

Multimedia Data Security

Final Project Report

Paolo Chistè

Claudio Facchinetti

Matteo Franzil

University of Trento

January 1, 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.

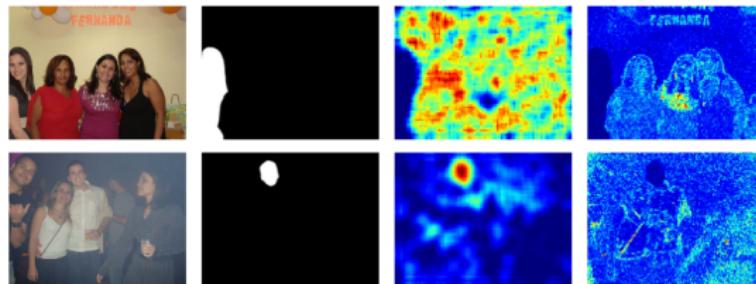


Figure: Example of tampering detection with Noiseprint.

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.

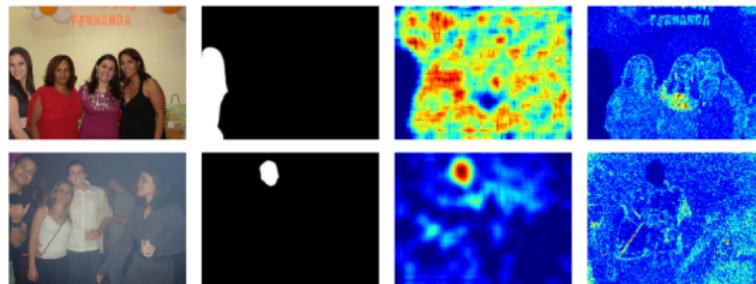


Figure: Example of tampering detection with Noiseprint.

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 batch contains photos corrected with a purportedly 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 batch contains photos corrected with a purportedly 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 batch contains photos corrected with a purportedly 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 batch contains photos corrected with a purportedly 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) / batch (Outcamera) of the two datasets:
 - ▶ choose M pictures of the subset for the H_1 hypothesis;
 - ▶ choose M random pictures from other *Dresden* pictures for H_0 ;
 - ▶ calculate *normalized cross-correlation* for each H_0/H_1 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) / batch (Outcamera) of the two datasets:
 - ▶ choose M pictures of the subset for the H_1 hypothesis;
 - ▶ choose M random pictures from other *Dresden* pictures for H_0 ;
 - ▶ calculate *normalized cross-correlation* for each H_0/H_1 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) / batch (Outcamera) of the two datasets:
 - ▶ choose M pictures of the subset for the H_1 hypothesis;
 - ▶ choose M random pictures from other *Dresden* pictures for H_0 ;
 - ▶ calculate *normalized cross-correlation* for each H_0/H_1 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) / batch (Outcamera) of the two datasets:
 - ▶ choose M pictures of the subset for the H_1 hypothesis;
 - ▶ choose M random pictures from other *Dresden* pictures for H_0 ;
 - ▶ calculate *normalized cross-correlation* for each H_0/H_1 picture and plot the results.

Summary

1 Abstract

2 Introduction

3 Execution

- Preparation
- Data extraction
- Validation

4 Results

5 Future work

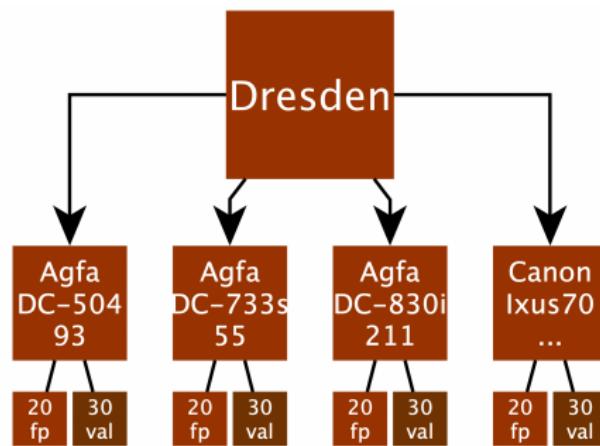
6 References

7 Appendix

Data selection

For each of the two datasets:

- sort them
- sort the amount of photos in each subfolder
- define subsets for fingerprints and validation



Data selection (Outcamera)

The dataset - with only Canon 1200D images - has:

- **20** images for creating a camera fingerprint
- **801** edited images
 - ▶ divided into equal batches of **160** pictures each

⇒ all fingerprints require **20** pictures.

For each batch (5 total):

- **160** H1 pictures of the batch 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
- **801** edited images
 - ▶ divided into equal batches of **160** pictures each

⇒ all fingerprints require **20** pictures.

For each batch (5 total):

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

Data selection (Dresden)

In order to make Dresden uniform, we need to extract x photos from each camera s.t. $27x \leq 801^1$.

⇒ all fingerprints use **20** pictures as before.

For each model (27 total):

- **30** H1 pictures ($30 \approx \frac{801}{27}$)
- **30** H0 pictures chosen from other models

⇒ we get a mini Dresden dataset of 810 pictures.

¹where 27 is the number of cameras in the Dresden dataset, and 801 is the total amount of photos remaining in the Outcamera dataset

Data selection (Dresden)

In order to make Dresden uniform, we need to extract x photos from each camera s.t. $27x \leq 801^1$.

⇒ all fingerprints use **20** pictures as before.

For each model (27 total):

- **30** H1 pictures ($30 \approx \frac{801}{27}$)
- **30** H0 pictures chosen from other models

⇒ we get a mini Dresden dataset of 810 pictures.

¹where 27 is the number of cameras in the Dresden dataset, and 801 is the total amount of photos remaining in the Outcamera dataset

Camera fingerprints

- ➊ verify image sizes with `verify-all-sizes.sh`
- ➋ use a dedicated parallelized script for extracting each Noiseprint (both fingerprint and validation sets)

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

⇒ once finished, we calculate each model's average (with a simple mean).

Camera fingerprints

- ① verify image sizes with `verify-all-sizes.sh`
- ② use a dedicated parallelized script for extracting each Noiseprint (both fingerprint and validation sets)

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

⇒ once finished, we calculate each model's average (with a simple mean).

Cross-correlating images

With the .mat files ready, another parallel script:

- pulls the directory of each camera fingerprint
- for each fingerprint:
 - ▶ enumerates the H1 images available and randomizes K more² for the H0 case;
 - ▶ obtains a 1024×1024 central crop of each image;
 - ▶ runs automated tests for each H1 and H0 image against the current template;
 - ▶ dumps the results to a JSON;
 - ▶ saves the log files for later use.

²30 for Dresden, 160 for Outcamera

Cross-correlating images

With the .mat files ready, another parallel script:

- pulls the directory of each camera fingerprint
- for each fingerprint:
 - ▶ enumerates the H1 images available and randomizes K more² for the H0 case;
 - ▶ obtains a 1024×1024 central crop of each image;
 - ▶ runs automated tests for each H1 and H0 image against the current template;
 - ▶ dumps the results to a JSON;
 - ▶ saves the log files for later use.

²30 for Dresden, 160 for Outcamera

Cross-correlating images

With the .mat files ready, another parallel script:

- pulls the directory of each camera fingerprint
- for each fingerprint:
 - ▶ enumerates the H1 images available and randomizes K more² for the H0 case;
 - ▶ obtains a 1024×1024 central crop of each image;
 - ▶ runs automated tests for each H1 and H0 image against the current template;
 - ▶ dumps the results to a JSON;
 - ▶ saves the log files for later use.

²30 for Dresden, 160 for Outcamera

Example of 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
}
```

Summary

1 Abstract

2 Introduction

3 Execution

- Preparation
- Data extraction
- Validation

4 Results

5 Future work

6 References

7 Appendix

ROC curves

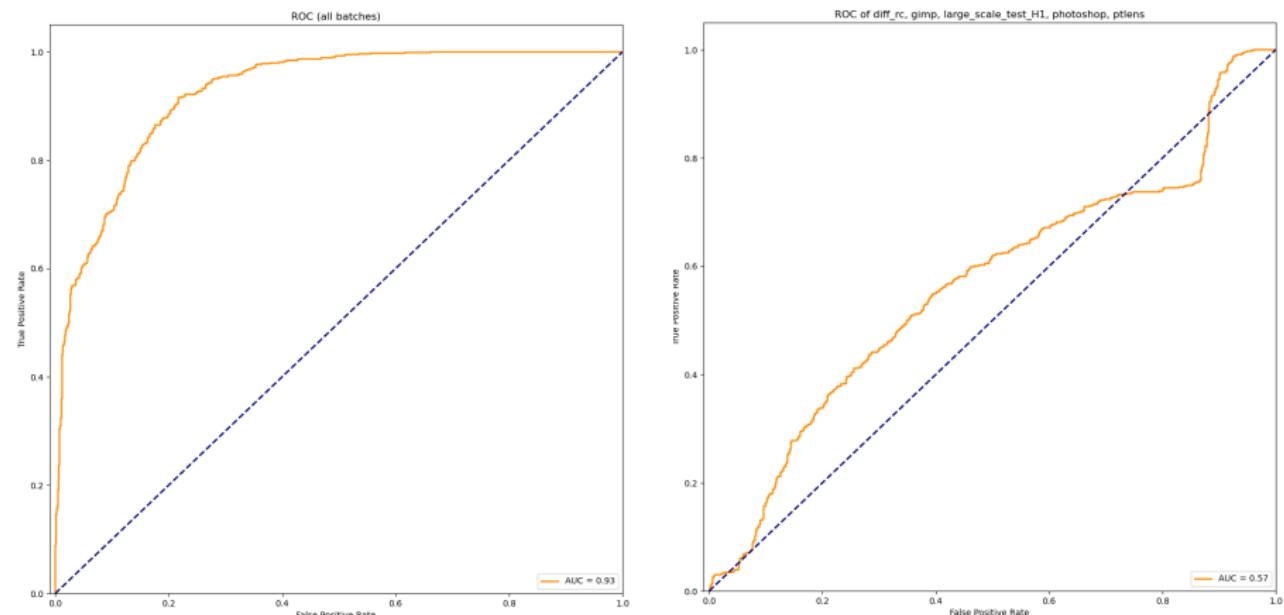
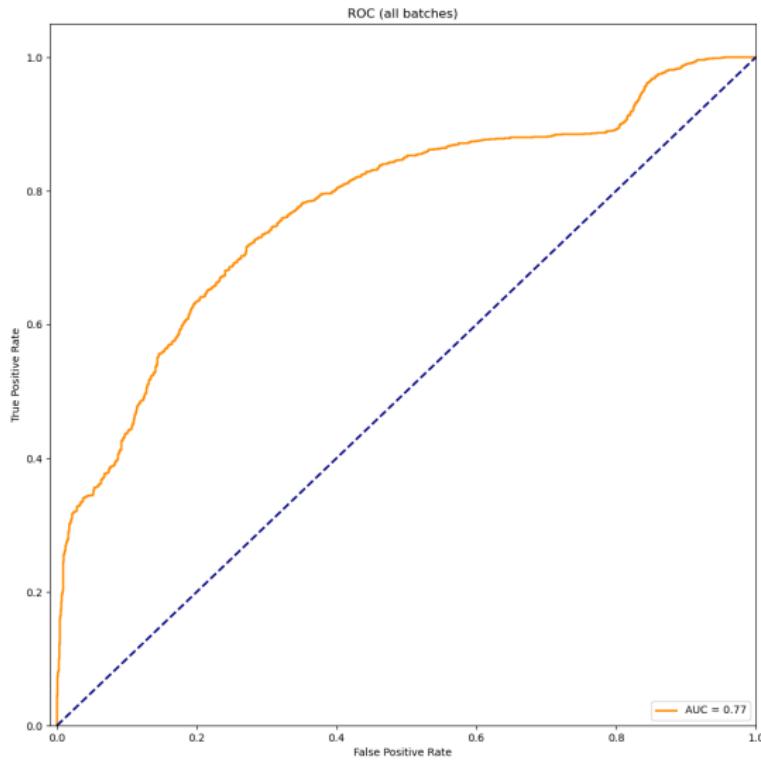


Figure: ROC curves for the Dresden and Outcamera dataset.
Left: $AUC = 0.91$; Right: $AUC = 0.57$

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

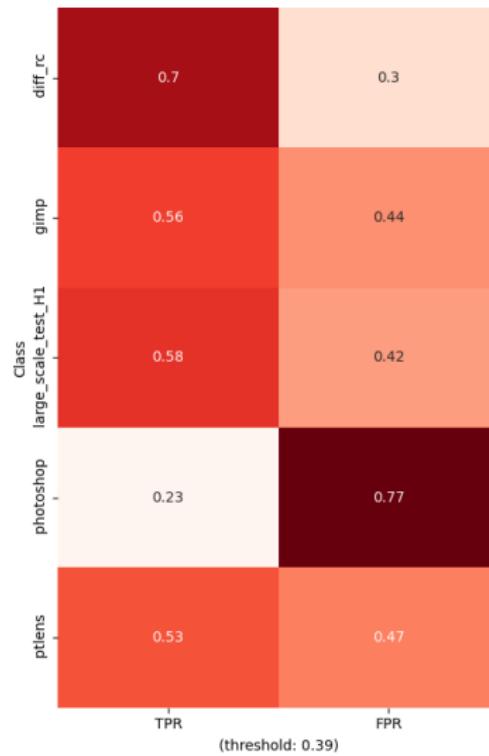


Tables - TPR/FPR Dresden

	TPR	FPR
Agfa_DC-504 -	0.28	0.72
Agfa_DC-733s -	0.6	0.4
Agfa_DC-830i -	0.48	0.52
Agfa_Sensor505-x -	0.37	0.63
Agfa_Sensor530s -	0.1	0.9
Canon_Ixus55 -	0.5	0.5
Canon_Ixus70 -	0.57	0.43
Canon_PowerShotA640 -	0.53	0.47
Casio_EX-Z150 -	0.48	0.52
FujiFilm_FinePixJ50 -	0.67	0.33
Kodak_M1063 -	0.42	0.58
Nikon_CoolpixS710 -	0.48	0.52
Nikon_D200 -	0.17	0.83
Nikon_D70 -	0.32	0.68
Nikon_D70s -	0.4	0.6
Olympus_mju_1050SW -	0.65	0.35
Panasonic_DMC-FZ50 -	0.7	0.3
Pentax_OptioA40 -	0.5	0.5
Pentax_OptioW60 -	0.45	0.55
Praktica_DCZ5.9 -	0.48	0.52
Ricoh_GX100 -	0.55	0.45
Rollei_RCP-7325XS -	0.58	0.42
Samsung_L74wide -	0.57	0.43
Samsung_NV15 -	0.7	0.3
Sony_DSC-H50 -	0.65	0.35
Sony_DSC-T77 -	0.57	0.43
Sony_DSC-W170 -	0.73	0.27

(threshold: 0.44)

Tables - TPR/FPR Outcamera



Tables - TPR/FPR combined

	TPR	FPR
diff_rc	0.63	0.37
gimp	0.41	0.59
large_scale_test_H1	0.49	0.51
photoshop	0.19	0.81
ptlens	0.49	0.51
Agfa_DC-504	0.43	0.57
Agfa_DC-733s	0.7	0.3
Agfa_DC-830i	0.58	0.42
Agfa_Sensor505-x	0.45	0.55
Agfa_Sensor530s	0.43	0.57
Canon_Ixus55	0.5	0.5
Canon_Ixus70	0.65	0.35
Canon_PowerShotA640	0.53	0.47
Casio_EX-Z150	0.57	0.43
FujiFilm_FinePixJ50	0.72	0.28
Kodak_M1063	0.42	0.58
Nikon_CoolPixS710	0.52	0.48
Nikon_D200	0.33	0.67
Nikon_D70	0.38	0.62
Nikon_D70s	0.45	0.55
Olympus_mju_1050SW	0.73	0.27
Panasonic_DMC-FZ50	0.75	0.25
Pentax_OptioA40	0.53	0.47
Pentax_OptioW60	0.52	0.48
Praktica_DC25.9	0.62	0.38
Ricoh_GX100	0.58	0.42
Rollei_RCP-7325XS	0.63	0.37
Samsung_L74wide	0.58	0.42
Samsung_NV15	0.77	0.23
Sony_DSC-H50	0.67	0.33
Sony_DSC-T77	0.62	0.38
Sony_DSC-W170	0.73	0.27

(threshold: 0.41)

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

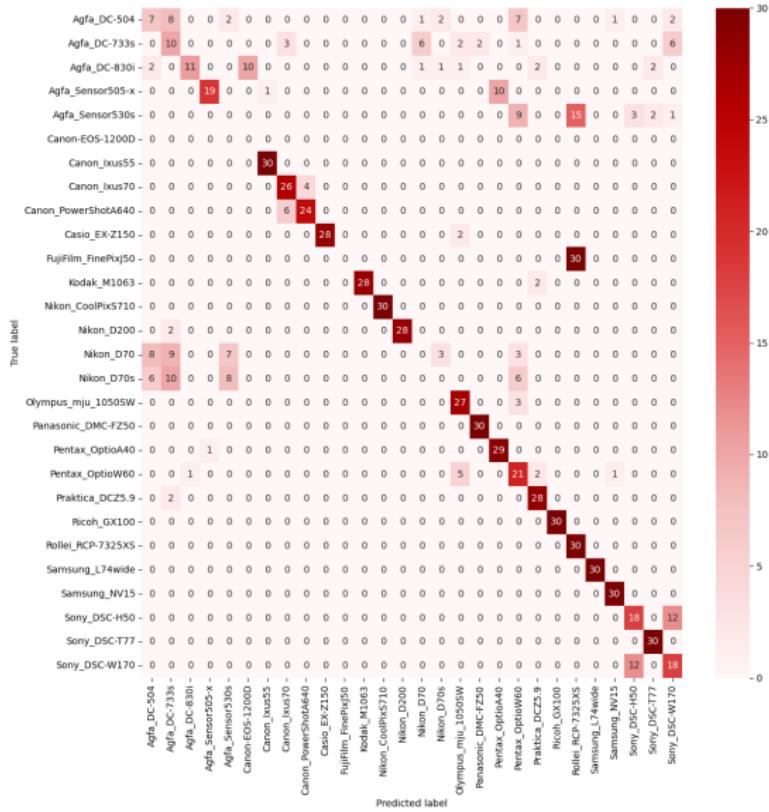
Multiclass classification

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

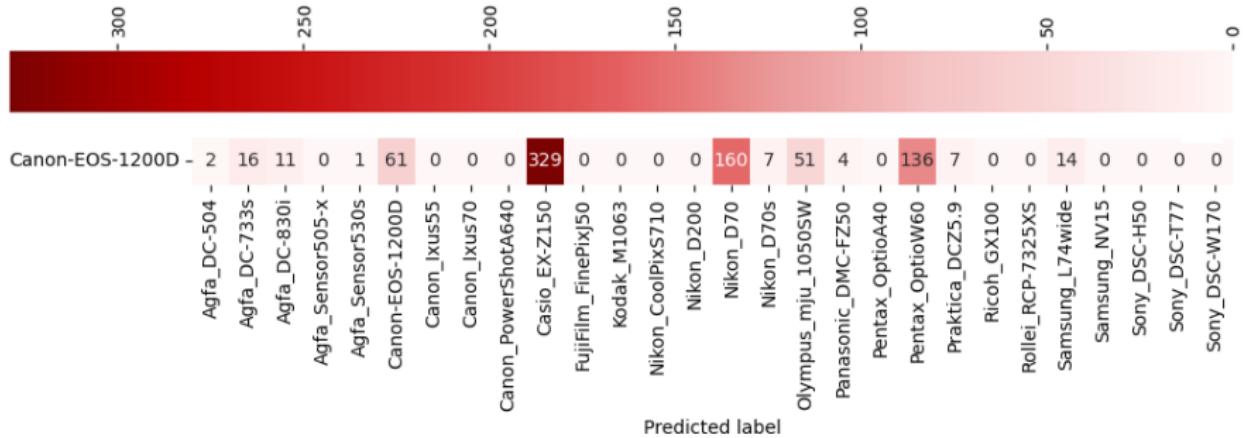
Being trained on the Dresden database, Noiseprint performs accurately for some models and terribly for others, coherently with the paper results.

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
```