Research Initiation Seminar

Glasses detection with CNN/Deep Learning

Safran Morpho

Maha Elbayad, Simon Rodriguez

Supervisor
C. Herold

SAFRAN Morpho

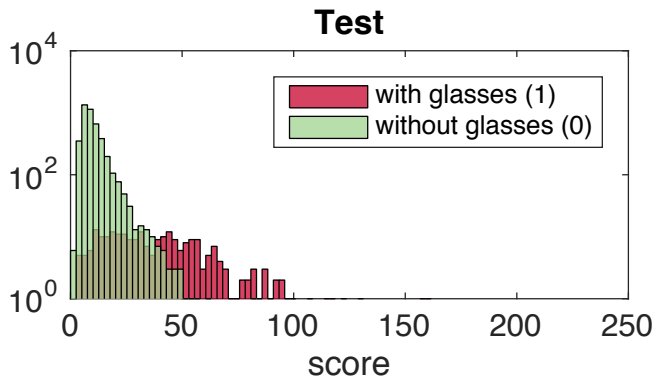
CentraleSupélec

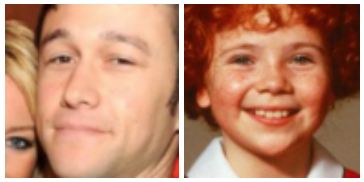
ECP Supervisor
N. Paragios

Subject

Train a CNN to say whether a facial image contains eyeglasses or not.

Training dataset : labelled using an older, gradient-based detector → noisy labels.





5.05 18.37

64.21 17.00

Old classifier scores

- **How to deal with noisy labels ?**
- **How to get the best performances ?**
- **How to binarize the old classifier scores ?**

Setting

We use the Caffe framework, with a fixed predefined network architecture:

- 3 x {conv-ReLU-pool} + 2 x fully connected layers
- Dropout (50%)
- Softmax with loss (multinomial logistic loss)

Database : *CASIA WebFace*

- 380k faces (collected from IMDB)
- 96 x 96 x RGB
- Cropped, scaled and centered on the eyes.
- Variable poses, lighting, expressions, ethnies and image quality.

Tackling the problem

Two approaches:

- **Preprocess the dataset**. Always needs to be adressed, as the scores given by the old classifier are in [0,255], and we want to convert it to a binary classification problem.

- Handle the noisy labels **during the training phase**.
  - Change the structure of the CNN, by adding new layers to correct the errors or learn the noise or using custom losses.
  - Use a small hand-annotated subset for regularization.
  - Classical perf. improvements methods of machine learning.

State of the art

- **Preprocessing**

  Thresholding the scores, studying the repartition of labels. * (Frenay & Verleysen, 2014)
  Reject uncertain samples. *
  Learn the error of the old classifier to correct the labels. *

- **During training**

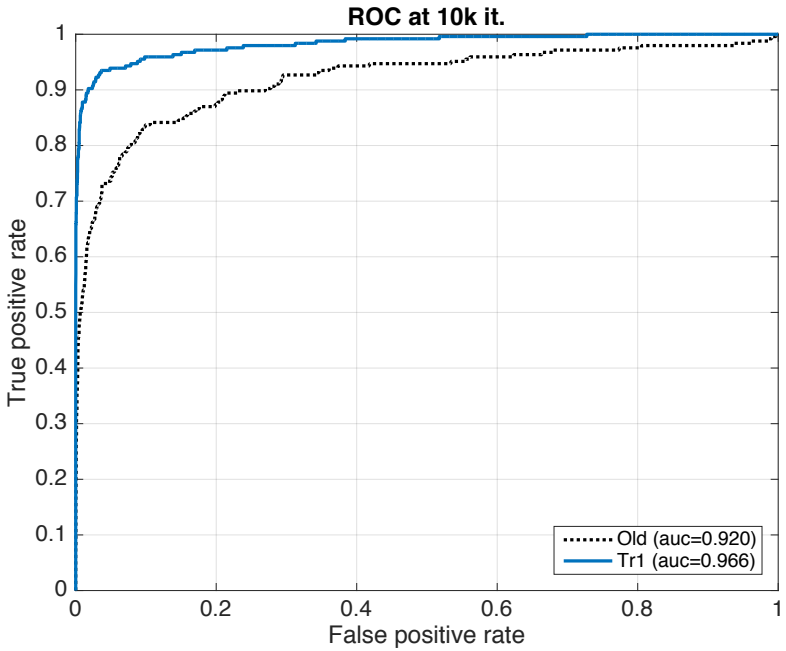
  Different losses on different subsets. (Zhou & Liu, 2006)
  Design or learn a confusion matrix. * (Sukhbaatar et al., 2014)
  Classes for different types of noise. (Natarajan et al., 2013; Xiao et al., 2015)
  Learn the error of the old classifier to correct the labels. *
  Bootstrapping, bagging. *

Various prerequisites on the knowledge of the noise distribution, on a small hand-labeled subset of the training set, etc.

Establishing a baseline

Datasets:

Name	#samples	Labels	% positives
Test	4.6K	hand-labelled	5
CASIA	375K	noisy	5
Tr1	5K	hand-labelled	5
Tr2	5K	hand-labelled	20 (other source)
Tr3	10K	hand-labelled	20 (Tr1+Tr2, adjusted)



Baseline:

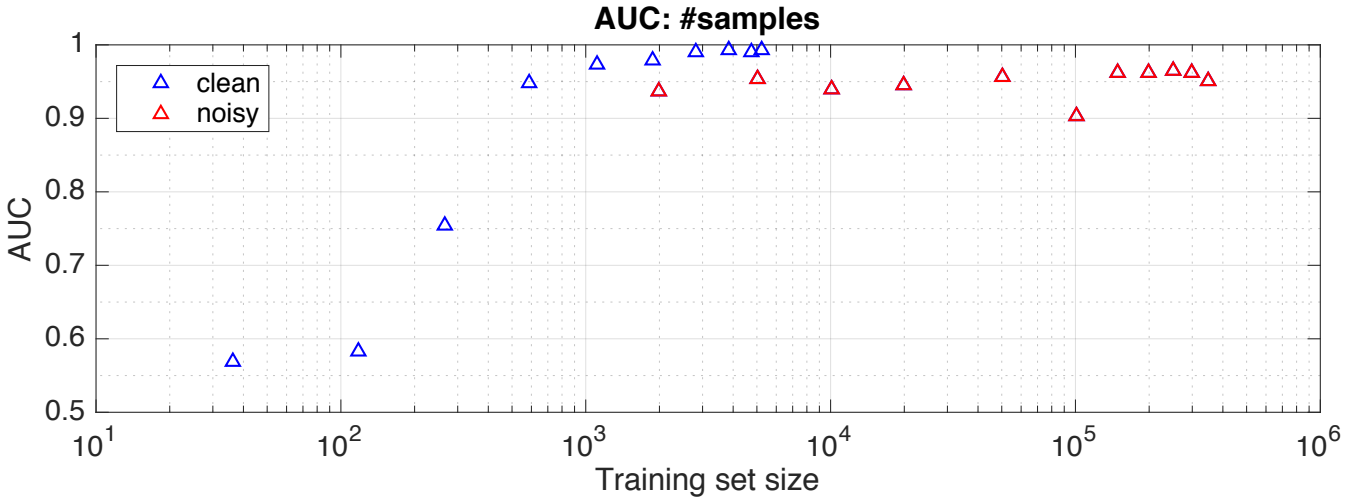
Compare the old gradient-based detector with a CNN trained on Tr1 (no noise).

Tr1:
Accuracy: 0.985
AUC: 0.966
Avg. Precision: 0.913

Old:
Accuracy: 0.962
AUC: 0.920
Avg. Precision: 0.679

Influence of the set size

Importance of the set size : how do training on a small hand-labelled subset and training on a big noisy set compare?



	Dataset	Labels	% of positives
Clean	Tr1	Hand-labelled	5%
Noisy	CASIA	Automatically generated using the old scores	Sampled and thresholded at 26.7 to get 5% of positives.

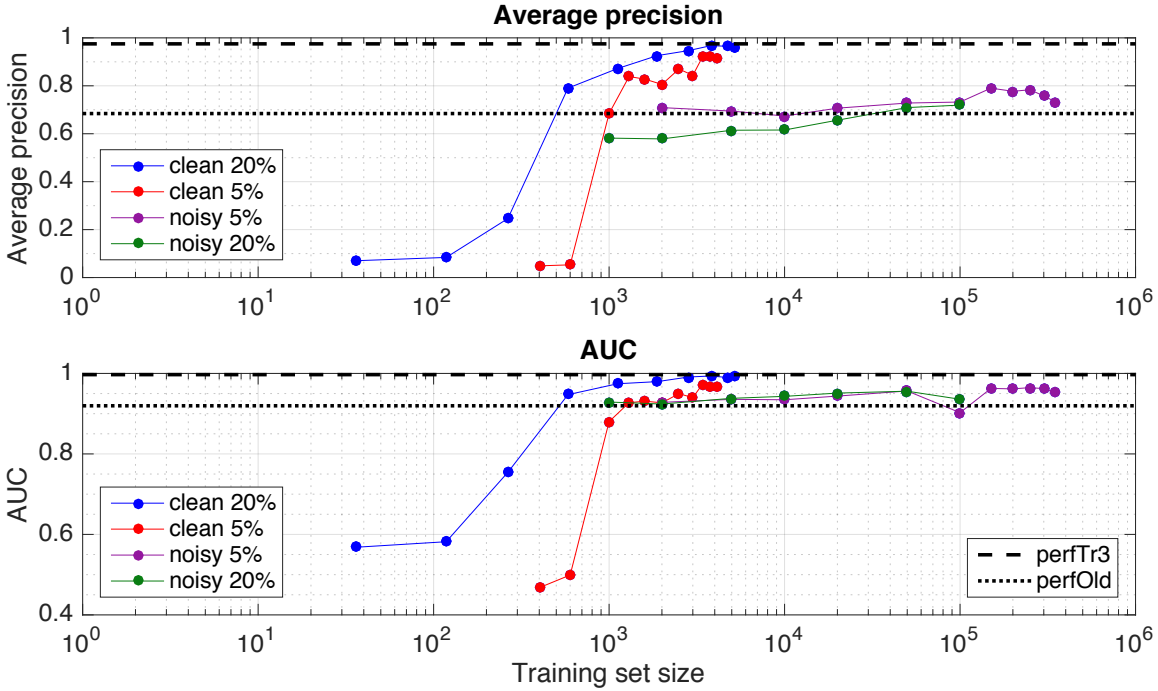
Annex

Impact of the labels ratio

What about the repartition of the labels ? Clearly unbalanced.

→ Clean : nested databases starting from TR3 (resp. TR1).

→ Noisy: nested databases starting from CASIA(5%) (resp. a balanced CASIA(20%)).



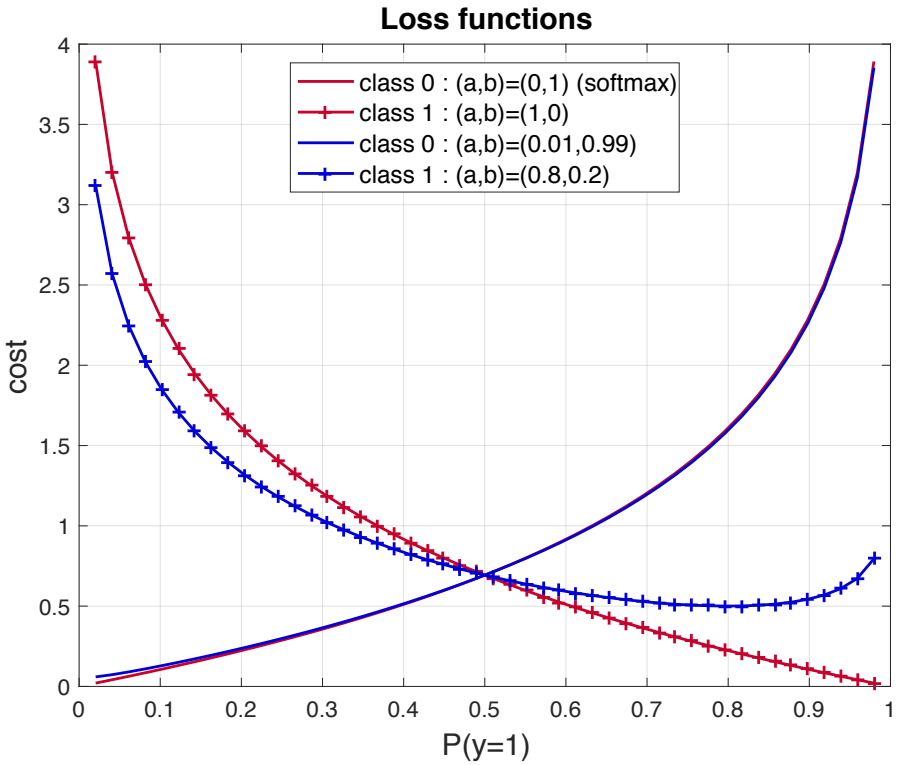
Improvement when trained on a clean set, not when trained on a noisy one.

Annex

Handling the class imbalance

Consider a loss per class to weight out the asymmetry of the classes:

$$\tilde{\mathcal{L}}_{(a,b)}(\hat{p}) = -[a \log \hat{p} + b \log(1 - \hat{p})]$$



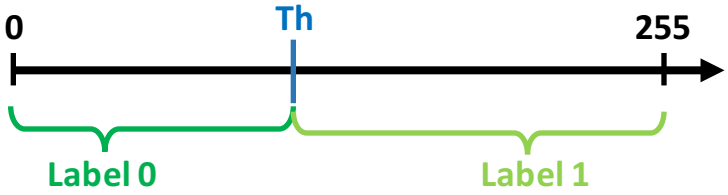
Intuition:
Differentiate the costs associated with false positives and false negatives.

Refinements with adjusted loss (blue) gives better results
Tr3 softmax:
(Ap:0.957, AUC:0.993)
Tr3 unbalanced :
(Ap: 0.975, AUC:0.997)

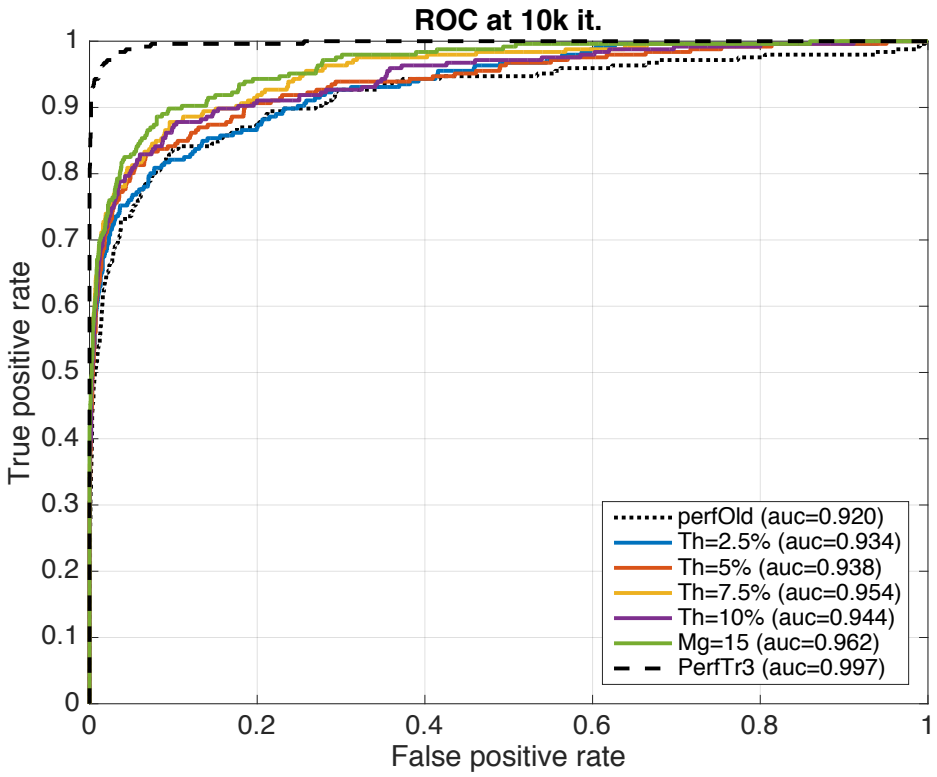
Annex

Preprocessing

Thresholding:
intuitive way to binarize while respecting the labels repartition.



By setting a threshold slightly above the observed repartition, we re-balance the training set in a non-uniform way.

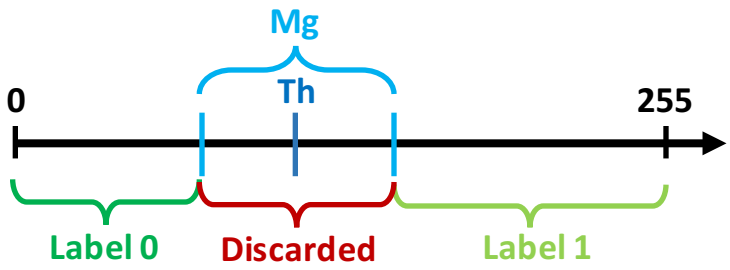


Margin: tested a rejection margin around the threshold (fixed at 5%).
Best performance for a margin of 15 points:
286K images, 9.5K positive labels, Acc: 0.968, AUC: 0.962

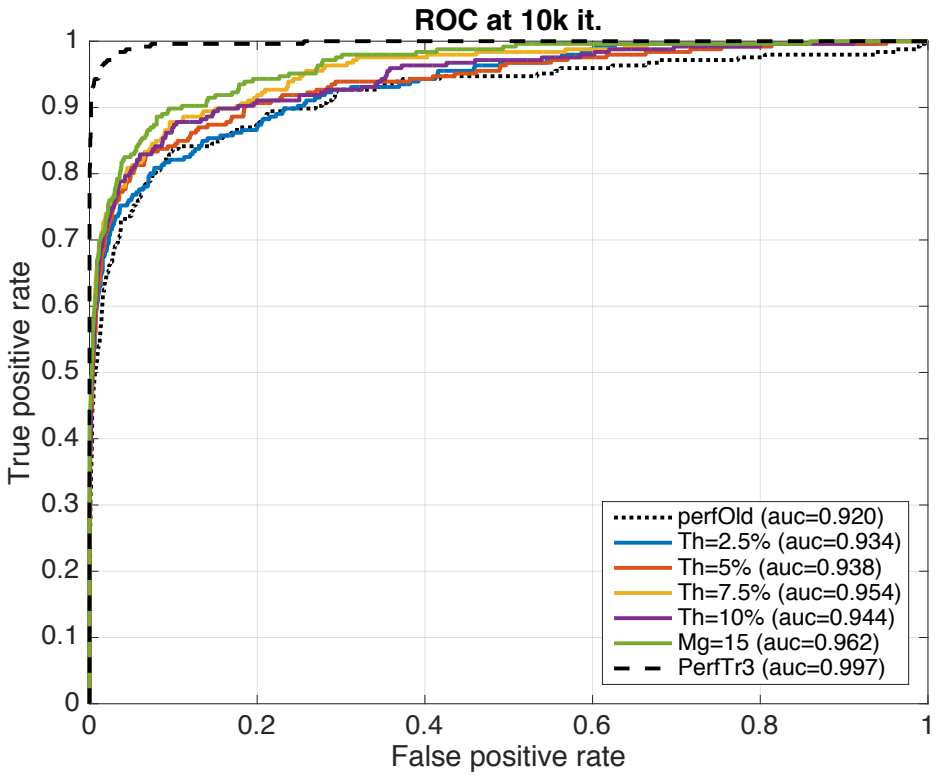
Annex

Preprocessing

Thresholding:
intuitive way to binarize while respecting the labels repartition.



By setting a threshold slightly above the observed repartition, we re-balance the training set in a non-uniform way.

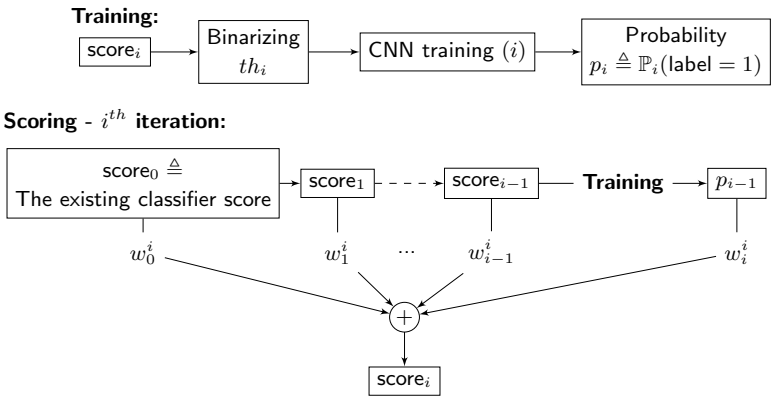


Margin: tested a rejection margin around the threshold (fixed at 5%).
Best performance for a margin of 15 points:
286K images, 9.5K positive labels, Acc: 0.968, AUC: 0.962

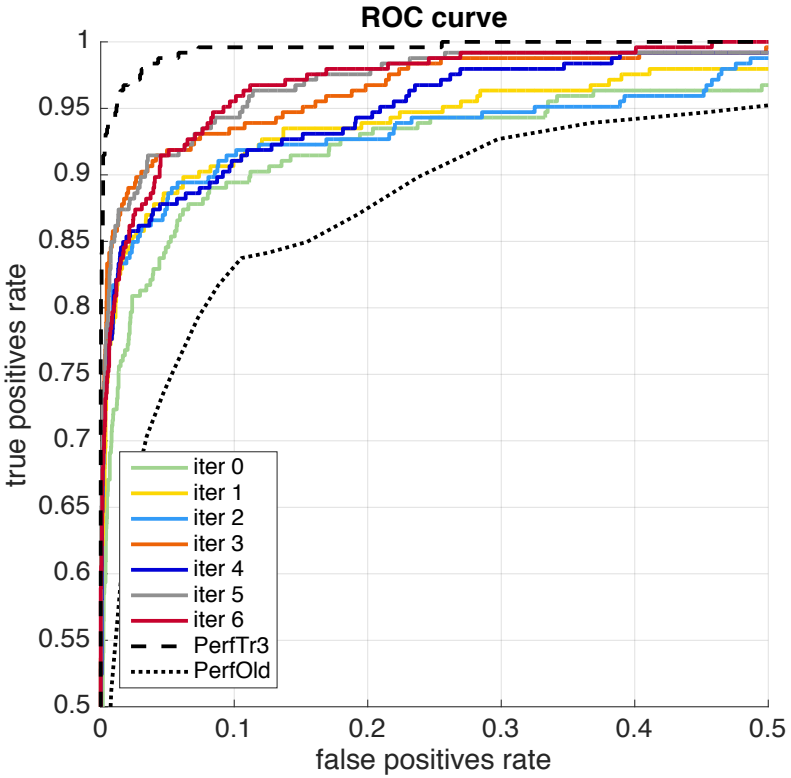
Annex

Bootstrapping

Inputs: images , old classifier scores
Outputs: bootstrapped scores, new model
Initialization Train a CNN on binarized labels
repeat
  compute new score
  update labels
  train a new CNN
until *convergence*



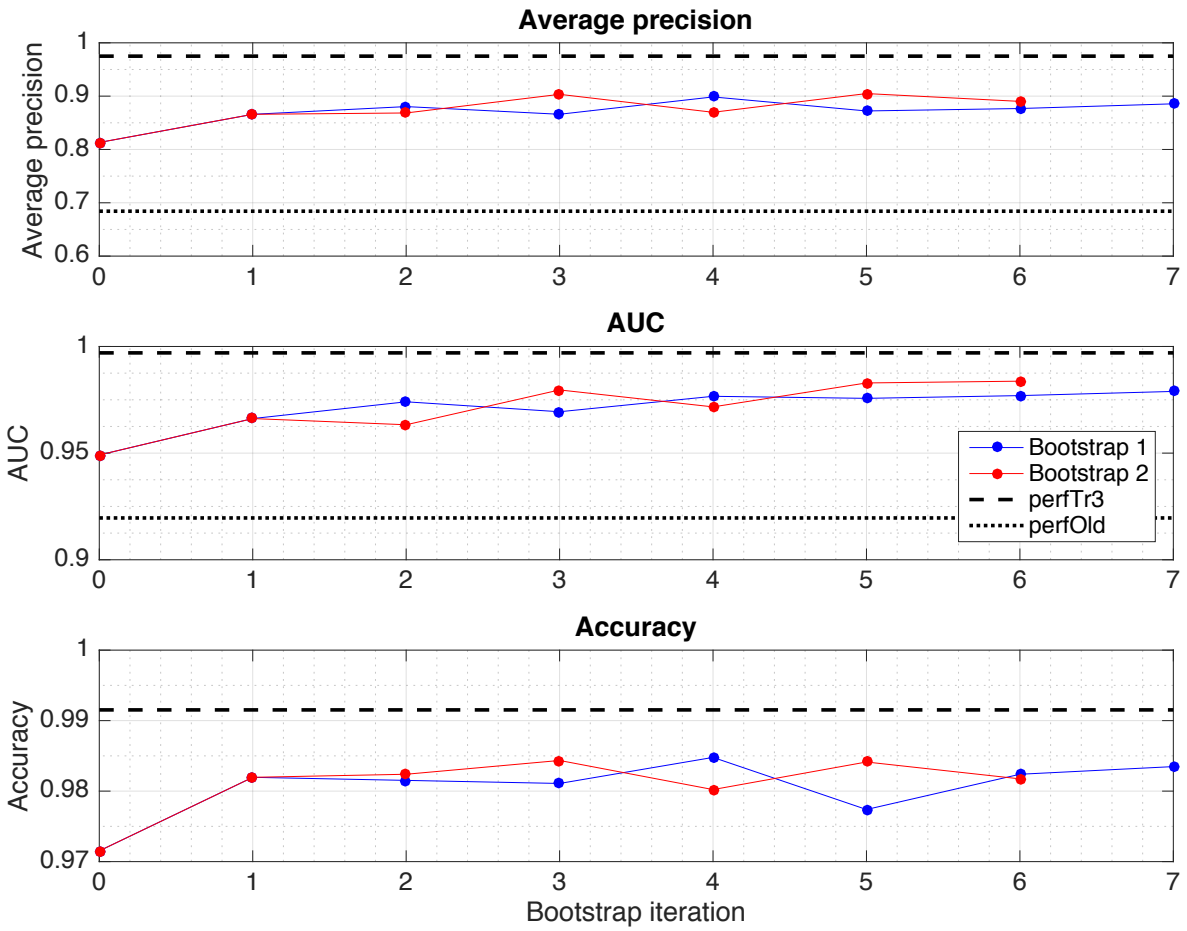
To ensure some « continuity », we set the new score as a weighted average of the previous scores.



Bootstrapping

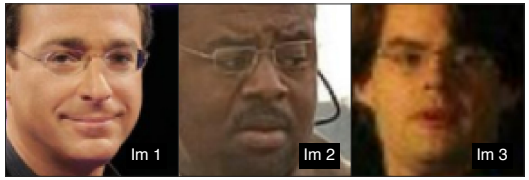
Bootstrap 1:
Average of the last two scores.

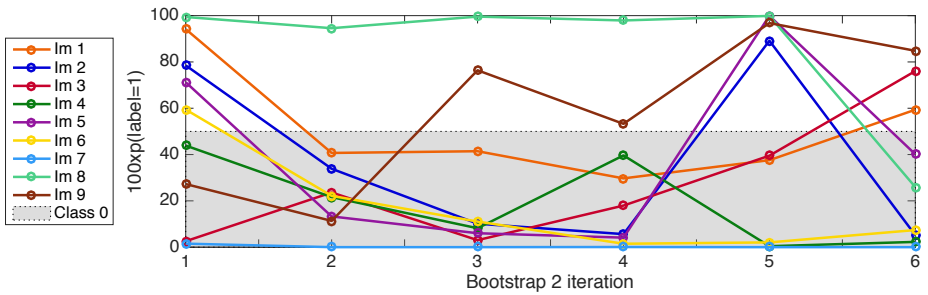
Bootstrap 2:
Average on all previous scores.

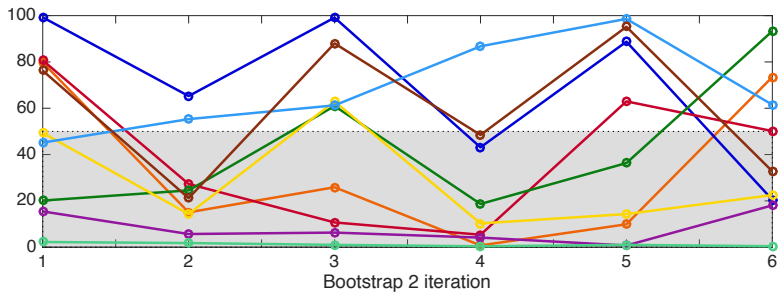


Bootstrapping

We observe strong fluctuations in the probabilities, yet the performances are better.

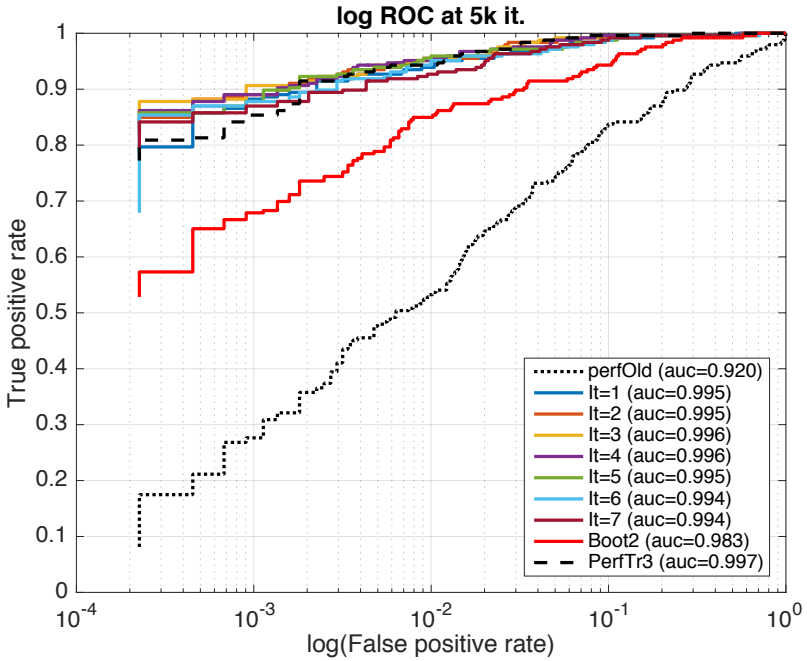


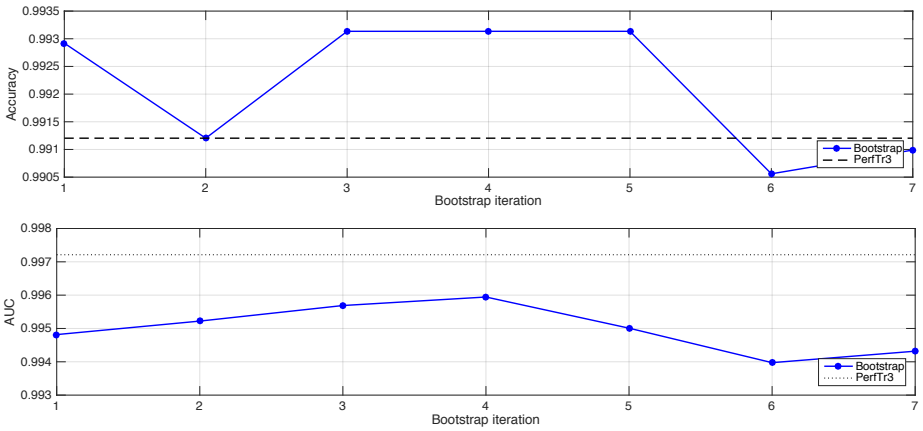




Bootstrapping extension

Instead of starting from the old classifier labels, use a CNN trained on Tr3 (clean) to initially re-label the training set, then perform bootstrap with averaged scores.



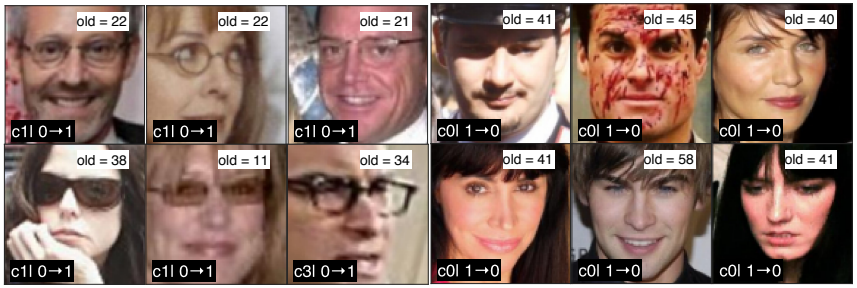


The starting point is much better than in the previous experiment, and the relative improvement is similar for low false positive rate.

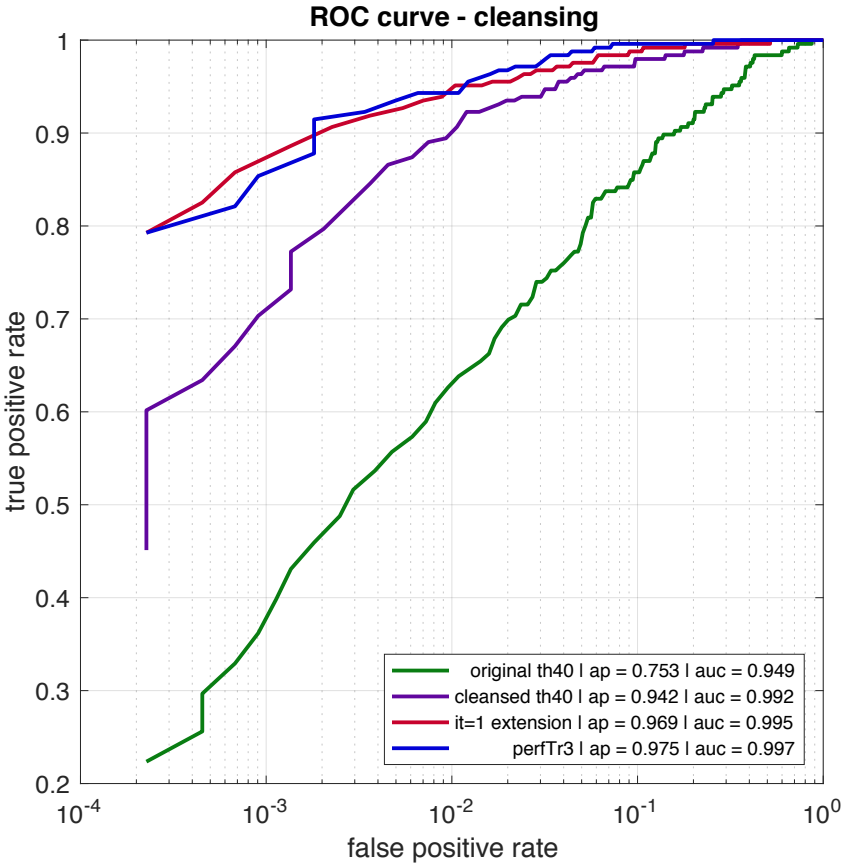
Predicting the errors

We train a 4-classes classifier to decide whether or not the binarized label from the old classifier is correct.

(Classes)	True 0	True 1
Predicted 0	0	1
Predicted 1	2	3



We then relabel the samples according to the predicted class and train a new CNN.



Annex

Conclusion

The best solution performance-wise remains to use a hand-labelled small subset. Classical performance enhancement methods work : bootstrapping has shown to be very efficient on both clean and noisy datasets.

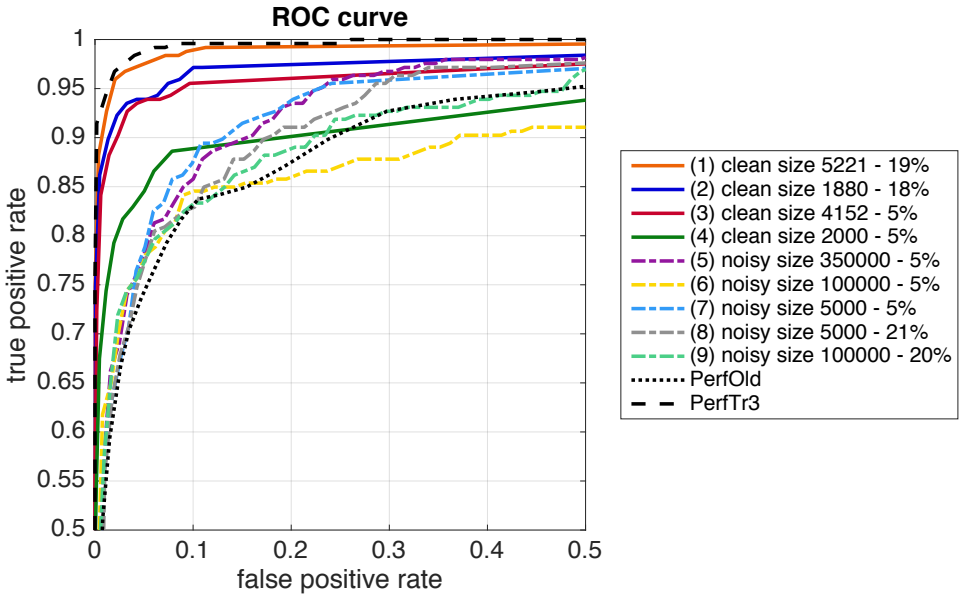
The models that are taking into account (either in the classifier or via another CNN) the dependence between the noise and the content of the pictures in more refined ways are still limited to simple cases, but are the most promising. Litterature on this topic still focuses on simplified cases, as the whole problem is much more complex.

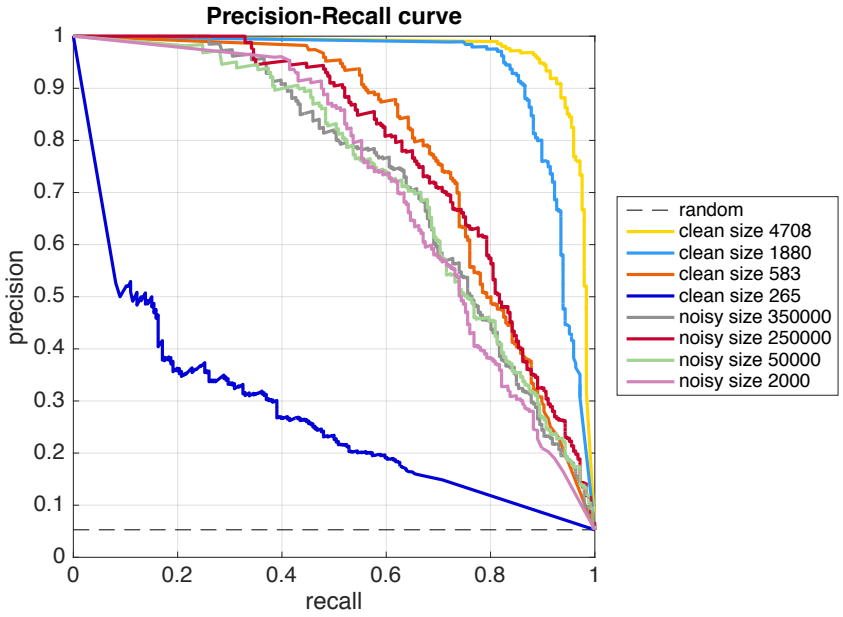
The performances of our models have been validated on other bases with a similar format.

Thank you!

Questions

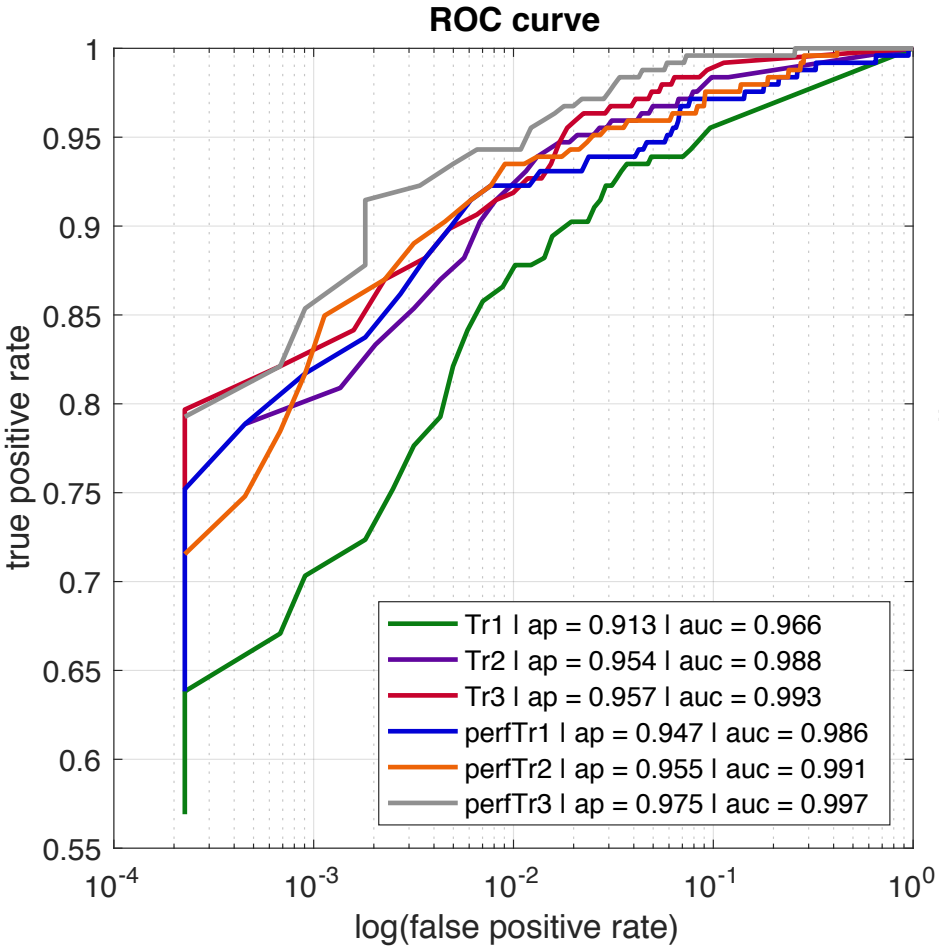
Appendices - Sizes

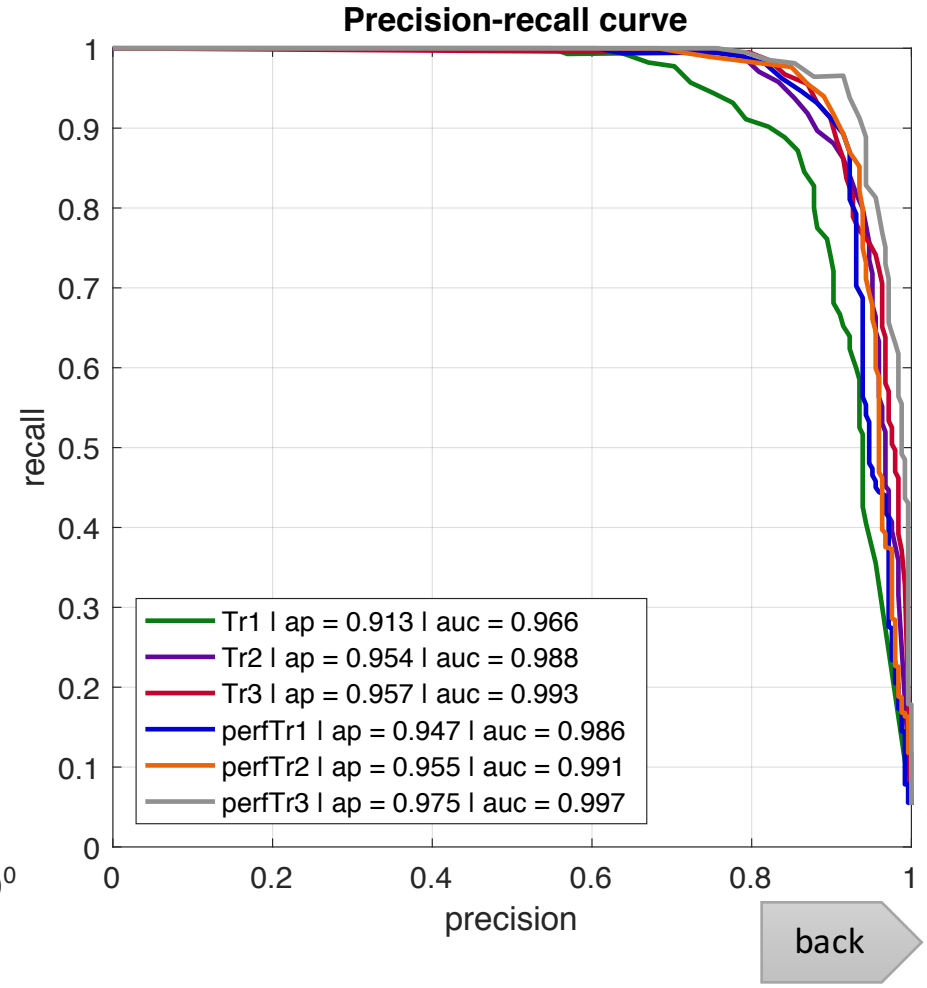




back

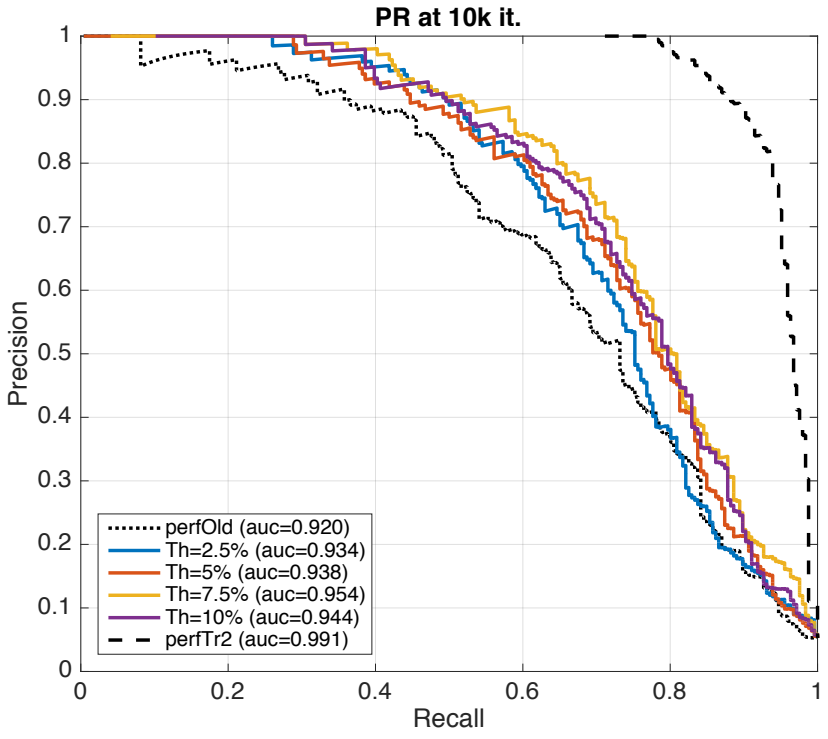
Appendices – Unbalanced

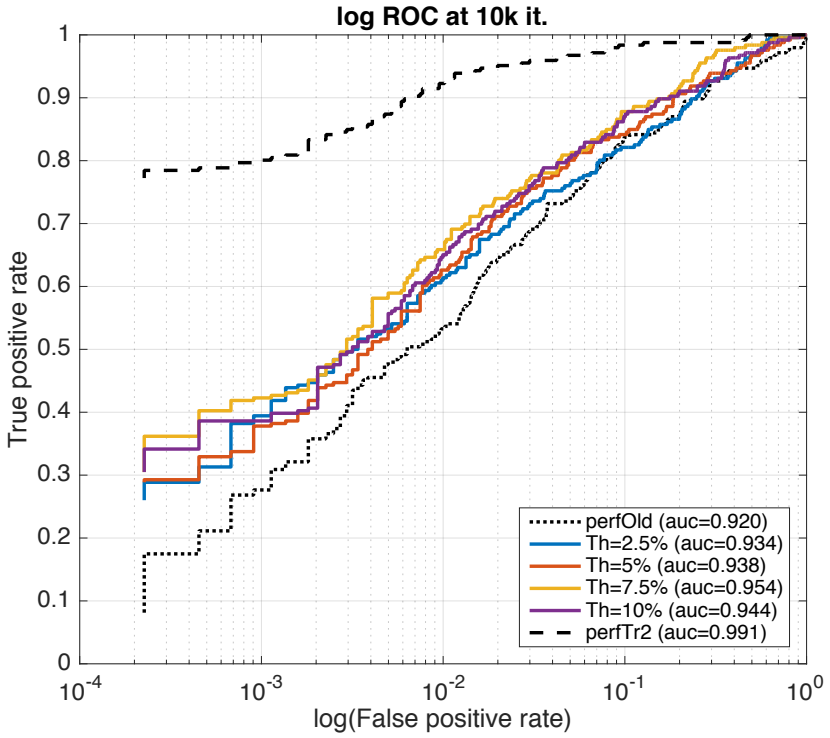




back

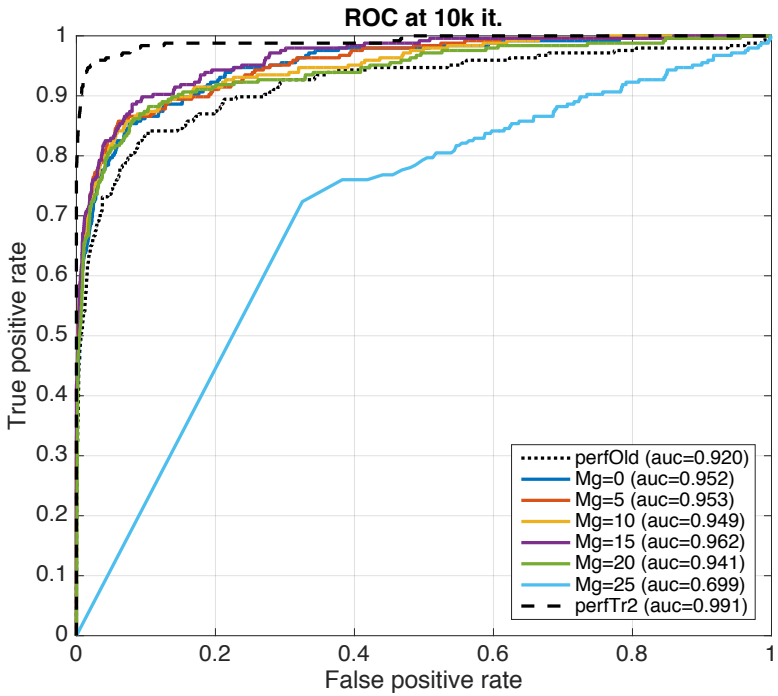
Appendices - Threshold

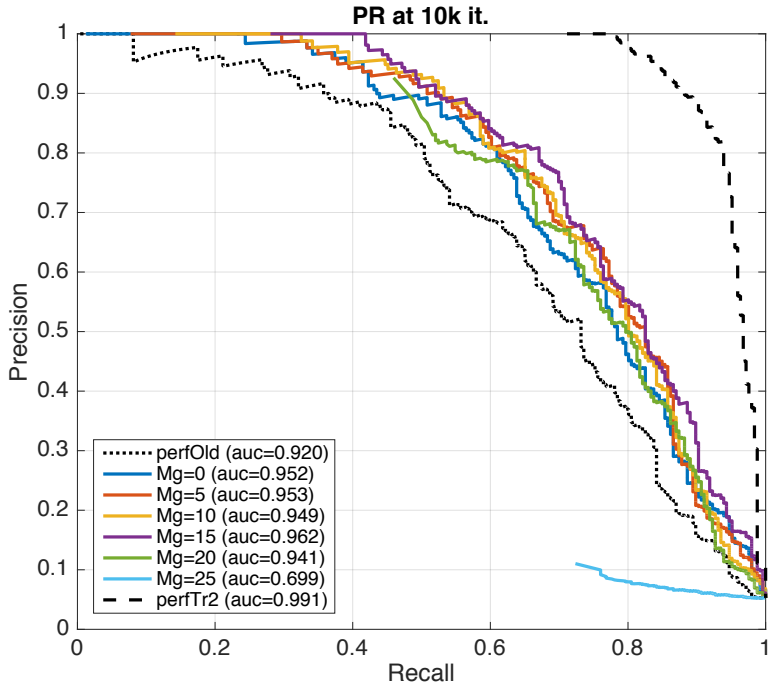




back

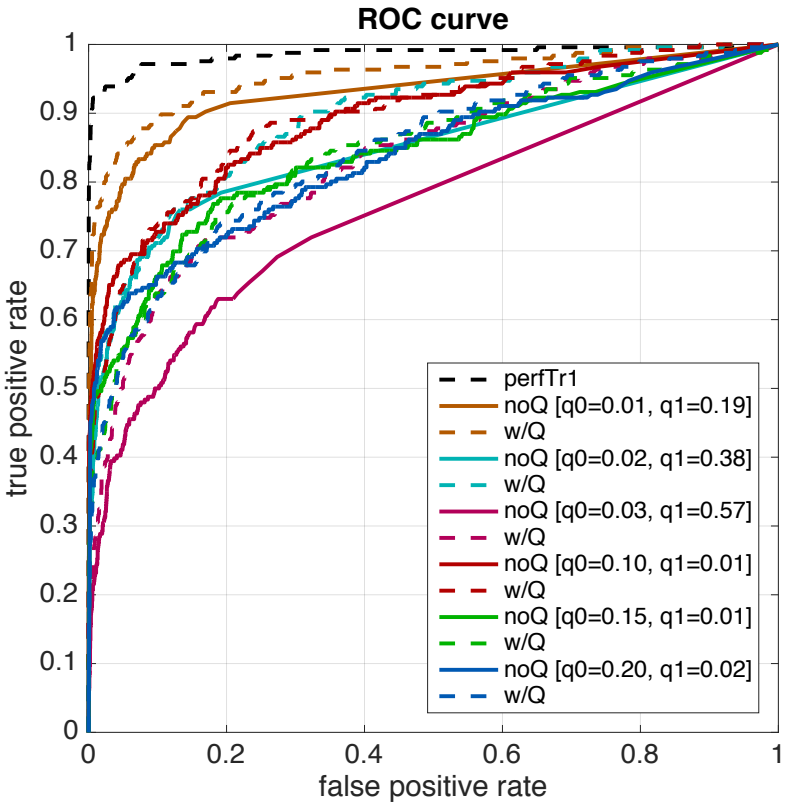
Appendices - Margins





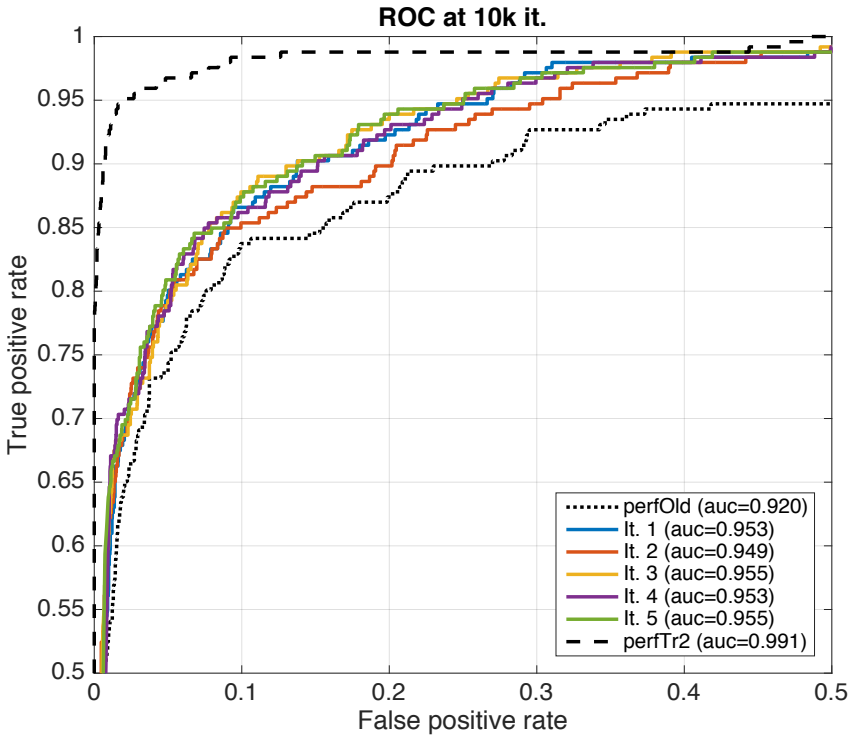
back

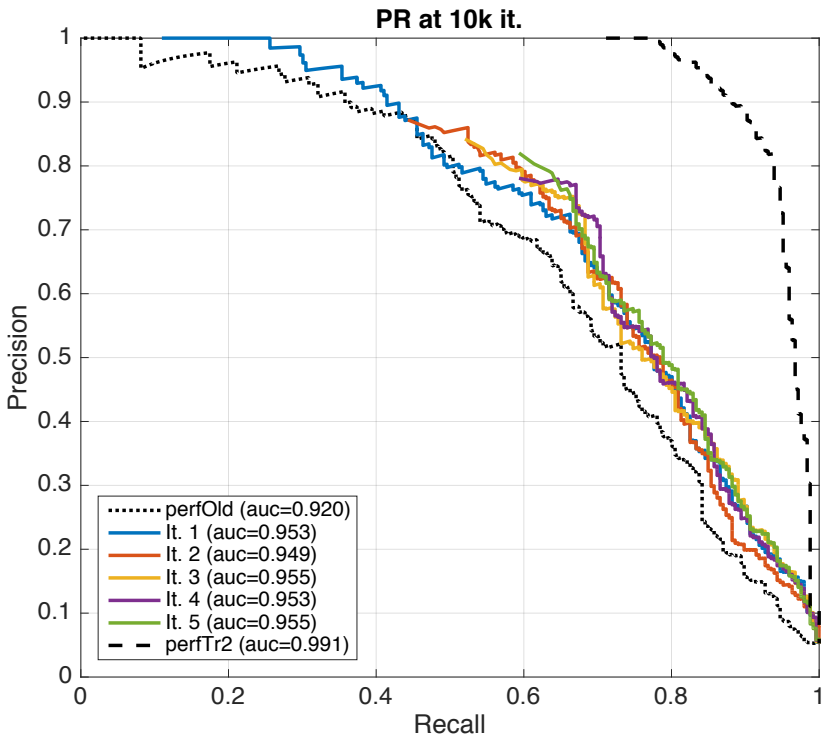
Appendices – Q matrix



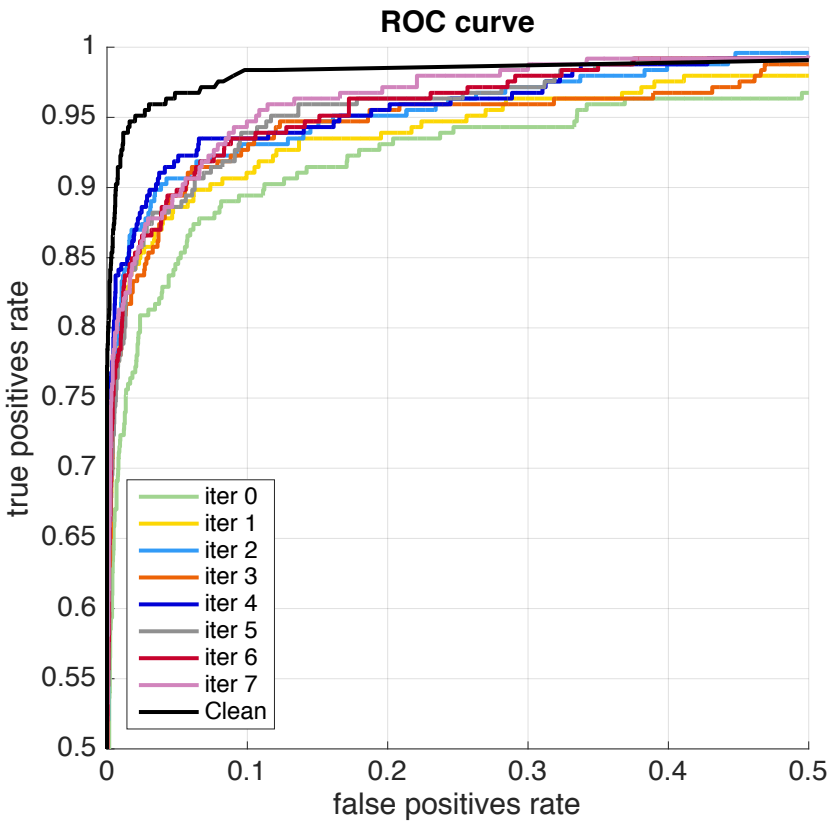
q_0	q_1	AUC
0.01	0.19	noQ: 0.9282 w/Q: 0.9552
0.02	0.38	noQ: 0.8475 w/Q: 0.8962
0.03	0.57	noQ: 0.7485 w/Q: 0.8323
0.1	0.01	noQ: 0.8927 w/Q: 0.8959
0.15	0.01	noQ: 0.8442 w/Q: 0.8455
0.2	0.02	noQ: 0.8381 w/Q: 0.8437

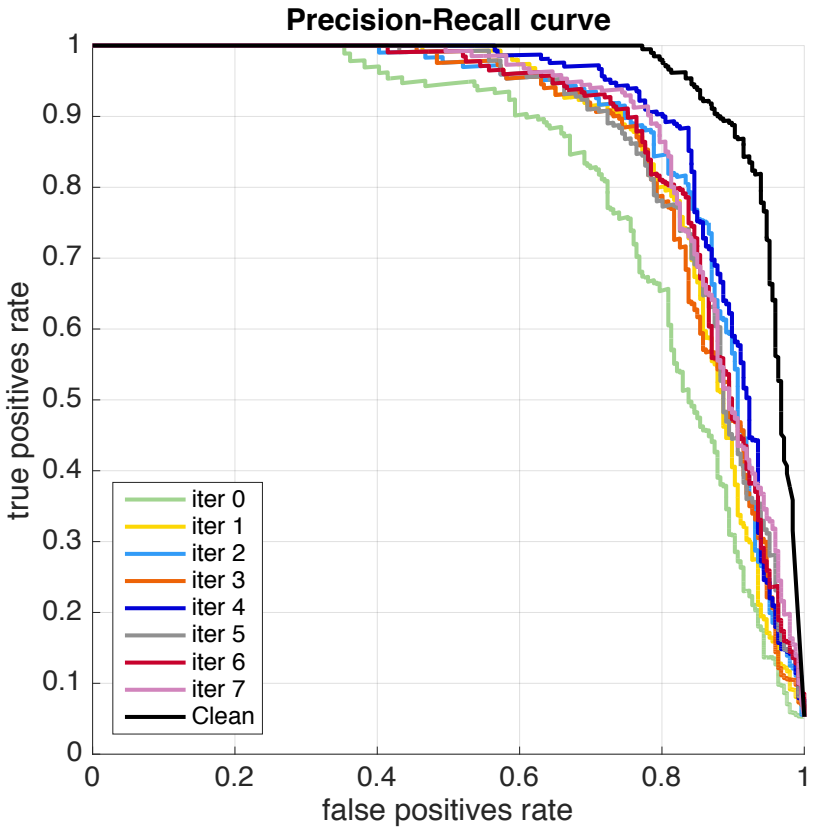
Appendices – Simple bootstrap





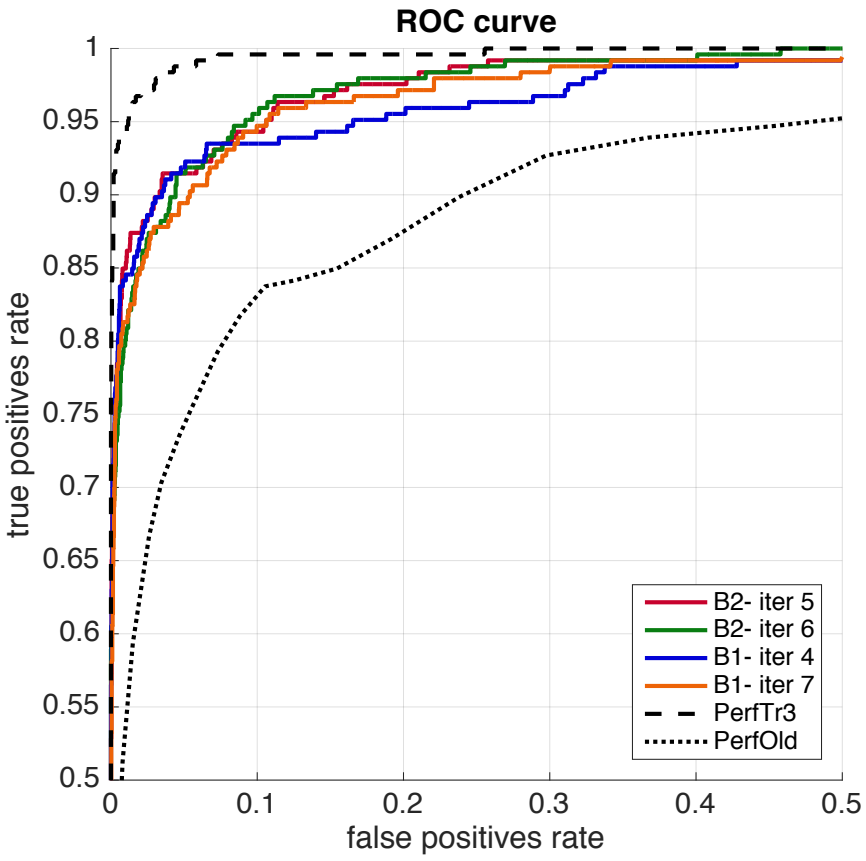
Appendices – Weighted bootstrap

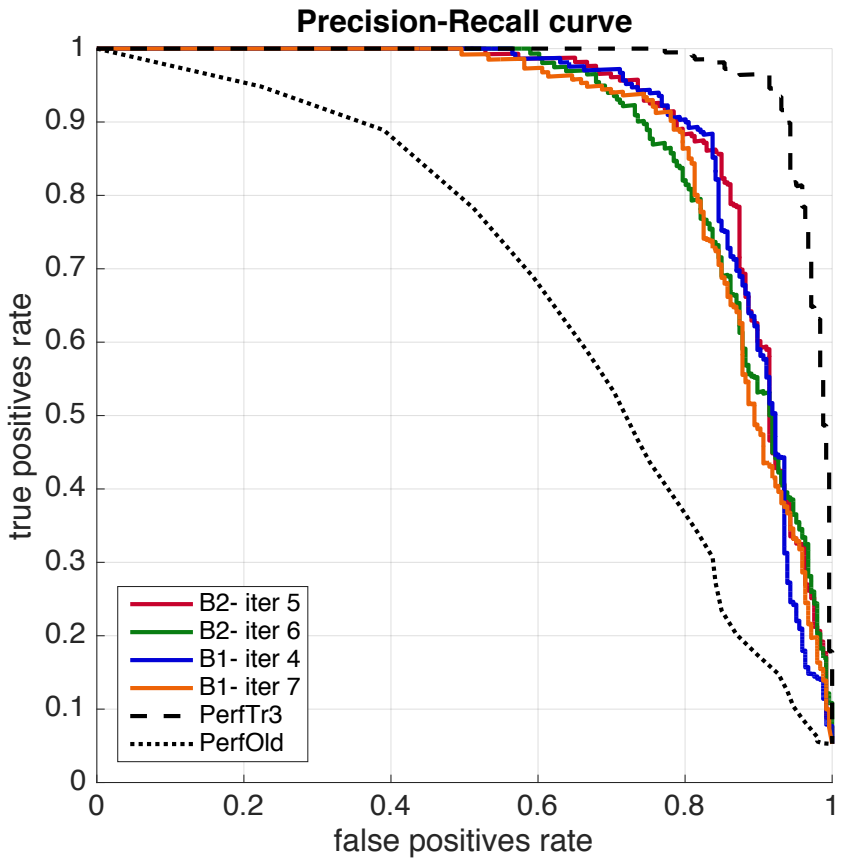




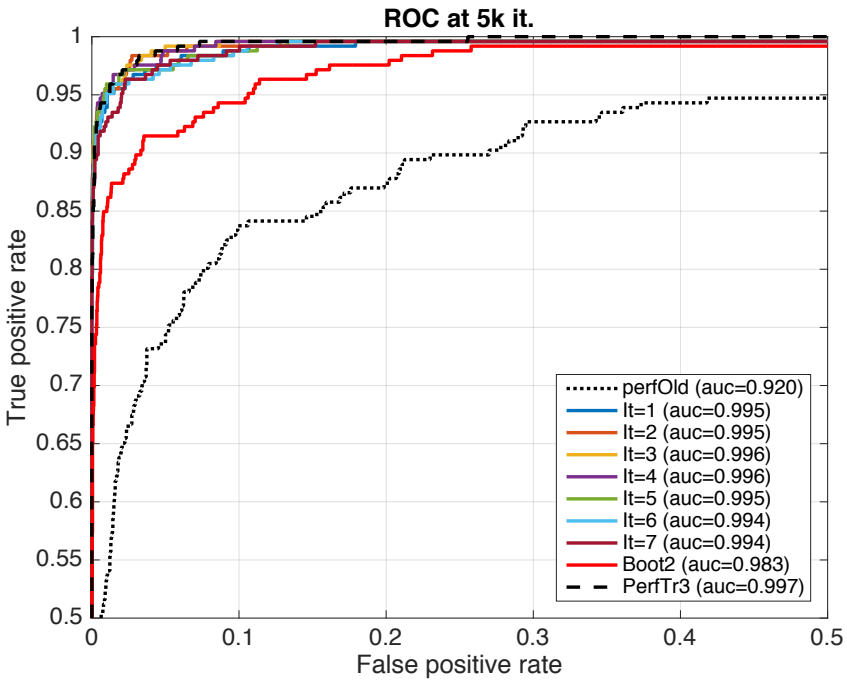
Bootstrap 1 – 2-moving average

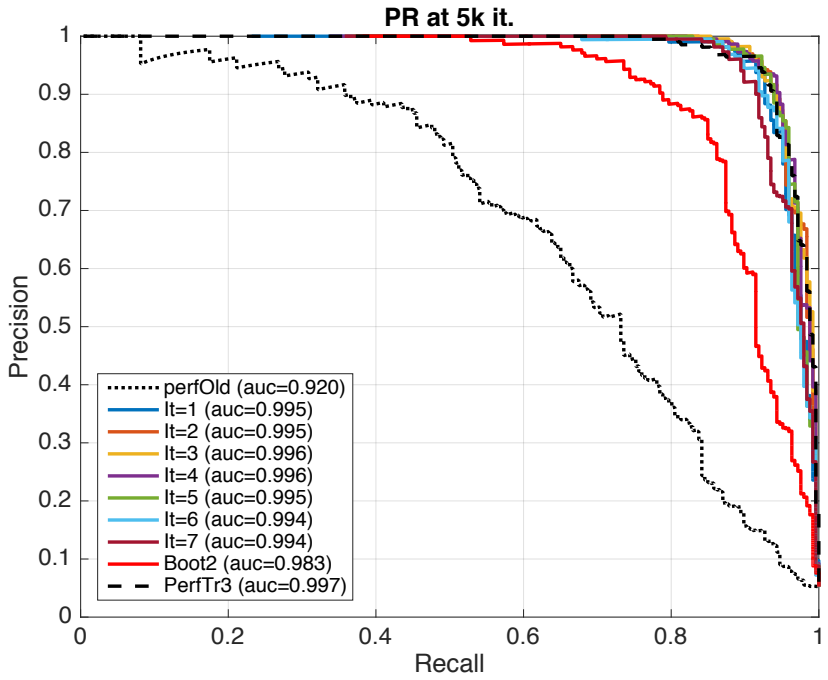
Appendices – Weighted bootstrap



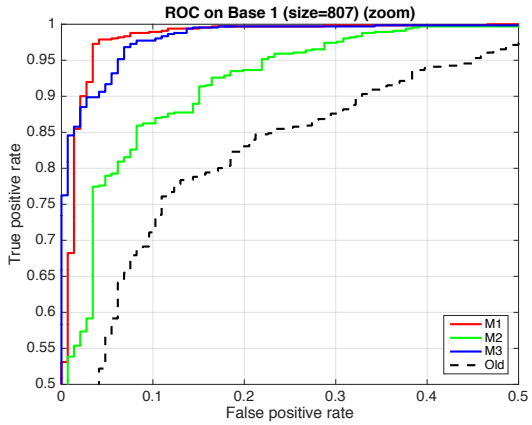


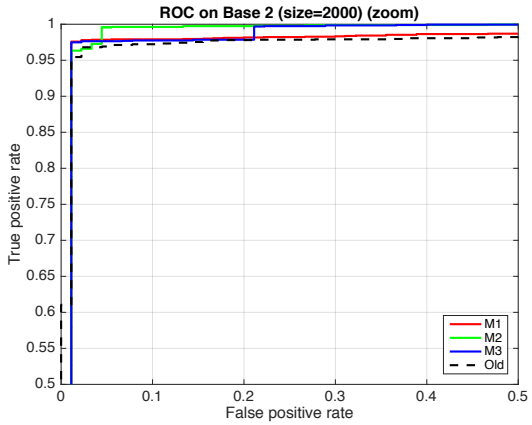
Appendices – Bootstrap extension

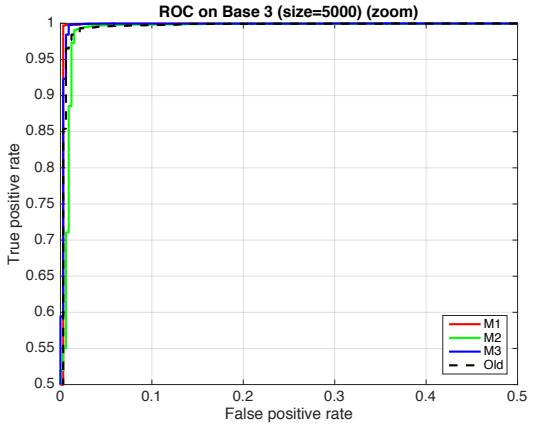


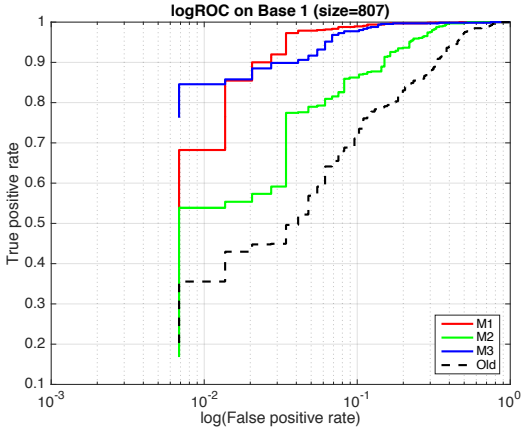


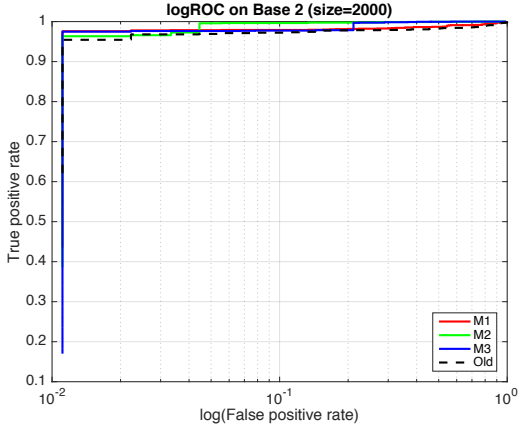
Appendices – Other test sets

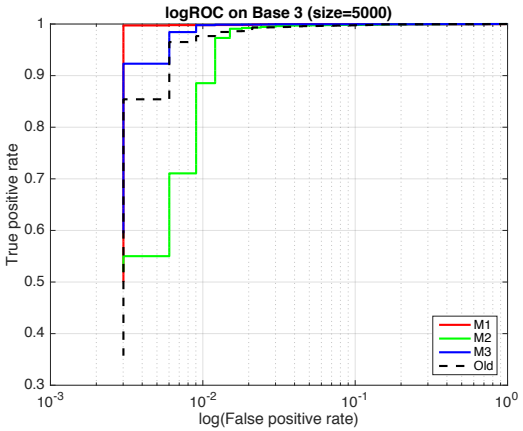




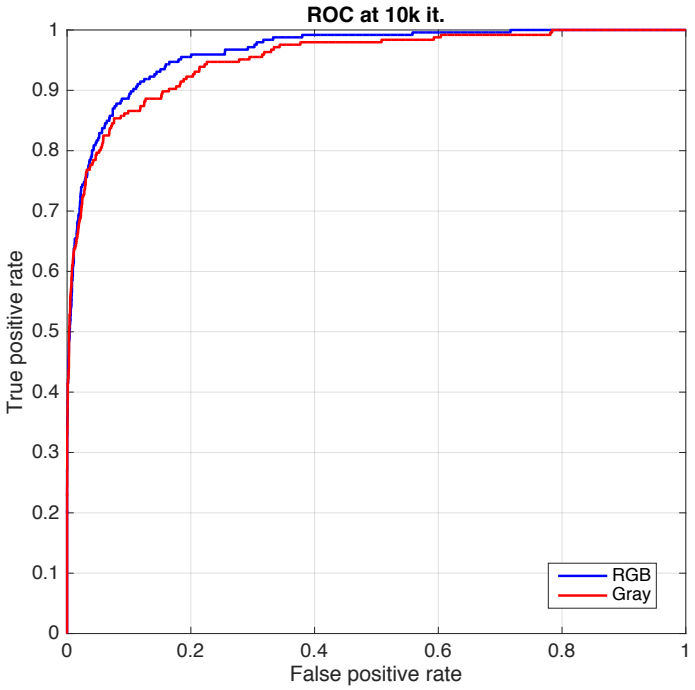


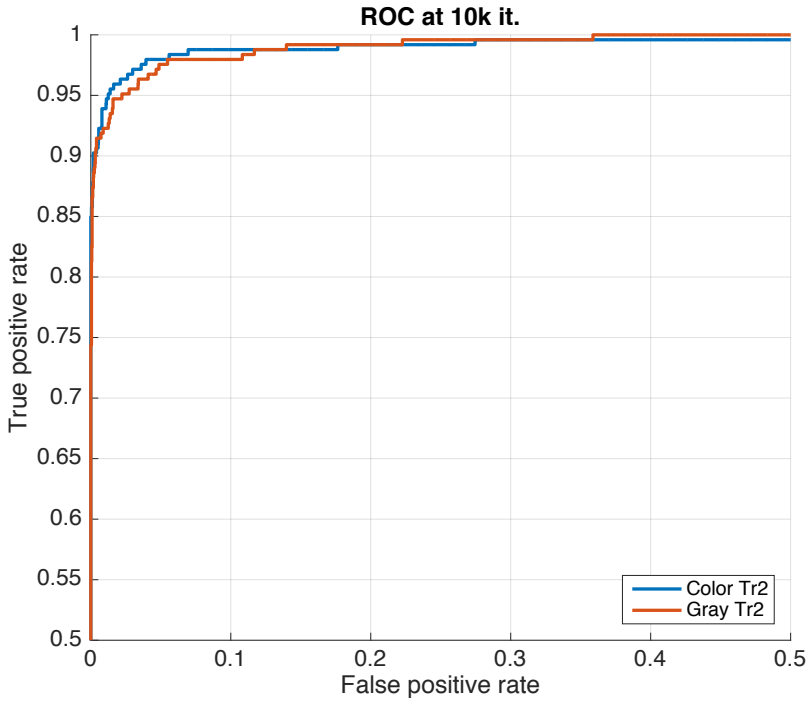




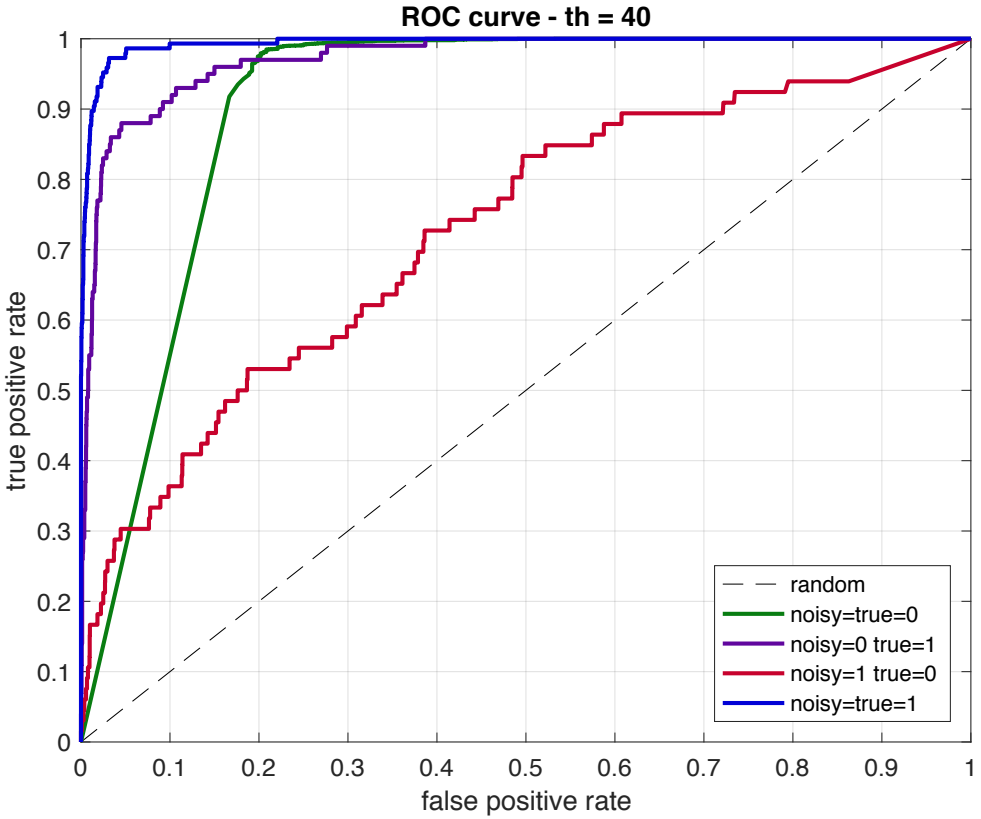


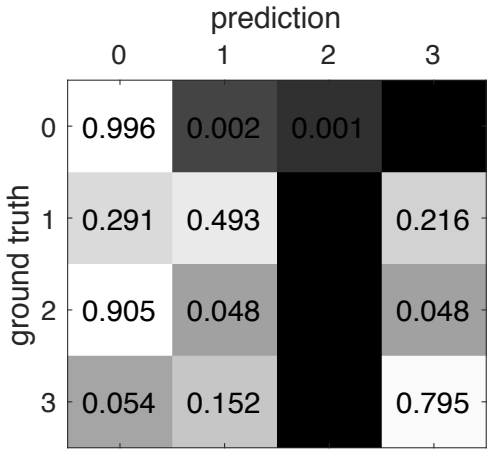
Appendices – RGB vs Grayscale





Appendices – noise prediction





Confusion matrix

back