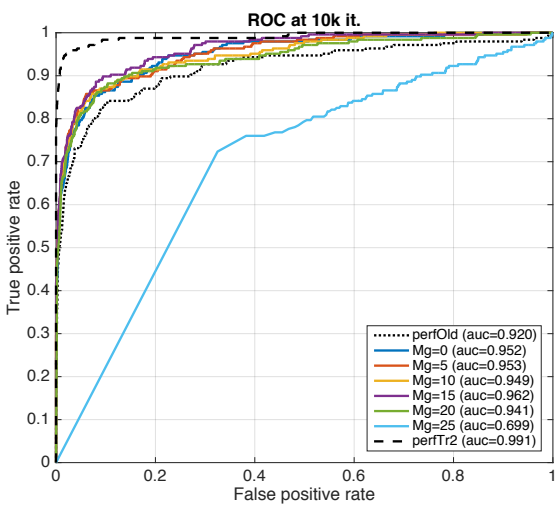
4.3.2 Introducing a margin

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-labelled by the old classifier have more influence on the results.

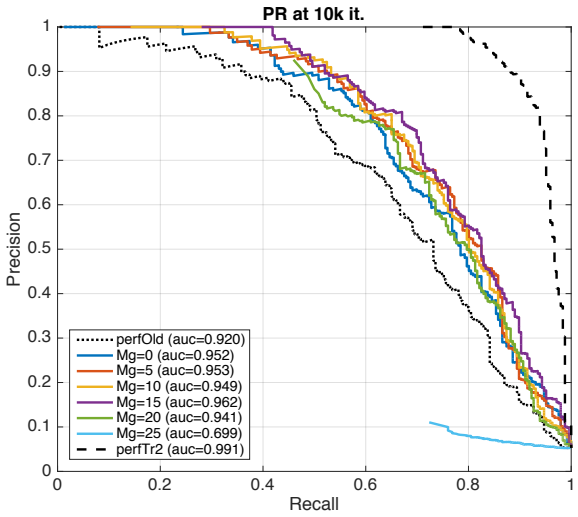
m	#images	#with glasses	r	Accuracy at 10k it.	AUC
0	381949	19086	4.9%	0.9722	0.952
5	368956	14565	3.9%	0.9713	0.953
10	347032	11551	3.3%	0.9711	0.949
15	286729	9470	3.3%	0.9680	0.962
20	107152	7851	7,3%	0.8759	0.941
25	6630	6587	98%	0.0878	0.699

Table 4: Repartition of labels and results for a variable margin

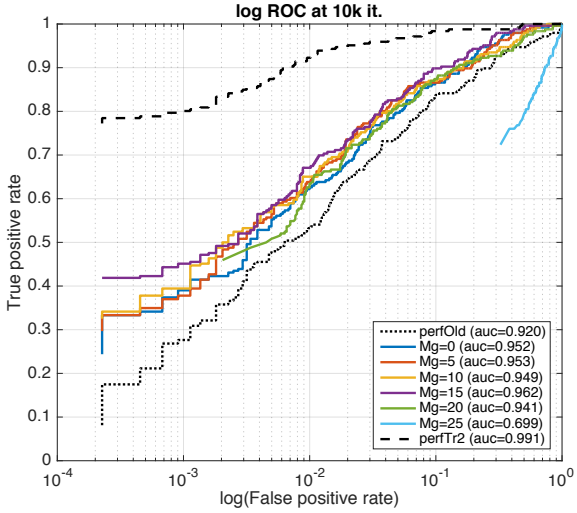
Table 4 shows that as the margin increases, the general accuracy decreases. But depending on the operating point (fig. 11), it is possible to get better results with a margin defined by $m = 15$, where many uncertain pictures have been rejected, but at the same time the dataset remains of a sensible size.



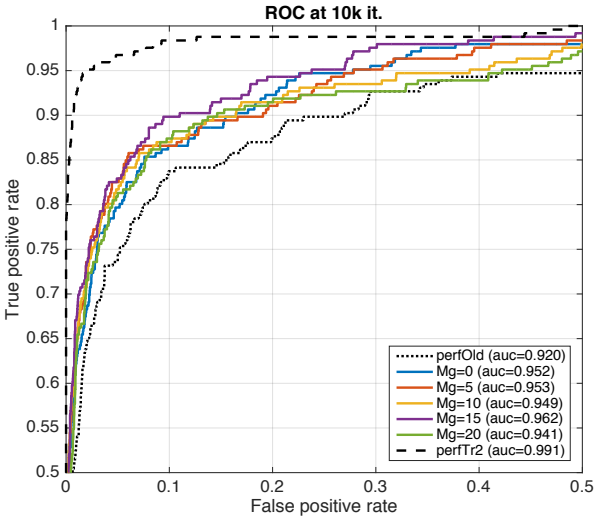
(a) ROC curves



(b) Precision-Recall curves



(c) ROC log curves



(d) ROC curves (zoom)

Figure 11: Impact of the margin width used for labelling on the classification performances

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-labelled 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.

4.4 Training on noisy labels with a confusion matrix

Sukhbaatar et al.[18] 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.

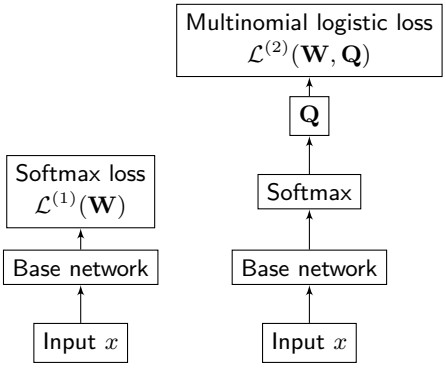
4.4.1 Noise modeling

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:

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

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:

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



(a) Phase 1 (b) Phase 2
Figure 12: Model outline

4.4.2 White noise - Case study

We introduce synthetic noise on the labels of `Tr1` with different levels q_i on each class (0/1) then train the baseline model (fig 12a) denoted noQ, as well as the model with the extra noise layer (fig 12b) denoted w/Q 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 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 (table 5). Thus, the accuracy is weakly affected by the introduction of learnt $\mathbf{Q}$.

definition		Learnt Q		True Q		Learnt Q		True Q		Learnt Q		True Q	
$1-q_0$	q_0	1.00	0.20	0.99	0.19	1.00	0.21	0.98	0.38	1.00	0.06	0.90	0.01
q_1	$1-q_1$	0.00	0.80	0.01	0.81	0.00	0.79	0.02	0.62	0.00	0.94	0.10	0.99
(a)		(b) $q_0 = 0.01, q_1 = 0.19$				(c) $q_0 = 0.02, q_1 = 0.38$				(d) $q_0 = 0.1, q_1 = 0.01$			

Table 5: Confusion matrices: the learnt and the truly used to generate the noisy labels

Evaluating Q on a subset: This time, 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 negatives) the confusion matrix fails to handle the noise.

q_0	q_1	AUC
0.01	0.19	noQ: 0.9282 w/Q: 0.9552
0.02	0.38	noQ: 0.8475 w/Q: 0.8962
0.03	0.57	noQ: 0.7485 w/Q: 0.8323
0.1	0.01	noQ: 0.8927 w/Q: 0.8959
0.15	0.01	noQ: 0.8442 w/Q: 0.8455
0.2	0.02	noQ: 0.8381 w/Q: 0.8437

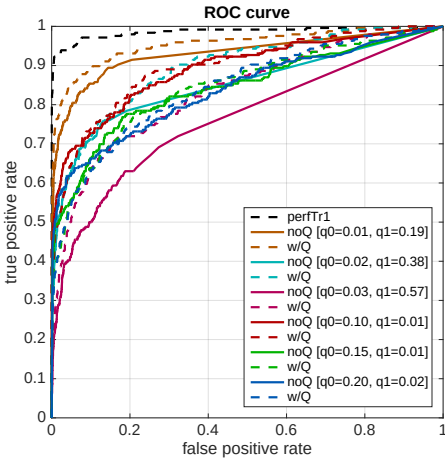


Figure 13: Confusion matrix Roc-curves with white noise

4.4.3 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, which is the case when binarizing the old classifier scores to label the training set; the existing classifier is based on gradient information strongly correlated with the glasses frame (low performance on recognizing rimless glasses) and is considerably sensitive to the contrast changes.

To challenge the confusion matrix method, we binarize the labels in 4 configurations with parameters (t_0, t_1) such that the image is labelled 0 if the score is smaller than t_0 and labelled 1 if it is larger than t_1. We consider the full training set `CASIA` and we estimate the $\mathbf{Q}$ matrix on a subset of the training set (in this case samples from `Tr1` that are still remaining in the training set)

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%

For each training set, the estimated confusion matrix Q_i on $\texttt{Tr1} \cap \texttt{Training\_set}_i$ are shown in figure 14. Although the estimated matrices are quite similar to the white noise matrices (low q_0 and a large of q_1), in this context of image dependent noise, the method fails to improve the classification performance.

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

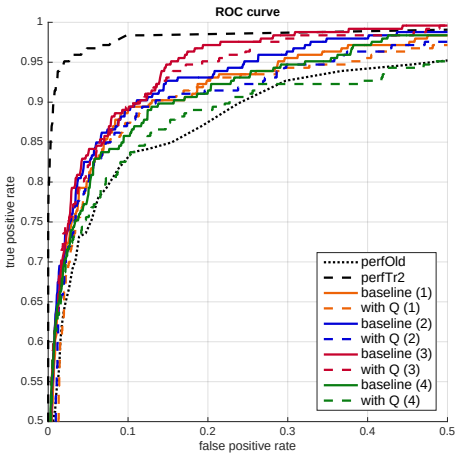


Figure 14: Q-matrix performance on image dependent noise

4.5 Bootstrapping methods

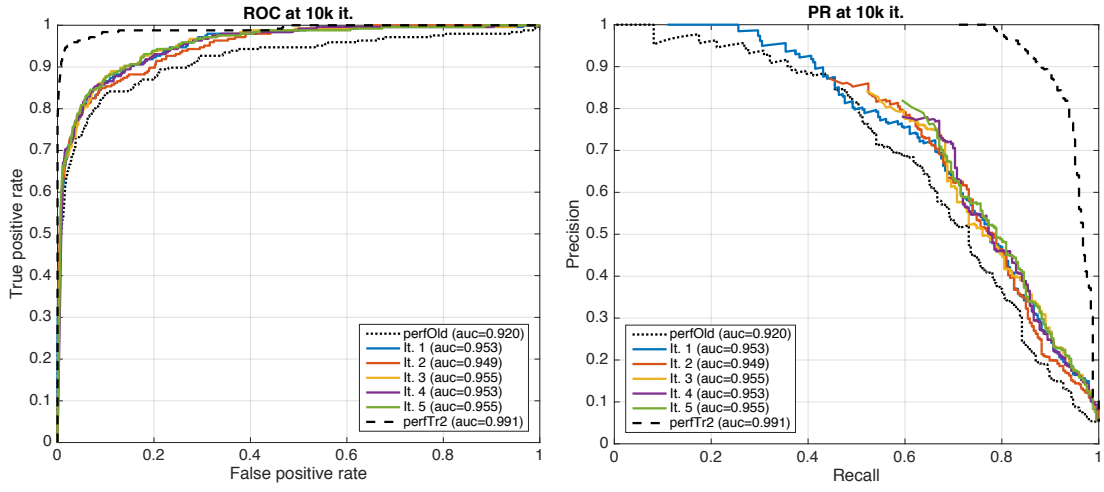
4.5.1 Bootstrapping at a fixed positives ratio

Another method we studied was the process of bootstrapping. We first focused on its simplest form, where:

1. the CNN is trained on a subset of `CASIA` with 20% of positive samples (detailed below).
2. the resulting network is then used to output new scores for the pictures of the training set.
3. the training set is re-labelled based on the new scores, by choosing a threshold that ensures 20% of the pictures are labelled 1.
4. This process is iterated, and we track the evolution of indicators through the experiment.

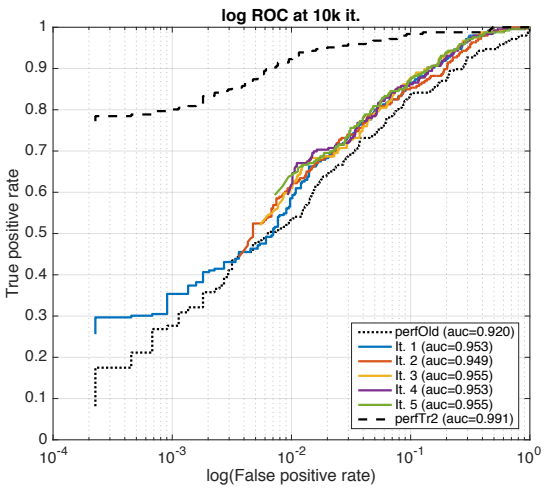
We work on a subset of 100k pictures of `CASIA`, balanced at 20% of positive labels, as this balancing has been shown to provide a better baseline in previous experiments. To build this set, we first binarize the scores of the `CASIA` images using the "natural" 5% threshold described earlier. We then sample 20k images from the pictures labelled as 1 in the previous step, and 80k images from the ones labelled as 0.
The third step was introduced as a way of regularizing the training set: if no ratio was imposed, the number of pictures labelled 1 could decrease with each iteration until the classifier learns to output only 0 labels. From one iteration to the next, we also keep track of the number of labels switched, shown below.
We notice fluctuations from an iteration to the next, with an overall small improvement to AUC and ROC shape. The initial set is probably still too unbalanced to allow for bootstrapping to improve the performance. We see that from an iteration to the next, the number of labels switched is decreasing, implying that the network is converging to some set of parameters. But the gain in performance is small, and calls for the use of finer bootstrapping methods.

	#modified labels wrt previous iteration	AUC
It 0 (old classifier)	0	0.920
It 1	4892	0.953
It 2	1582	0.949
It 3	1184	0.955
It 4	828	0.953
It 5	800	0.955
`perfTr2`	-	0.991

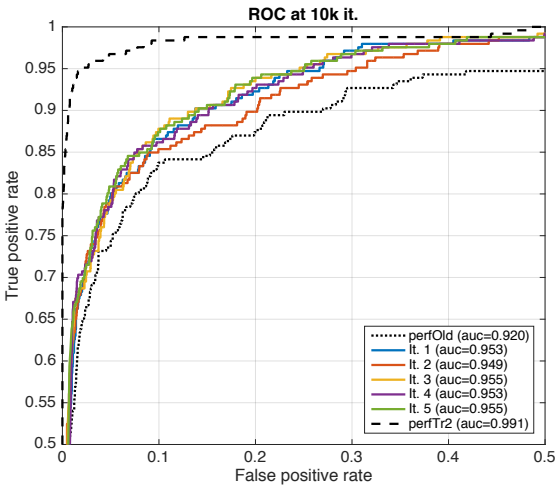


(a) ROC curves

(b) Precision-Recall curves



(c) ROC log curves



(d) ROC curves (zoom)

4.5.2 Bootstrapping a balanced training set

We use the same training set as in 4.5.1 but instead of relabelling at a fixed ratio, we mix the already known training scores with the models' predicted probability and consider their average as a new score. We experimented with a moving average of width 2 and an average of all the previous scores.

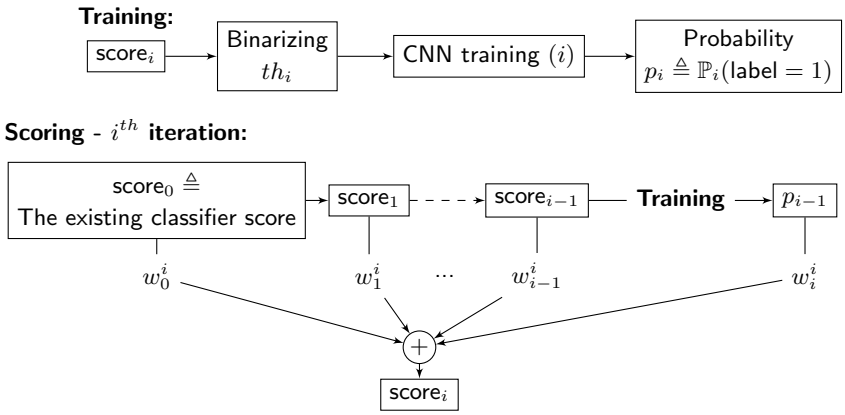


Figure 15: Relabelling strategy

2-moving average: $\forall i \, [w_i^i = w_{i-1}^i = 1/2$ **and** $w_j^i = 0, \forall j \in [\![0, i-2]\!]]$: In this case we only consider the previous score and the CNN training score to relabel the samples. Right after the first iteration we improve the overall performance by around 2%. After 7 iterations, we notice that the performance (average precision and AUC) is not monotonically improving (figure 16: bootstrap 1 and figure 17a). One possible interpretation is that some informative hard examples are sensitive to bootstrapping and they keep changing sides after each iteration e.g. the 4^{th} and 9^{th} image in figure 19

Average, $\forall i \, [\forall j \in [\![0, i]\!], \, w_j^i = \frac{1}{i+1}]$: Now we consider all the previous scores plus the CNN training score to relabel the samples. This tends to stabilize the relabelling as we are less dependent on a single CNN training. After few iterations we can give more confidence to the CNN score by going back to the 2-moving average (starting from iteration 5). Again the performance is fluctuating but it's still improving upon the bootstrapping with the 2-moving average (figure 16: bootstrap 2 and figure 17c).

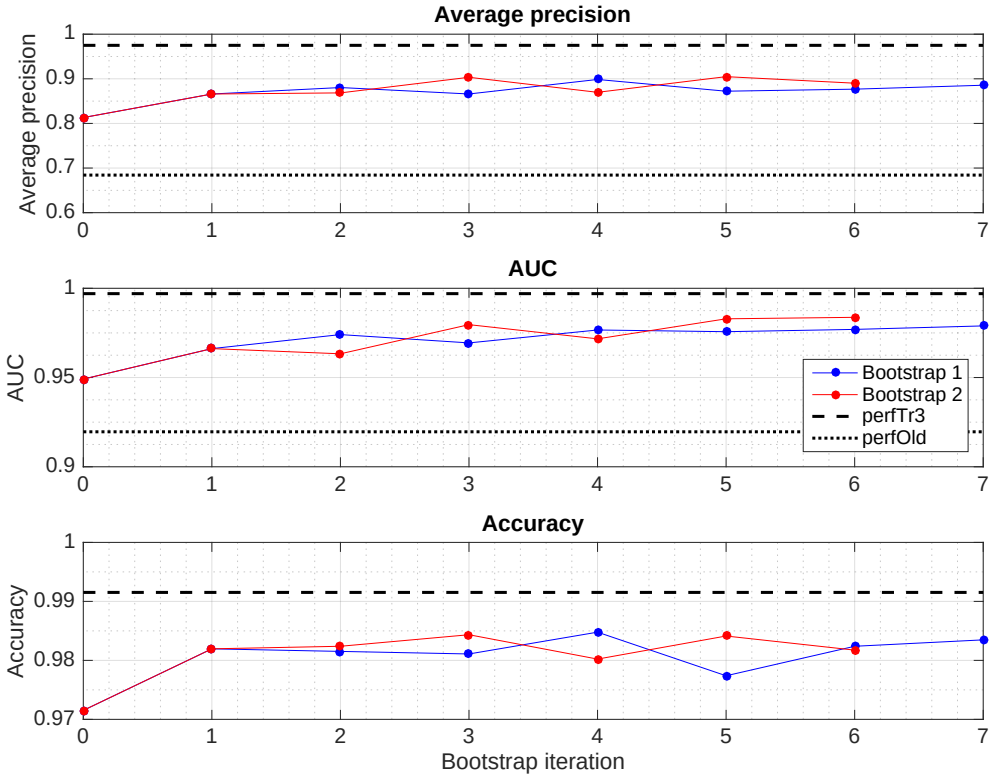
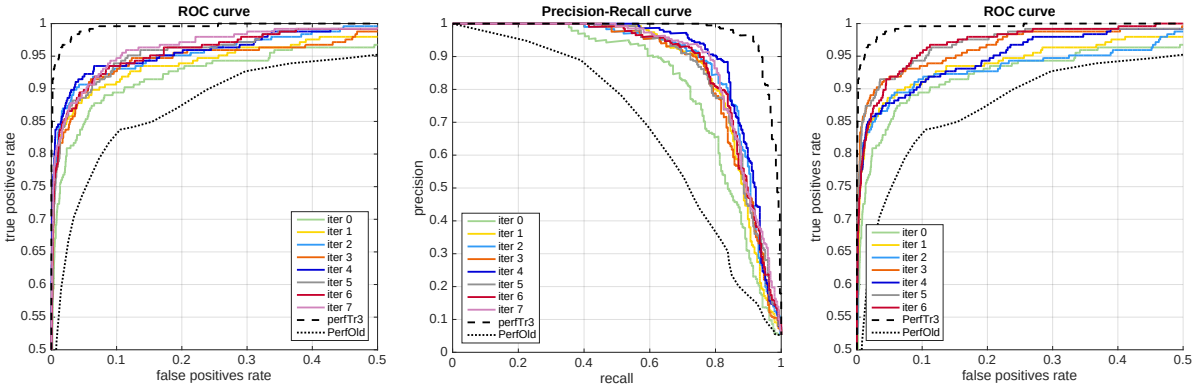
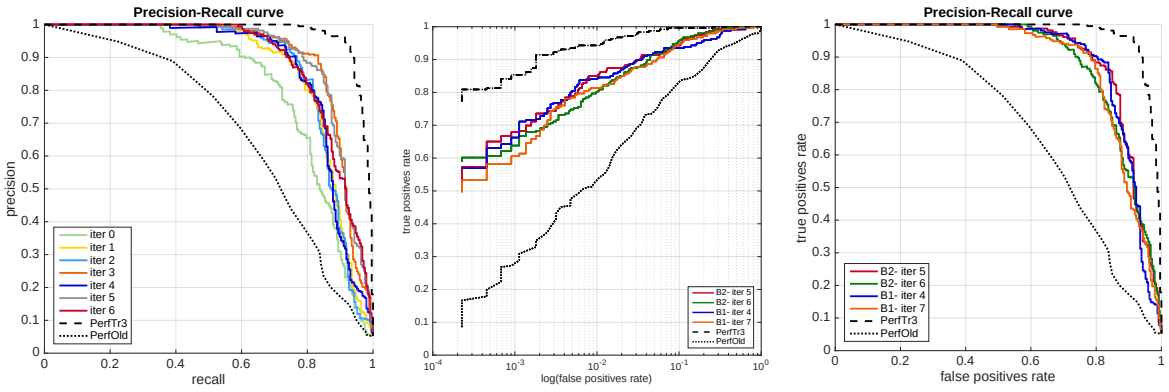


Figure 16: Performance improvement at each iteration of bootstrapping



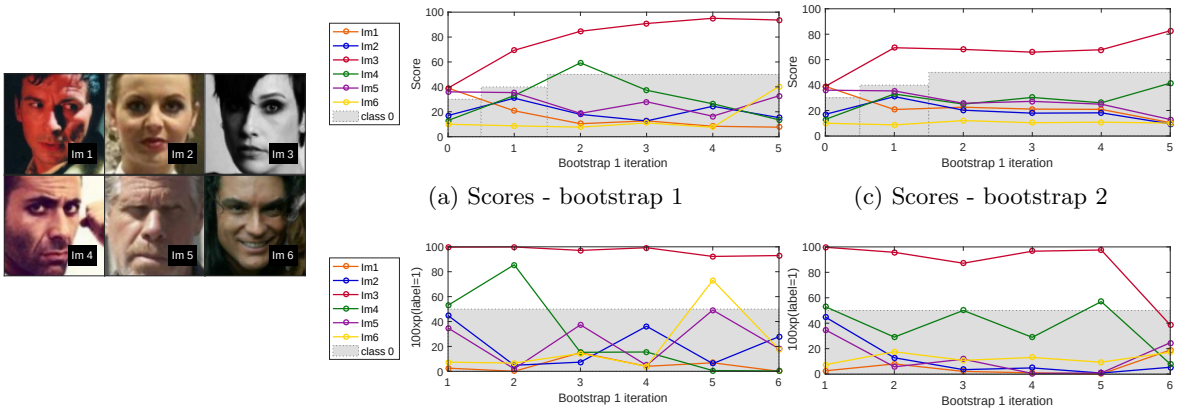
(a) ROC - bootstrap 1 (b) PR - bootstrap 1 (c) ROC - bootstrap 2



(d) PR - bootstrap 2 (e) ROC - comparison (f) PR - comparison

Figure 17: Performance at each bootstrap iteration : bootstrap 1&2

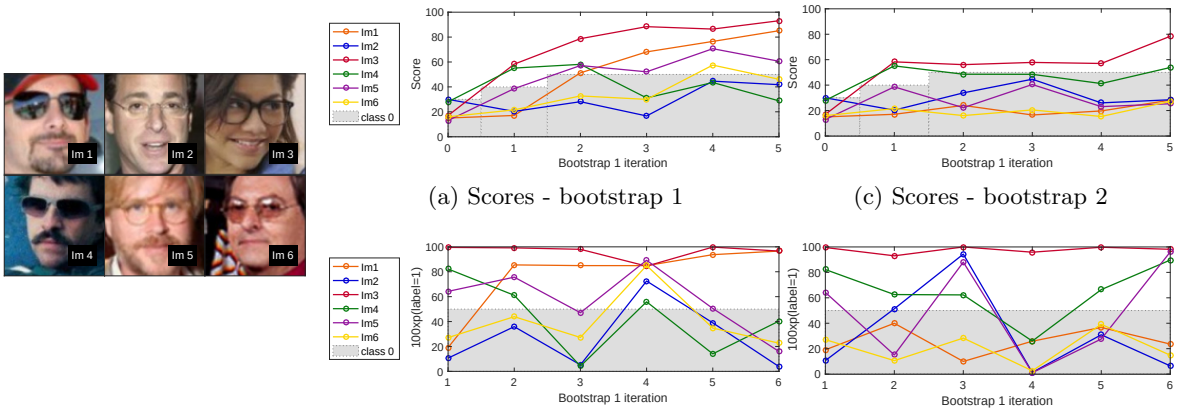
Some samples from the training set are difficult to label and after multiple bootstrap iterations the model fails to reveal their true label, the bootstrap scores are either fluctuating around the decision threshold or completely outside the desired domain.



(a) Scores - bootstrap 1 (c) Scores - bootstrap 2

(b) Probabilities - bootstrap 1 (d) Probabilities - bootstrap 2

Figure 18: Negative samples - scores and probabilities at each bootstrap iteration



(a) Scores - bootstrap 1 (c) Scores - bootstrap 2

(b) Probabilities - bootstrap 1 (d) Probabilities - bootstrap 2

Figure 19: Positive samples - scores and probabilities at each bootstrap iteration

Extension

Instead of relying on the scores of the old classifier for the initial labeling, we can try to use our best classifier, `PerfTr3`, to perform the initial annotation, before bootstraping with the *Average* method described above. As the initial classification is much better than the old classifier, the performances obtained are much better than in the previous section. Due to the initial labelling, the network is more prone to overfitting, and we had to limit the training phase to 5000 iterations to avoid a strong decrease in performances. Yet, we succeeded in obtaining a better accuracy than `PerfTr3` after three iterations.

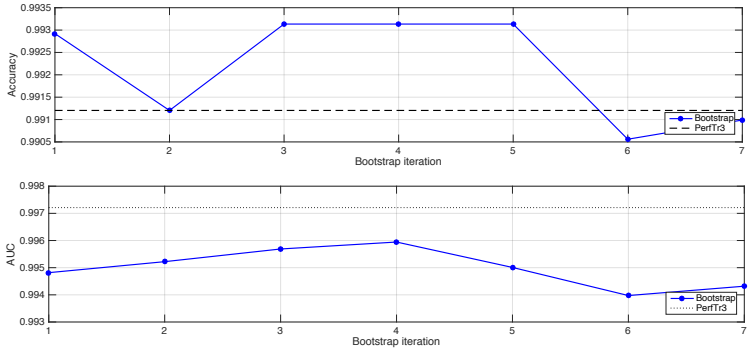
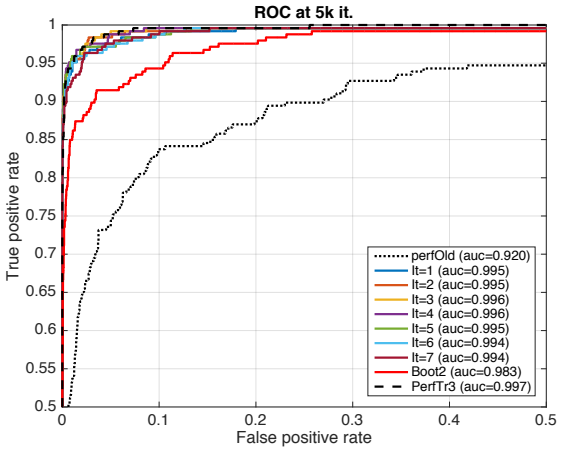
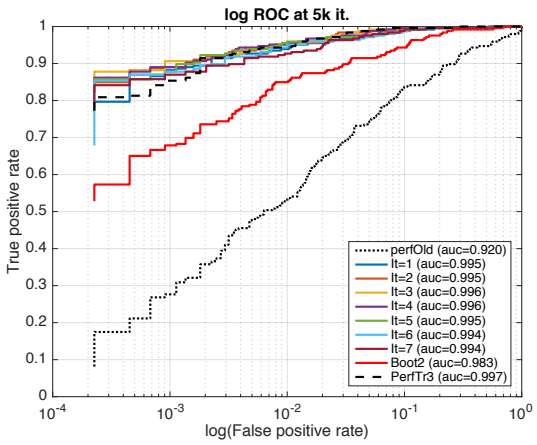


Figure 20

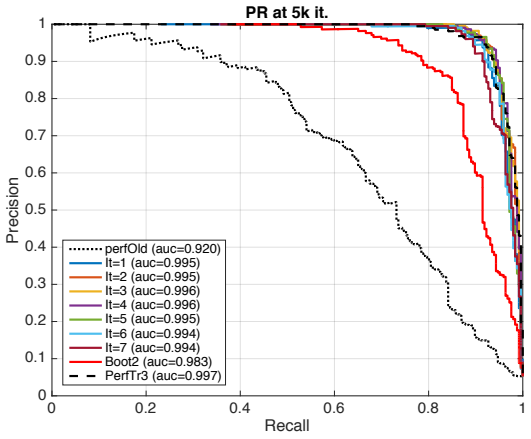
Overall, the Area Under Curve indicator remains close to the initial performance of `PerfTr3`. Even if the gain in performance is small in absolute, compared to the advance obtained when starting the bootstrap with noisy labels (curve *Boot2*) we still get a significant relative increase in performances compared to the `PerfTr3` results. In the table 21d, we see how for small false positive rates (below 0.1) the improvement brought by the bootstrap over the simple `PerfTR3` CNN is significative, peaking at a 55% improvement of the true positive rate after three iterations. The instability of the CNN over the iterations is also apparent: after four iterations the improvement over the baseline decreases, maybe caused by an overfitting.



(a) ROC



(b) log-ROC



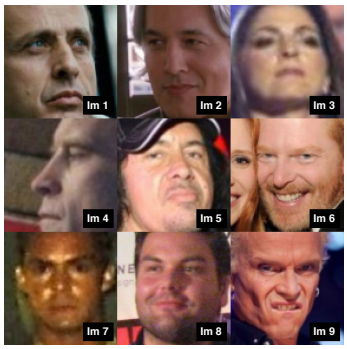
(c) PR

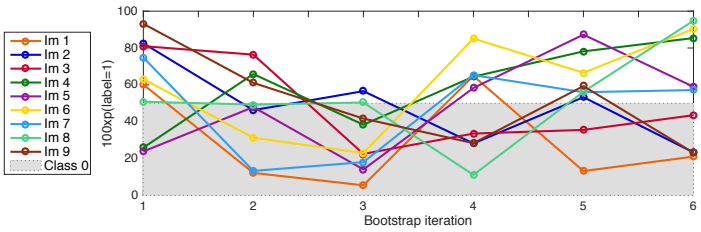
CNN	True pos. rate	% improv.
PerfOld	0.7358	-
PerfTr3	0.9820	0%
It 1	0.9756	-35.5%
It 2	0.9837	9.5%
It 3	0.9919	55.0%
It 4	0.9878	32.3%
It 5	0.9715	-58.3%
It 6	0.9715	-58.3%
It 7	0.9756	-35.5%

(d) Compared performances at 0.05 false pos. rate

Figure 21: Performance at each bootstrap iteration

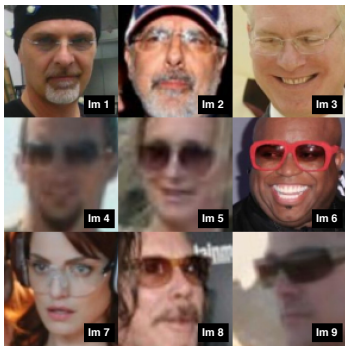
Finally we display selected examples of pictures from the test set where the CNN is uncertain : the classification of those images oscillates from an iteration to the next. The pictures that should be labelled positive are similar to some exhibited before, where the glasses are either extremely thin, or where the central junction of the glasses is not clearly visible. The negative pictures also exhibit examples of unexpected poses (facial orientation), light effects or eyebrows that still disturbs the classifier even if it is not relying on the knowledge of the old classifier.

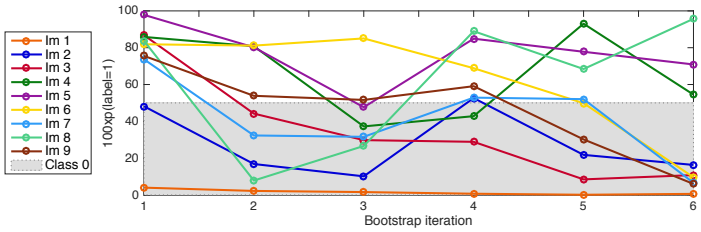




(a) Probabilities $\mathbb{P}(label = 1)$ at every bootstrap iteration

Figure 22: Negative samples (test set)





(a) Probabilities $\mathbb{P}(label = 1)$ at every bootstrap iteration

Figure 23: Positive samples (test set)

By tracking the evolution of the scores of those images, it appears that at the beginning the bootstrap has a contractive action: when considering the negative labels, at the third iteration seven pictures over nine are correctly identified; the same behavior can be observed for positive labels at the fourth iteration, where six out of the nine pictures are correctly sorted. But those evolutions are not synced between both labels, and the scores are not stables in the following iterations. It is important to note that those pictures were selected for their variability from an iteration to the next and the scores evolution is thus generally worst than for a randomly selected image.

4.6 Noise prediction

Instead of considering the problem as a binary classification we distinguish 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 $i \in [\![0, 3]\!]$ 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.

The CNN classifier, trained on a hand labelled 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 $t = 40$ where the training set has the following distributions:

	truth	
	0	1
score $<$ th	c0	c1
score $>$ th	c2	c3

(a) Classes

class	# training	# test
noisy=true=0	3888	4349
noisy=0 true=1	88	100
noisy=1 true=0	50	66
noisy=true=1	126	146

(b) t=30

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

(c) t=40

Table 6: Training & test set statistics when thresholding at t=30 and at t=40

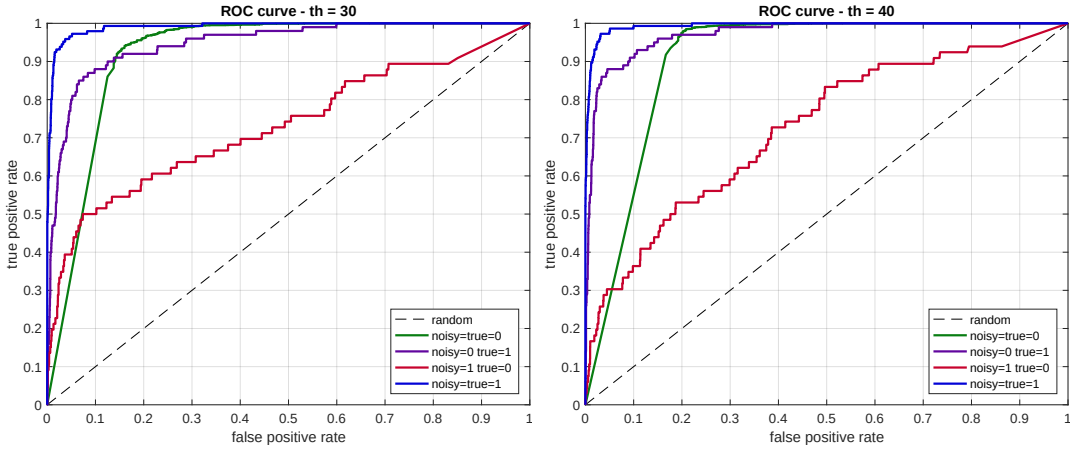
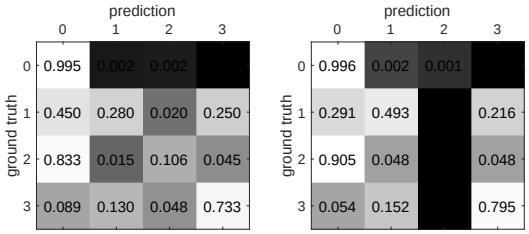


Figure 24: Multiclass ROC curves

The model is unable to expose the false positives, which is quite expected seeing the dataset skewness, and most samples of class 2 end up in class 0. This fact is more pronounced with $t = 40$. In the context of labels cleansing this is harmless as both classes will lead to the same correction.

Class	Truth	Old label
0	0	0
1	1	0
2	0	1
3	1	1

(a) Classes



(b) t=30 (c) t=40

Figure 25: Confusion matrices

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 (figure 26). On the bright side, with all the labels flipped we can reach an accuracy of 98.8% with the model learned from cleansed labels originally thresholded at $t = 40$.

(a) Flipped labels t = 30

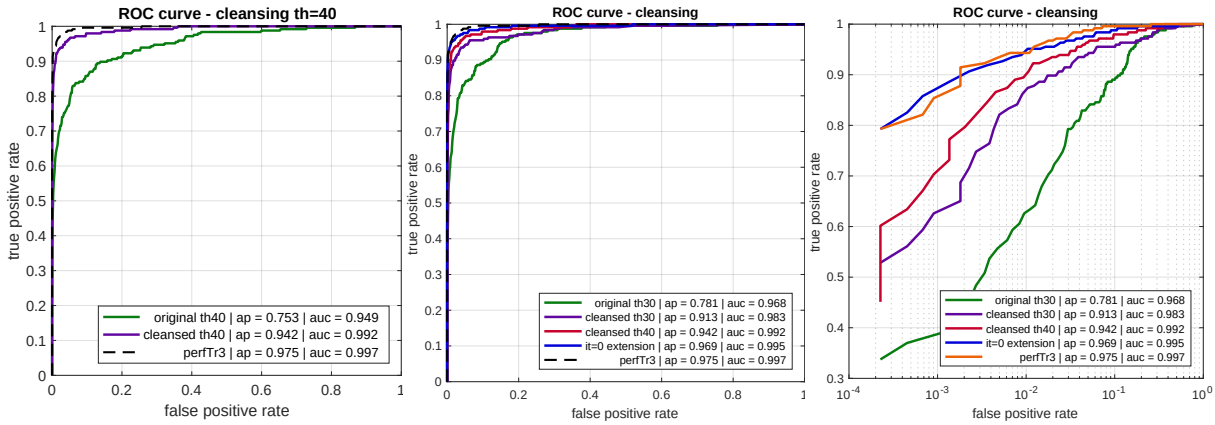
(b) Misclassified t = 30

(c) Flipped labels t = 40

(d) Misclassified t = 40

Figure 26: Extract from the test set

After rectifying the labels we run the baseline (softmax loss) on the full training set `CASIA` $\approx 382k$ images. Although thresholding at $t = 40$ gave poor performances compared to other thresholds (c.f. 4.3.1) it's more prone to noise prediction with more false negatives that the model has proven to deal with successfully and less false positives harder to unmask (figure 27).



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

Figure 27: Performances improvement: for $t = 30$ and $t = 40$, baseline model on noisy labels compared to the model with cleansed labels. $it = 0$ extension is the first bootstrap iteration when labelling the full training set with `perfTr3`. This first iteration could be seen as cleansing the labels with a binary classifier trained on clean labels (`Tr3`)

5 Testing

In order to assess the performances of our models independently from the `CASIA` dataset specificities, we tested it on other datasets with similar samples. We compare the performances of the old classifier with three of our models:

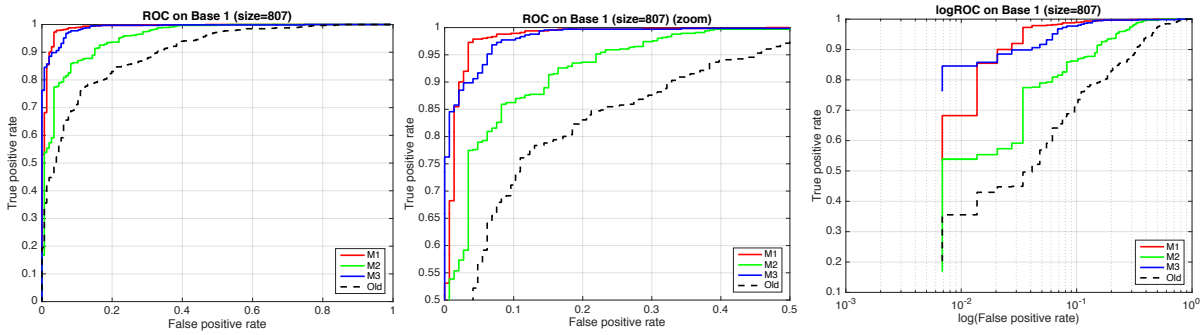
- the model trained on the hand-labelled set `Tr3` with the infogain loss, denoted `M1`.
- the model learned by bootstrapping (5 iterations) on the noisy dataset, using the *average* update rule, denoted `M2`.
- the model learned by bootstrapping (3 iterations) on `CASIA` initially relabelled with `PerfTr3` with the same *average* update rule, denoted `M3`.

The pictures in each of the datasets used are in a similar form-factor as the ones in `CASIA`, and have the following characteristics:

Base	#samples	#positive samples	positive ratio	Specificities
B1	807	146	18.7%	All poses, no expressions, mainly caucasian.
B2	2000	90	4.5%	Fixed pose, few expressions.
B3	5000	332	6.6%	Fixed pose, few expressions, mainly caucasian.

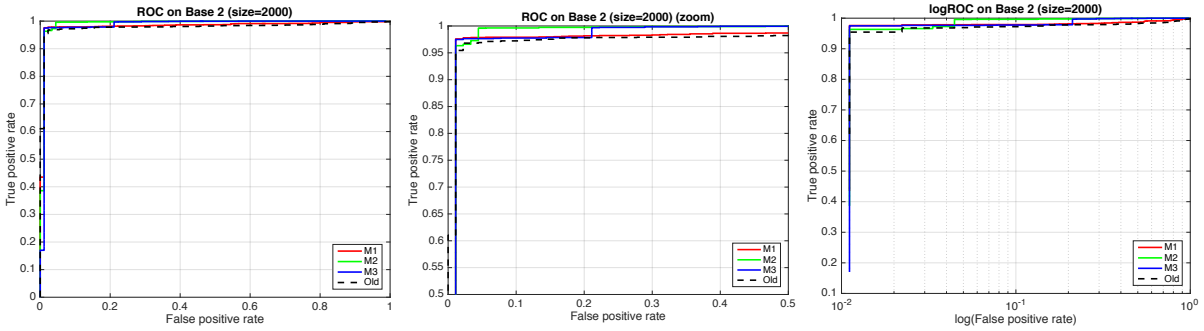
Table 7: Characteristics of the test bases.

On the first set, as shown in figure 28, we obtain results similar to those seen while validating our models: the models learnt on clean subsets perform better, while the bootstrap learned on the noisy set performs halfway between those and the old classifier. Results on the two other sets (resp. figures 29 and 30) are much harder to interpet due to the repartition of data. Nevertheless both M1 and M3 always perform better than the old classifier baseline. The M2 model is probably more sensitive to the specificities of each dataset as it relies on the scores of the old classifier. Overall, those tests allowed us to validate our models on other similar datasets, showing that the CNN did not overfit during the learning phase and succeeded in capturing significant features of the images.



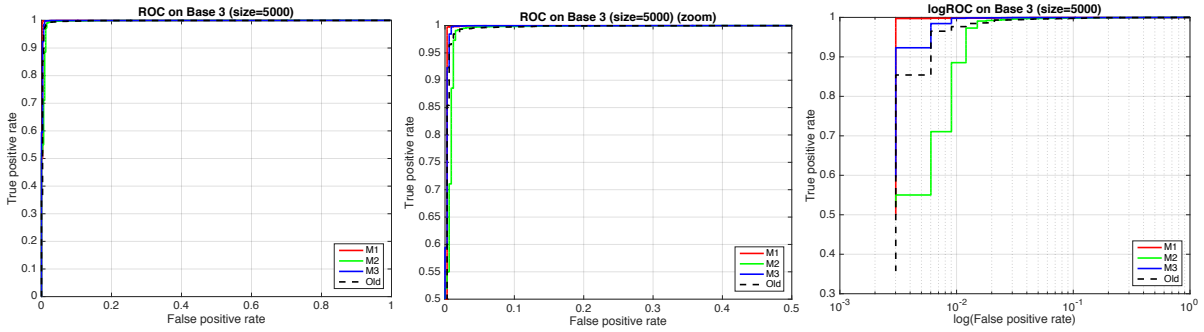
(a) ROC (b) ROC (zoom) (c) log-ROC

Figure 28: Results on B1



(a) ROC (b) ROC (zoom) (c) log-ROC

Figure 29: Results on B2



(a) ROC (b) ROC (zoom) (c) log-ROC

Figure 30: Results on B3

6 Conclusion

Through this project, we studied multiple methods, from pre-processing the data to modifying the CNN learning phase. Overall, the best solution performance-wise remains to use a hand-labelled small subset: our best results were obtained by labelling less than 2% of the `CASIA` dataset. Classical performance enhancement techniques such as bootstrapping have shown to be very efficient on both clean and noisy datasets, improving accuracy and precision.

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-labelling option, the error-prediction experiment appears as extremely interesting in the sense that it can be plugged into an existing data processing pipeline.

7 Acknowledgments

We would like to thank our supervisor, C. Herold, along with S. Gentric and V. Despiegel at Morpho, for guiding us during the whole project, taking the time to answer our questions, helping us with our results and proof-reading this report. We also thank J. Milgram and J. Bohné from Morpho for their insightful remarks. Finally, we would like to thank the pedagogic team at Centrale, especially N. Paragios for his supervision and L. Series for helping us with all technical matters related to Caffe.

8 Appendices

8.1 Hyperparameters of the CNN

Configuration file used to set up the CNN in Caffe.

```
#Hyper-parameters

base_lr: 0.01
lr_policy: "inv"
power: 0.75
gamma: 0.0001
stepsize: 4000

momentum: 0.95
weight_decay: 0.00

test_iter: 100
test_interval: 100
test_initialization: false
```

8.2 Other experiments

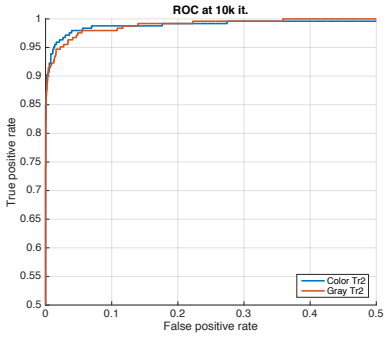
Other peripheral experiments were made during our study. We shortly describe them below, along with the obtained results.

8.2.1 General run time

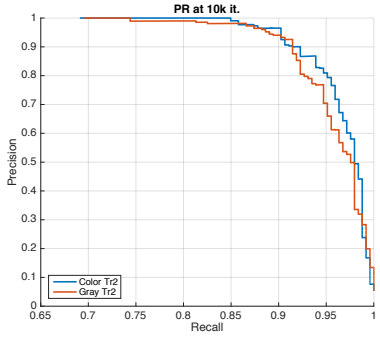
Our first test was devoted to finding which number of iterations should be necessary in order for our network to stabilize. We thus started by letting the training phase run for 150k iterations. As the results exhibited overfitting and by tracking the evolution of accuracy and AUC, we settled for an ideal number of iterations of 10k. This corresponds to a run time of 20 to 25 minutes on a single Nvidia K40 GPU card. We also generally extracted the results at 5k and 20k iterations for most of our runs, in order to detect variations of performance in different experiments.

8.2.2 Color vs gray

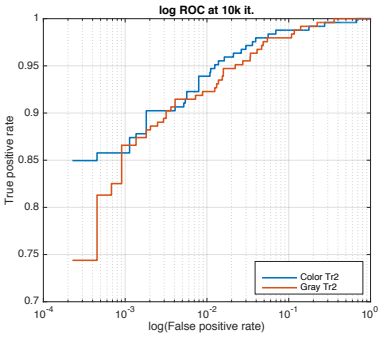
Another early experiment was performed in order to assess the importance of using RGB pictures instead of gray levels pictures. A test led on the `Tr2` base and validated on the `Test` set gives AUCs of 0.991 and 0.982 for RGB and Gray levels respectively. The difference brought by keeping the three channels of the images is thus non-negligible ; this confirms the fact that the gain in information brought by having access to the color (and not only the luminance) is useful for detecting (an absence of) glasses in some pictures.



(a) ROC curves (zoom)



(b) Precision-Recall curves



(c) ROC log curves

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[3] David Flatow and Daniel Penner. On the robustness of convnets to training on noisy labels.

[4] Benoît Frénay, Ata Kabán, et al. A comprehensive introduction to label noise. 2014.

[5] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(5):845–869, 2014.

[6] Jiaqi Ge, Yuni Xia, and Chandima Nadungodage. Unn: a neural network for uncertain data classification. In *Advances in Knowledge Discovery and Data Mining*, pages 449–460. Springer, 2010.

[7] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1196–1204. Curran Associates, Inc., 2013.

[8] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training Convolutional Networks with Noisy Labels. *arXiv.org*, June 2014.

[9] Jan Larsen, L Nonboe, Mads Hintz-Madsen, and Lars Kai Hansen. Design of robust neural network classifiers. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1205–1208. IEEE, 1998.

[10] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training Deep Neural Networks on Noisy Labels with Bootstrapping. *arXiv.org*, December 2014.

[11] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[12] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

[13] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep convolutional network cascade for facial point detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3476–3483, 2013.

[14] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision–ECCV 2014*, pages 94–108. Springer, 2014.

[15] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11. Citeseer, 2003.

[16] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978, 2001.

[17] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):63–77, 2006.

[18] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels, 2015. proc, ICLR.