Multimedia Data Security
Final Project Report

Paolo Chistè Claudio Facchinetti Matteo Franzil

University of Trento

January 13, 2022

Summary

1. Abstract
2. Introduction
3. Execution
   - Preparation
   - Data extraction
   - Validation
4. Results
5. Future work
6. References
7. Appendix

Abstract

Noiseprint [1] is a CNN-based method, born for tampering detection, able to extract a camera model fingerprint, resistant to scene content and enhancing model-related artifacts. In this presentation, we show the results of testing Noiseprint against a dataset [3] of out-camera images edited with modern post-processing software.

Figure: Example of a Noiseprint extraction.

Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

Problem

Noiseprint [1]'s original goal was to provide a tampering detection technique, but further studies outlined that it happened to provide acceptable results in the field of camera model identification.

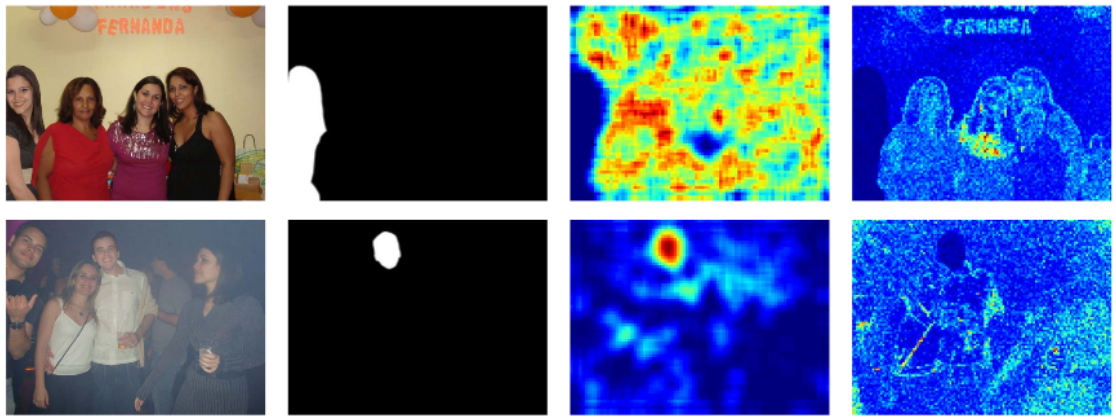


Figure: Example of tampering detection with Noiseprint.

While being a relatively new paper (2018), Noiseprint has been trained on quite old and free-from-edits image database, the Dresden database [2].

Problem

Noiseprint [1]'s original goal was to provide a tampering detection technique, but further studies outlined that it happened to provide acceptable results in the field of camera model identification.

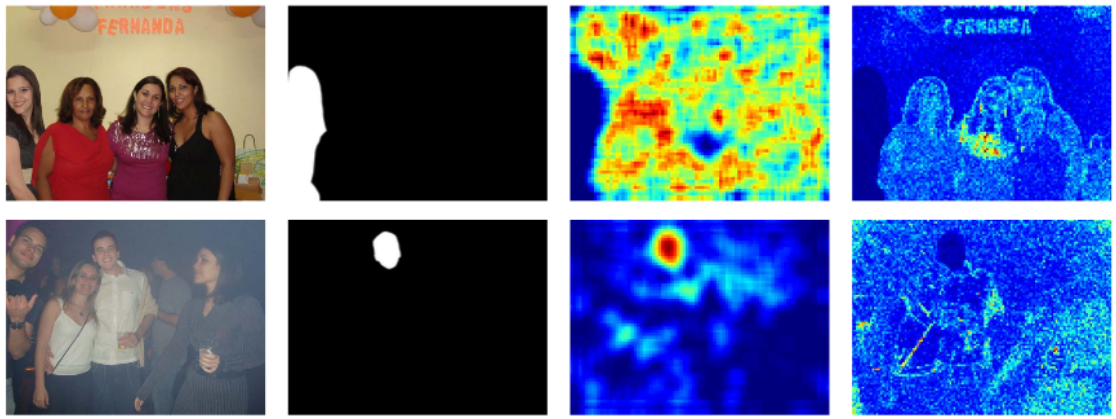


Figure: Example of tampering detection with Noiseprint.

While being a relatively new paper (2018), Noiseprint has been trained on quite old and free-from-edits image database, the Dresden database [2].

Problem

This database features in-camera photographs from models released in the late 2000s, and none of them feature any kind of image manipulation.

We aim to verify if the results on the Noiseprint paper are replicable on photos edited using image post-processing software.

Problem

This database features in-camera photographs from models released in the late 2000s, and none of them feature any kind of image manipulation.

We aim to verify if the results on the Noiseprint paper are replicable on photos edited using image post-processing software.

Dresden database

The Dresden database used in this presentation is a narrowed down variant [2].

- **9458** photos:
  - **27** camera models
  - between **80** and **1357** photos per model
  - **73** different devices
  - between **1** and **4** devices per model

Dresden database

The Dresden database used in this presentation is a narrowed down variant [2].

- **9458** photos:
  - **27** camera models
  - between **80** and **1357** photos per model
  - **73** different devices
  - between **1** and **4** devices per model

Dresden database

The Dresden database used in this presentation is a narrowed down variant [2].

- **9458** photos:
  - **27** camera models
  - between **80** and **1357** photos per model
  - **73** different devices
  - between **1** and **4** devices per model

Dresden database

The Dresden database used in this presentation is a narrowed down variant [2].

- **9458** photos:
  - **27** camera models
  - between **80** and **1357** photos per model
  - **73** different devices
  - between **1** and **4** devices per model

Outcamera database

- **821** photos shot by Andrea Montibeller [3]:
  - **20** clean images, for the fingerprint
  - the others received radial correction with **Gimp**, **PTLens**, **Adobe Photoshop**, and **Adobe Lightroom**
  - a fifth group contains photos corrected with a purposedly wrong camera profile

Outcamera database

- **821** photos shot by Andrea Montibeller [3]:
  - **20** clean images, for the fingerprint
  - the others received radial correction with **Gimp**, **PTLens**, **Adobe Photoshop**, and **Adobe Lightroom**
  - a fifth group contains photos corrected with a purposedly wrong camera profile

Outcamera database

- **821** photos shot by Andrea Montibeller [3]:
  - **20** clean images, for the fingerprint
  - the others received radial correction with **Gimp**, **PTLens**, **Adobe Photoshop**, and **Adobe Lightroom**
  - a fifth group contains photos corrected with a purposedly wrong camera profile

Outcamera database

- **821** photos shot by Andrea Montibeller [3]:
  - **20** clean images, for the fingerprint
  - the others received radial correction with **Gimp**, **PTLens**, **Adobe Photoshop**, and **Adobe Lightroom**
  - a fifth group contains photos corrected with a purposedly wrong camera profile

Expected output

We want to obtain a handful of ROC curves that show us the performance of Noiseprint on both datasets.

To obtain this, we will:

- filter the datasets to make them comparable;
- choose N pictures for each camera model $(27 + 1)$ to calculate a camera fingerprint;
- for each camera model (Dresden) / group (Outcamera) of the two datasets:
  - choose M pictures of the subset for the `H1` hypothesis;
  - choose M random pictures from other *Dresden* pictures for `H0`;
  - calculate *normalized cross-correlation* for each `H0`/`H1` picture and plot the results.

Expected output

We want to obtain a handful of ROC curves that show us the performance of Noiseprint on both datasets.

To obtain this, we will:

- filter the datasets to make them comparable;
- choose N pictures for each camera model $(27 + 1)$ to calculate a camera fingerprint;
- for each camera model (Dresden) / group (Outcamera) of the two datasets:
  - choose M pictures of the subset for the `H1` hypothesis;
  - choose M random pictures from other *Dresden* pictures for `H0`;
  - calculate *normalized cross-correlation* for each `H0`/`H1` picture and plot the results.

Expected output

We want to obtain a handful of ROC curves that show us the performance of Noiseprint on both datasets.

To obtain this, we will:

- filter the datasets to make them comparable;
- choose N pictures for each camera model $(27 + 1)$ to calculate a camera fingerprint;
- for each camera model (Dresden) / group (Outcamera) of the two datasets:
  - choose M pictures of the subset for the `H1` hypothesis;
  - choose M random pictures from other *Dresden* pictures for `H0`;
  - calculate *normalized cross-correlation* for each `H0`/`H1` picture and plot the results.

Expected output

We want to obtain a handful of ROC curves that show us the performance of Noiseprint on both datasets.

To obtain this, we will:

- filter the datasets to make them comparable;
- choose N pictures for each camera model $(27 + 1)$ to calculate a camera fingerprint;
- for each camera model (Dresden) / group (Outcamera) of the two datasets:
  - choose M pictures of the subset for the `H1` hypothesis;
  - choose M random pictures from other *Dresden* pictures for `H0`;
  - calculate *normalized cross-correlation* for each `H0`/`H1` picture and plot the results.

Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

Balancing the datasets

Data selection (Outcamera)

The dataset - with only Canon 1200D images - has

- **20** images for creating a camera fingerprint (as required) $\Rightarrow$ all fingerprints will require **20** images each.
- **801** edited images
  - divided into groups of roughly **160** pictures each

For each group (5 total):

- **160** `H1` pictures of the group itself
- **160** `H0` pictures chosen at random from the Dresden database

Data selection (Outcamera)

The dataset - with only Canon 1200D images - has

- **20** images for creating a camera fingerprint (as required) $\Rightarrow$ all fingerprints will require **20** images each.
- **801** edited images
  - divided into groups of roughly **160** pictures each

For each group (5 total):

- **160** `H1` pictures of the group itself
- **160** `H0` pictures chosen at random from the Dresden database

Data selection (Dresden)

Since our goal is to make an unbiased test, we will make this dataset uniform w.r.t. Outcamera and extract roughly **800** photos, obtaining a mini Dresden dataset.

To further make it uniform over camera models, we extract 30 photos from each camera[1].

For each model:

- **30** `H1` pictures
- **30** `H0` pictures randomly chosen from other models

[1] $30 \approx \frac{801}{27}$, where 801 is the amount of photos in the Outcamera dataset, and 27 is the number of cameras in the Dresden dataset

Data selection (Dresden)

Since our goal is to make an unbiased test, we will make this dataset uniform w.r.t. Outcamera and extract roughly **800** photos, obtaining a mini Dresden dataset.

To further make it uniform over camera models, we extract 30 photos from each camera[1].

For each model:

- **30** `H1` pictures
- **30** `H0` pictures randomly chosen from other models

[1] $30 \approx \frac{801}{27}$, where 801 is the amount of photos in the Outcamera dataset, and 27 is the number of cameras in the Dresden dataset

Data selection (Dresden)

Since our goal is to make an unbiased test, we will make this dataset uniform w.r.t. Outcamera and extract roughly **800** photos, obtaining a mini Dresden dataset.

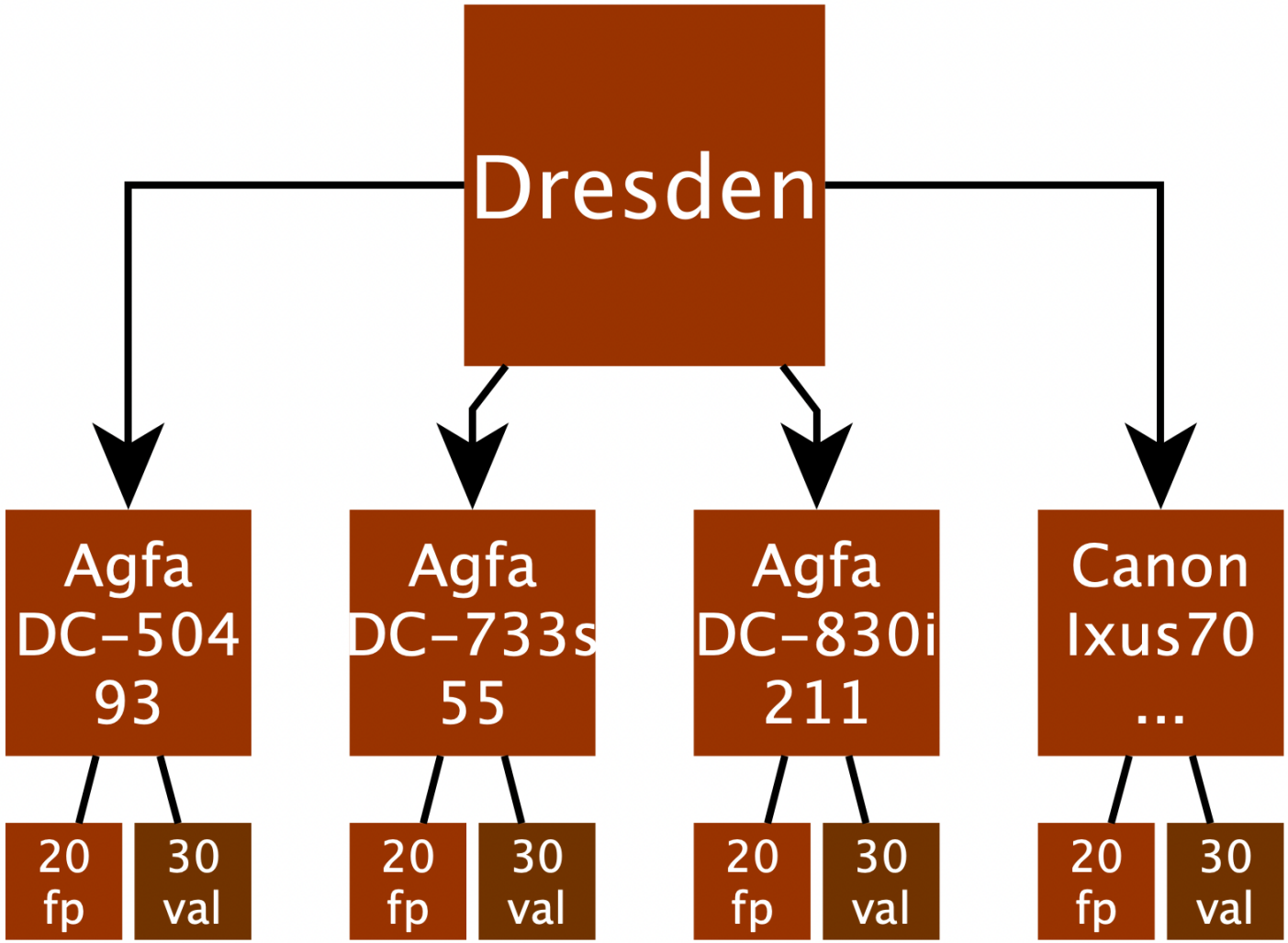
To further make it uniform over camera models, we extract 30 photos from each camera[1].

For each model:

- **30** `H1` pictures
- **30** `H0` pictures randomly chosen from other models

[1] $30 \approx \frac{801}{27}$, where 801 is the amount of photos in the Outcamera dataset, and 27 is the number of cameras in the Dresden dataset

Data selection



Camera fingerprints

1. verify that all chosen images from the same camera have same size
2. use a parallel script for extracting each Noiseprint (for convenience, we extract **all** of them)

```
python3 noiseprint/main_extraction.py "$image"
    "$OUTDIR/${image//\.JPG/}.mat"
```

$\Rightarrow$ the fingerprint of each camera model is the average of its 20 image Noiseprints

Camera fingerprints

1. verify that all chosen images from the same camera have same size
2. use a parallel script for extracting each Noiseprint (for convenience, we extract **all** of them)

```
python3 noiseprint/main_extraction.py "$image"
    "$OUTDIR/${image//\.JPG/}.mat"
```

$\Rightarrow$ the fingerprint of each camera model is the average of its 20 image Noiseprints

Cross-correlating images

With the `.mat` files ready, for each fingerprint *FP* another script:

- selects the `H1` images available and randomly chooses K images for the `H0` case[2];
- obtains a 1024×1024 central crop of each image;
- computes *ncc* for each `H1` and `H0` image against *FP*;
- dumps the results to a JSON for later analysis.

[2] 30 for Dresden, 160 for Outcamera

Cross-correlating images

With the `.mat` files ready, for each fingerprint FP another script:

- selects the `H1` images available and randomly chooses K images for the `H0` case[2];
- obtains a 1024×1024 central crop of each image;
- computes *ncc* for each `H1` and `H0` image against FP;
- dumps the results to a JSON for later analysis.

[2] 30 for Dresden, 160 for Outcamera

Cross-correlating images

With the `.mat` files ready, for each fingerprint FP another script:

- selects the `H1` images available and randomly chooses K images for the `H0` case[2];
- obtains a 1024×1024 central crop of each image;
- computes *ncc* for each `H1` and `H0` image against FP;
- dumps the results to a JSON for later analysis.

[2] 30 for Dresden, 160 for Outcamera

Cross-correlating images

With the `.mat` files ready, for each fingerprint FP another script:

- selects the `H1` images available and randomly chooses K images for the `H0` case[2];
- obtains a 1024×1024 central crop of each image;
- computes *ncc* for each `H1` and `H0` image against FP;
- dumps the results to a JSON for later analysis.

[2] 30 for Dresden, 160 for Outcamera

Cross-correlating images

With the `.mat` files ready, for each fingerprint FP another script:

- selects the `H1` images available and randomly chooses K images for the `H0` case[2];
- obtains a 1024×1024 central crop of each image;
- computes *ncc* for each `H1` and `H0` image against FP;
- dumps the results to a JSON for later analysis.

[2] 30 for Dresden, 160 for Outcamera

Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

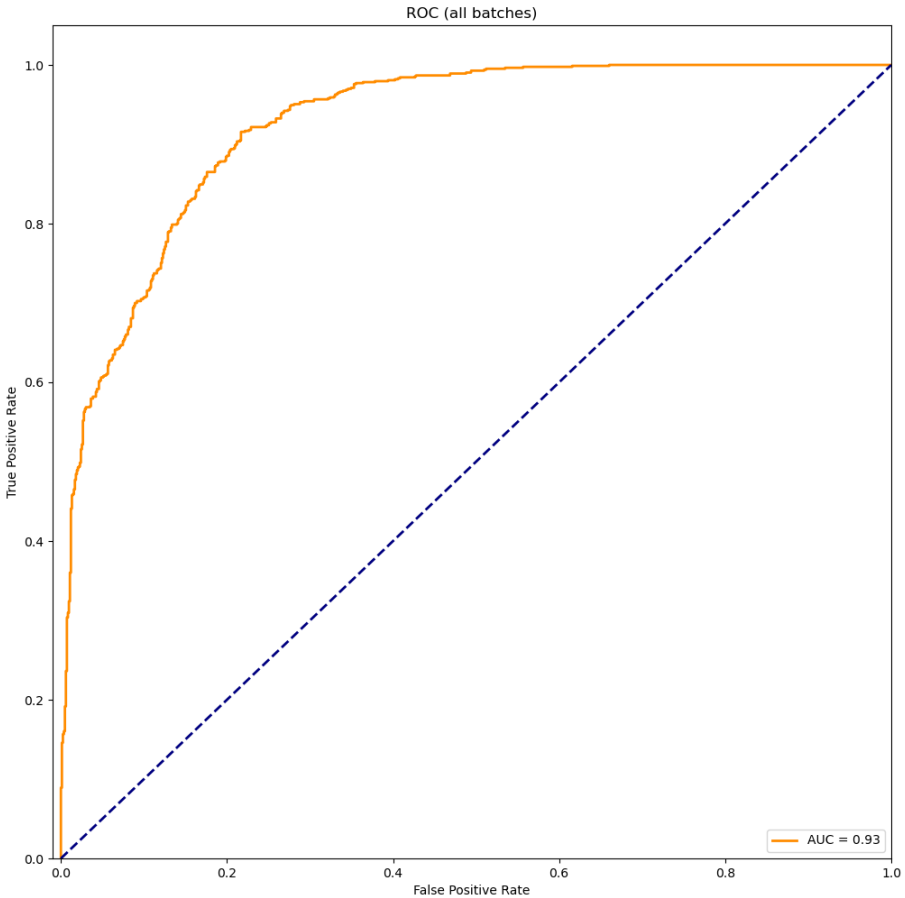
4 Results

5 Future work

6 References

7 Appendix

ROC curves



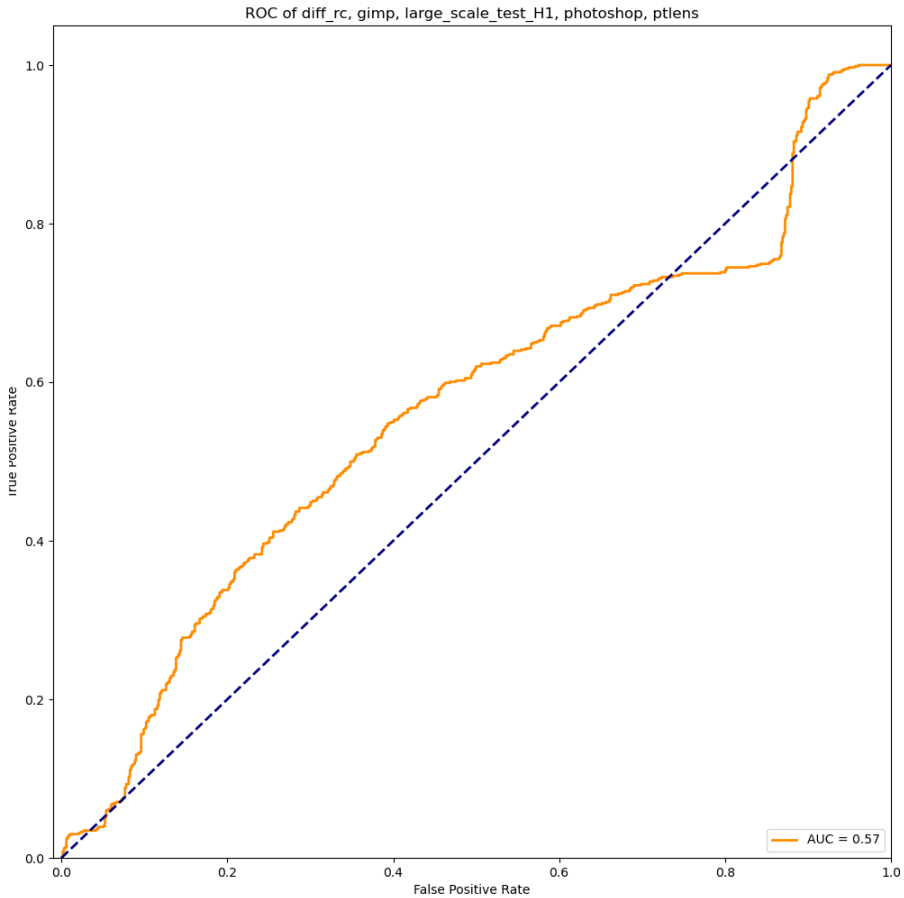
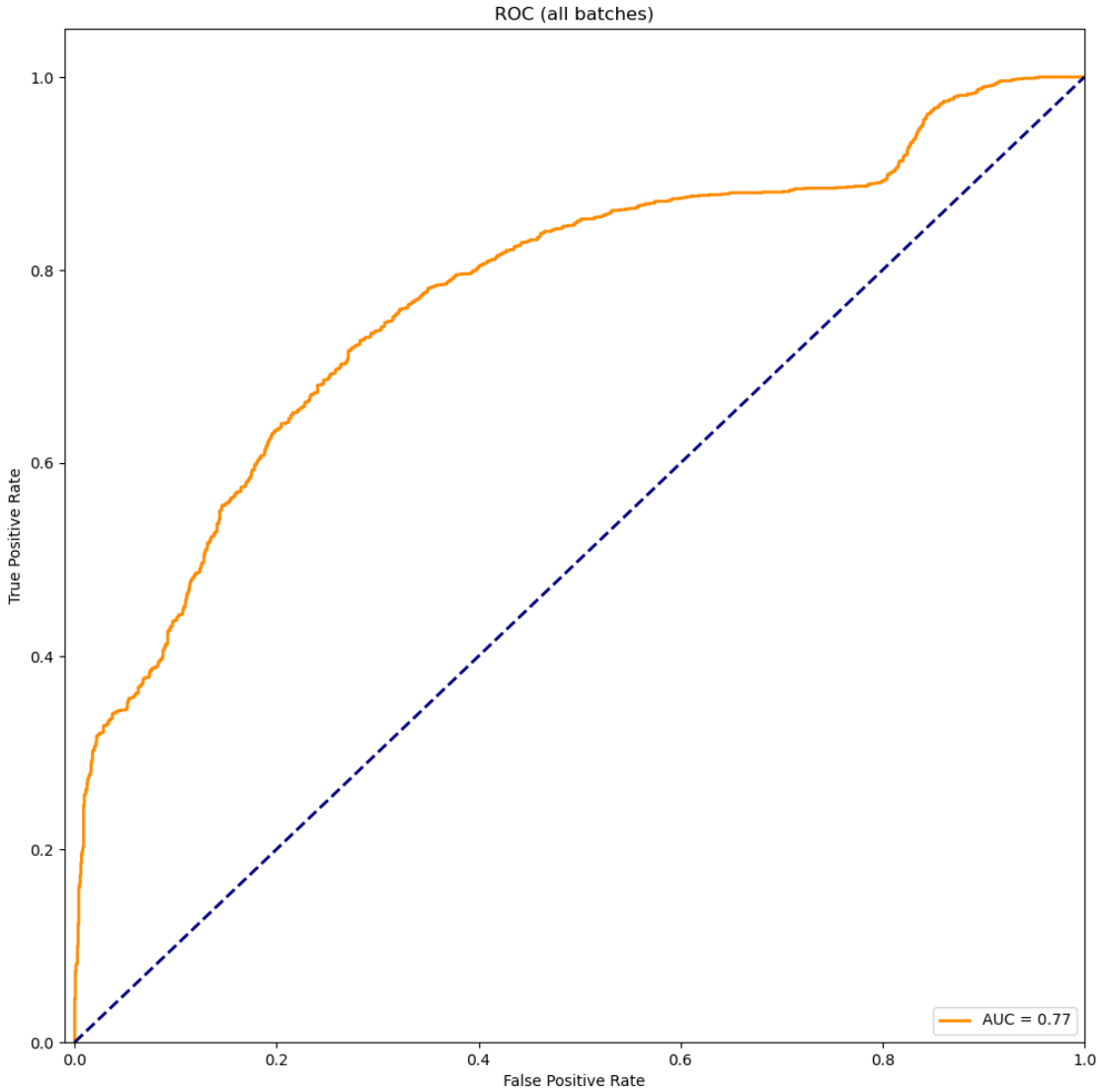
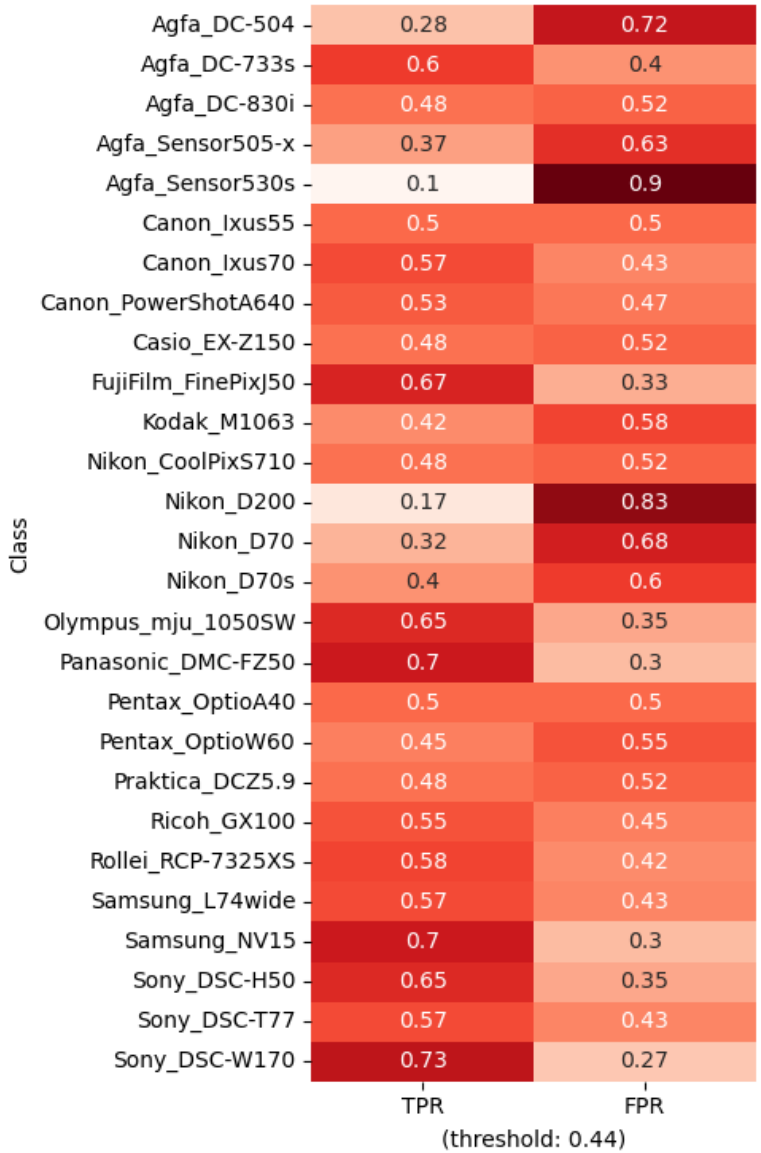


Figure: ROC curves for the Dresden and Outcamera dataset.
Left: $AUC = 0.91$; Right: $AUC = 0.57$

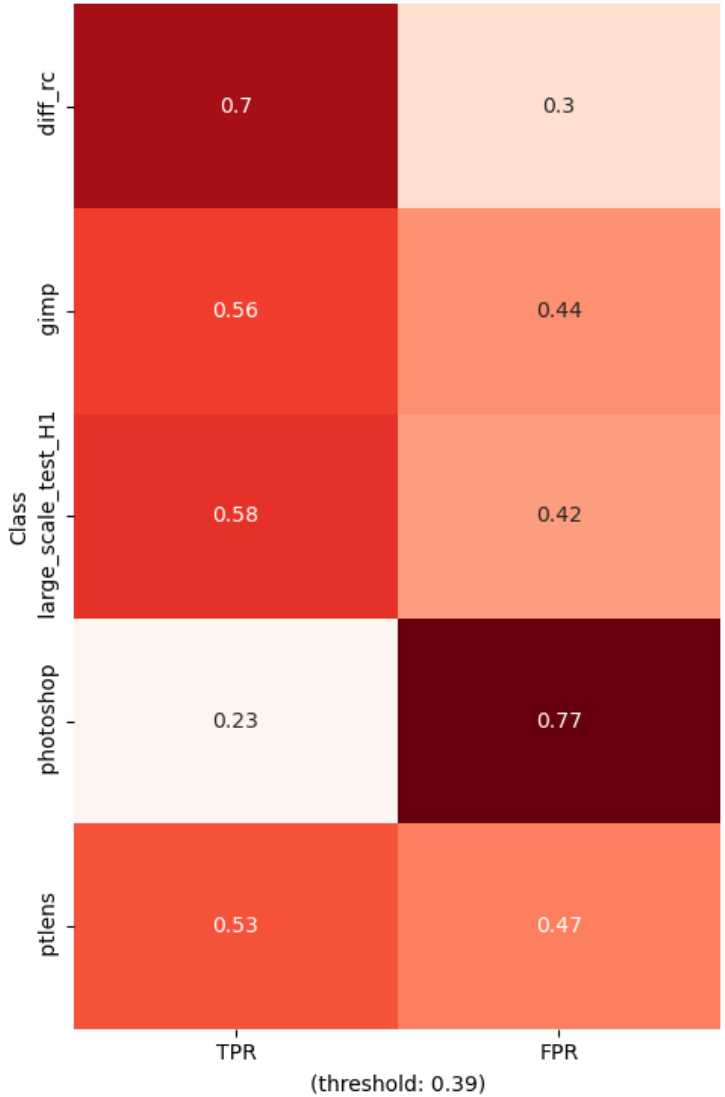
ROC curve (both datasets; $AUC = 0.77$)



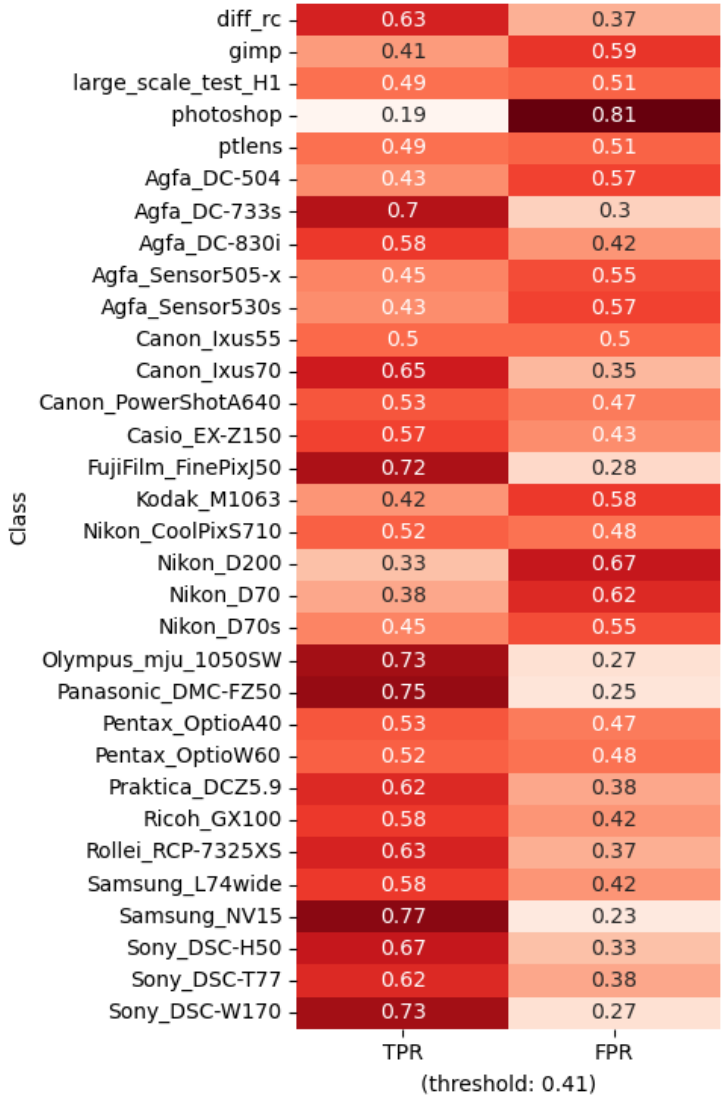
Tables - TPR/FPR Dresden



Tables - TPR/FPR Outcamera



Tables - TPR/FPR combined



Setting the thresholds

Experiment	0.2 FPR	0.1 FPR	0.05 FPR	Optimal
`dresden`	0.4207	0.4941	0.5294	0.4411 (0.1630 FPR)
`outcamera`	0.4490	0.4993	0.5221	0.3864 (0.4324 FPR)
`all`	0.4364	0.4951	0.5280	0.4081 (0.2785 FPR)

Table: Alternative thresholds, chosen using the *set FPR* method, and the optimal one, chosen with the *minimum ROC distance* method [4]

.

Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

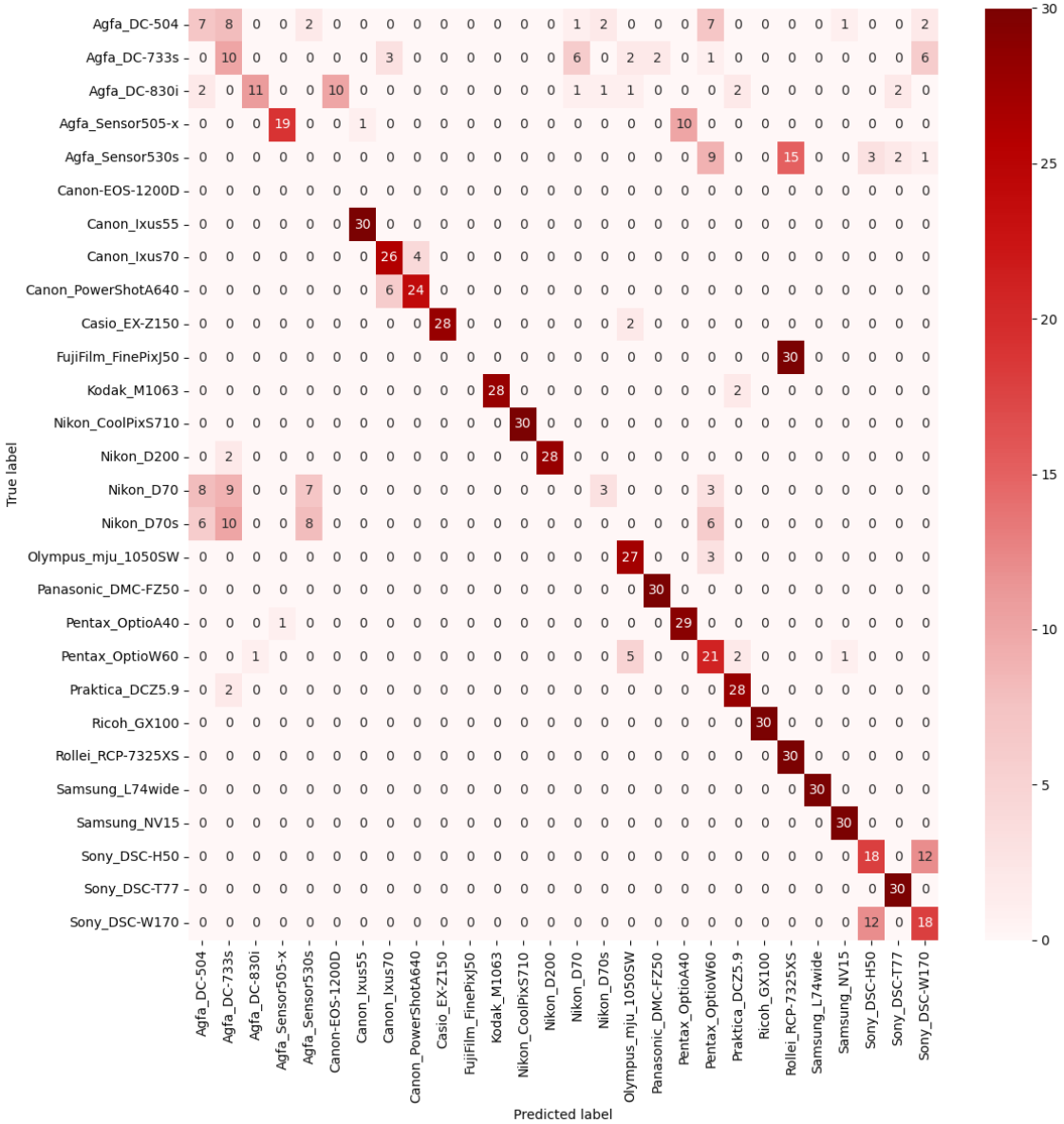
Multiclass classification

The availability of a large number of fingerprints means that Noiseprint translates decently in a multiclass classification scenario.

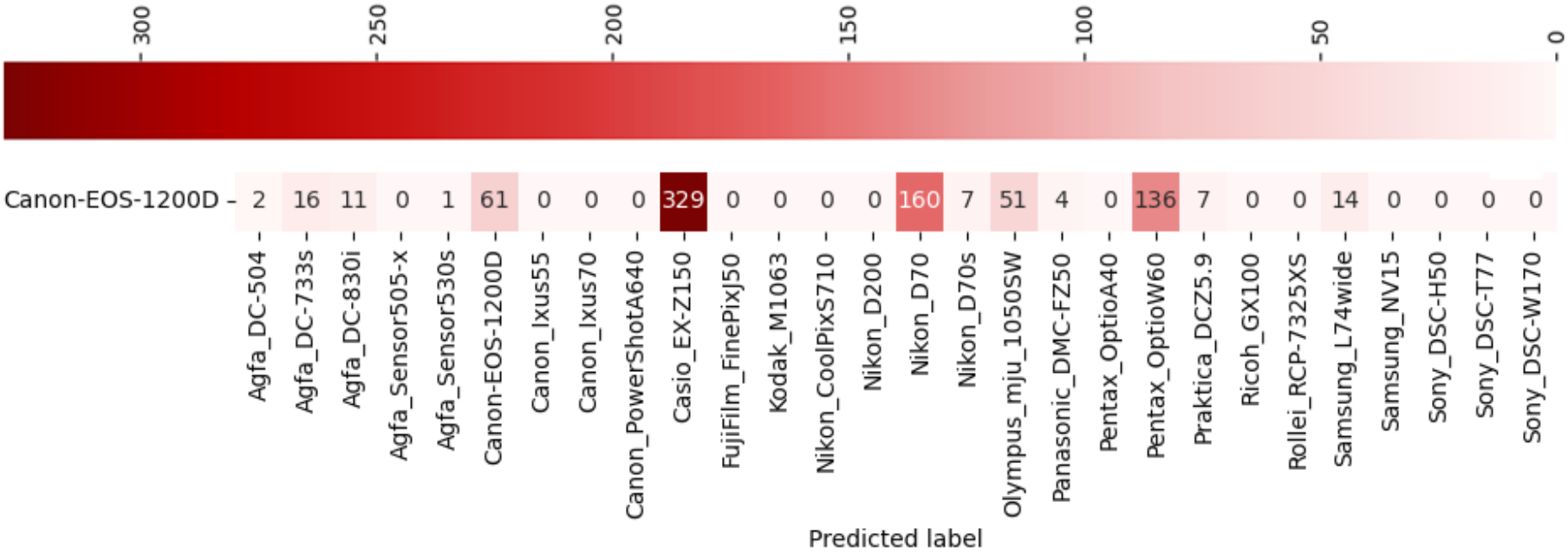
Being trained on the Dresden database, Noiseprint performs accurately for some models and terribly for others, coherently with the paper results.

A larger and more diverse dataset could allow Noiseprint to be re-trained on newer camera models.

Confusion matrix - Dresden



Confusion vector - Canon EOS 1200D



Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

References


D. Cozzolino and L. Verdoliva, "Noiseprint: a CNN-based camera model fingerprint," 2018.


T. Gloe and R. Böhme, "The 'Dresden Image Database' for Benchmarking Digital Image Forensics," in *Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1584–1590. [Online]. Available: https://doi.org/10.1145/1774088.1774427


A. Montibeller, "Out-Camera Dataset, with Canon EOS 1200D photos," 2017.


Stack Overflow, "Roc curve and cut off point. Python," 2015. [Online]. Available: https://stackoverflow.com/questions/28719067/roc-curve-and-cut-off-point-python?answertab=votes#tab-top

Summary

1 Abstract

2 Introduction

3 Execution
- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

Extra: code for reordering photos

We can sort the database with these commands.

```
ls | sed -r "s|^(.*)_([0-9]*)_(.*) mv \1_\2_\3 \1|g" \
   > ../move.sh

ls | sed -r "s/^(.*)_([0-9]*)_.*/\1/g" \
   | uniq | xargs mkdir -p

../move.sh
```

We can now get the amount of photos per folder:

```
for d in *; do cd $d  ls | wc -l  cd ..; done
```

Extra: code for selecting random photos

To select 20 random pictures:

```
for d in *; do cd $d  ls | shuf | head -n 20
          | xargs -I {} /bin/mv {} ../..  cd ..; done
```

Extra: output of verify-all-sizes.sh

```
for image in *; do
    exif=$(exiftool $image)
    if [[ ctr -eq 0 ]]; then
        width=$(echo "$exif" | grep -E 'Width' | grep 'Exif' | cut -d ':' -f 2 | cut -d ' ' -f 2)
        height=$(echo "$exif" | grep -E 'Height' | grep 'Exif' | cut -d ':' -f 2 | cut -d ' ' -f 2)
        echo "Base: $width x $height"
    else
        __width=$(echo "$exif" | grep -E 'Width' | grep 'Exif' | cut -d ':' -f 2 | cut -d ' ' -f 2)
        __height=$(echo "$exif" | grep -E 'Height' | grep 'Exif' | cut -d ':' -f 2 | cut -d ' ' -f 2)
        if [[ $width -ne $__width ]] || [[ $height -ne $__height ]]; then
            echo "Image $image has different dimensions than the first image"
            echo "First image: $width x $height"
            echo "Current image: $__width x $__height"
            echo "Fix this..."
            break
        fi
    fi
    ctr=$((ctr+1))
done%
```

Extra: example of JSON output


```
    "Agfa_DC-504_0_71.mat": {
      "template": {
        "name": "/Users/matte/Downloads/mds/fingerprints/Agfa_DC-504.mat",
        "width": 4032,
        "height": 3024
      },
      "image": {
        "name": "/Volumes/Extreme SSD/_move/dataset-dresden-validation-noise/
Agfa_DC-504/Agfa_DC-504_0_71.mat",
        "width": 4032,
        "height": 3024
      },
      "pce": 7.071446514934132,
      "ncc": 0.44186145067214966,
      "distance": 566.3068237304688
    }
```