# Watermarking Deep Neural Networks in Image Processing

Yuhui Quan [1] Huan Teng [1] Yixin Chen [1] Hui Ji [1]

Scaramuzzino Giovanna, Greavu Fabian

[1] authors of the paper

July 9, 2021

# Overview

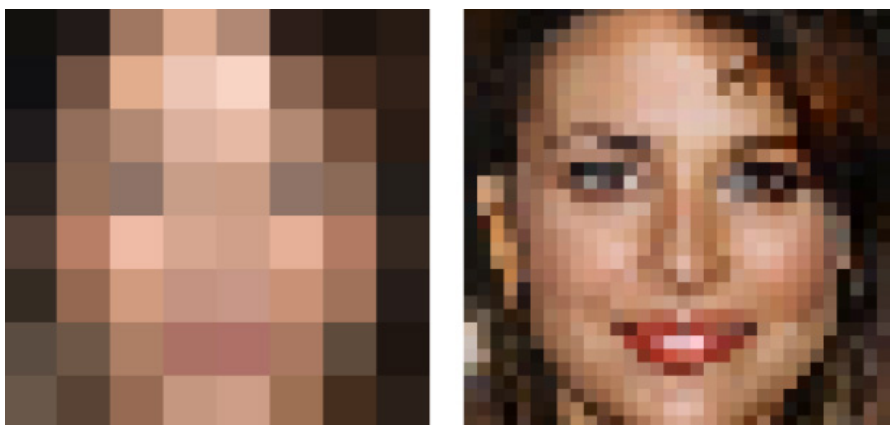
# Watermarking

- Machine Learning has gone beyond expectations over last years and due to significant model sizes and large amount of variables, lots of models are trained using hundreds of **GPUs** running for a few **weeks** on million-images datasets.
- Many companies/istitutes publish their pretrained models with **charges** for commercial uses.
- Being undoubtedly the owners' intelligent proprieties **IPs**, they need to be protected against **copyright infringements** or breaking of license agreements.
- One solution is **DNN watermarking**: concealing watermark information in a published model for ownership identification and copyright protection.

# Goal

- The authors studied the basic rules and principles for **watermarking DNNs for image processing tasks that map images to images**
- A **black-box approach** was developed for this problem
- *Image denoising* and *superresolution (SR)* were used as case studies


(a) Image denoising



(b) Superresolution

# White-Box and Black-Box methods

- **White-Box** methods: the watermark data are directly embedded into the weights of the model.

- **Black-Box** methods: encoding the watermark by specific model inputs (called *trigger keys*) and the expected model outputs (called *verification keys*). The embedding is done by training the model to satisfy such an expectation, and the verification is to check whether the expected input–output relationship exists.

# Watermarking background

- There are other DNN watermarking **methods** developed for classification tasks. But they map images to labels.
- It was not possible to use the existing methods because there are several fundamental differences between the two kinds of DNNs:
  1. DNNs for classification output a **label**. In contrast, DNNs for image processing output an **image**.
  2. The classification is about finding the **decision boundaries** among different classes while image processing being about finding **manifold** where desired images stays on. For this reason, the idea of adversarial examples used for watermarking classification DNNs it is **more** difficult to transfer to watermaking DNNs for image processing
  3. DNNs for image processing are not as deep as classification/detection ones. This results in **less redundancy** (overparametrizatrion) and yields a more **challenging** watermarking solution.

# Initial definitions

Before explaining the proposed method, some **definitions** are needed for understanding following slides:

- $\mathrm{M}(\cdot;\theta)$: DNN model of image processing parameterized by $\theta$
- $\mathbb{M}$: space of all DNNs solving the same task $\mathrm{M}$
- $\mathbb{X}$: set of images
- $\theta_{\mathrm{o}}$: parameters of $\mathrm{M}$ trained on $\mathbb{X}$
- $\theta^*$: paremeters of $\mathrm{M}$ after watermarking
- $\mathbb{K}$: space of all trigger keys
- $\mu(\cdot)$: a visual quality measure for images (es: PSNR)

# Principles for watermarking DNNs

There are 3 principles for DNN watermarking about image processing that need to be kept:

1. **Fidelity**: don't degrade the processing performance of the host model
$$\mu(\mathrm{M}(\mathbf{X};\boldsymbol{\theta}^*)) \approx \mu(\mathrm{M}(\mathbf{X};\boldsymbol{\theta}_\mathrm{o})), \text{ s.t. } \forall \mathbf{X} \in \mathbb{X}$$
2. **Uniqueness**: any DNN model for the same task cannot map the *trigger image* to the known *verification image* without related knowledge
$$\forall \mathbf{K} \in \mathbb{K},\ \mathrm{A}(\mathbf{K};\boldsymbol{\psi}) = \mathrm{M}(\mathbf{K};\boldsymbol{\theta}^*) \text{ iff } \mathrm{A} = \mathrm{M}, \boldsymbol{\psi} = \boldsymbol{\theta}^*$$
where $\mathrm{A} \in \mathbb{M}$ and $\boldsymbol{\psi}$ encodes the parameters that have not been tuned on $(\mathbf{K}, \mathrm{M}(\mathbf{K}, \boldsymbol{\psi}))$
3. **Robustness**: preserve the watermark information under attacks (i.e., small perturbation $\boldsymbol{\epsilon}$ on $\boldsymbol{\theta}^*$)
$$\mathrm{M}(\mathrm{K};\boldsymbol{\theta}^* + \boldsymbol{\epsilon}) \approx \mathrm{M}(\mathbf{K};\boldsymbol{\theta}^*)$$

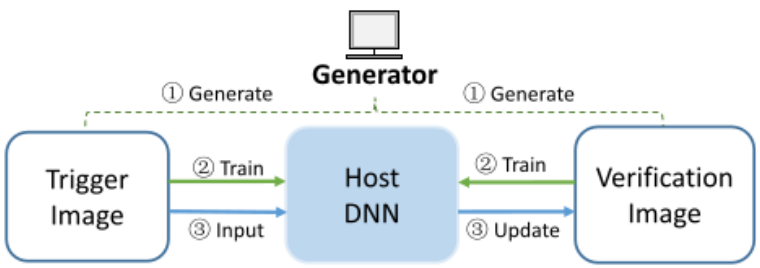
# Main idea

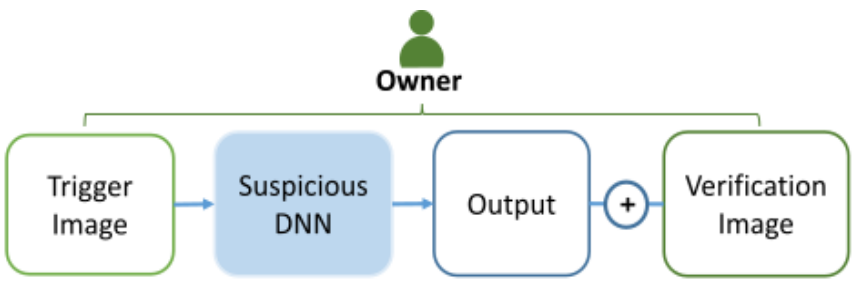
- The desidered output images are on **manifold**. DNN for image processing can cover only partial view of such a manifold, because is difficult to have training samples that sufficiently cover all important characteristics of all images.
- We will denote following spaces:
  - $\mathbf{B}$: the portion of manifold covered by the training samples
  - $\mathbf{D}$: space of all possible trigger images
- The **basic idea** for watermarking was to fine-tune the DNN model to ensure that, giving as input *trigger images* (from the domain $\mathrm{D}$), the output images approximate *verification images*
- There are two problems:
  - *define trigger images*: choose $\mathbf{D}$ **distant to** $\mathbf{B}$, using a random process to generate them
  - *define verification images*: appplying a simple non learning image processing method on the trigger images to obtain verification images

# Proposed Method

The proposed black-box method for watermarking image processing DNNs, is made up of 3 modules:

1. **watermark generation**: generate a *trigger image* and initial *verification image*. These are known by the owner
2. **watermark embedding**: trains the host DNN model to carry the watermark information
3. **watermark verification**: checks the existence of watermark on suspicious model



# 1. Watermark generation

- Let $\mathrm{U}(\mathrm{a}, \mathrm{b})$ the uniform distribution on the interval $[\mathrm{a},\mathrm{b}]$
- The *trigger key* image $\mathbf{K} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ was sampled as the uniform distribution: $\mathbf{K}(\mathrm{i}, \mathrm{j}) \sim \mathrm{U}(0, 1)$
- The *verification key* image $\mathbf{S} = \mathrm{G}(\mathbf{K}) \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ was generated by the operation $\mathrm{G}(\cdot)$
  - Deinoising: $\mathrm{G}(\mathbf{K}) = \mathbf{K} - \nabla\mathbf{K}$, where $\nabla$ denotes the gradient operator
  - Superresolution: $\mathrm{G}(\mathbf{K}) = \hat{\mathbf{K}} + \nabla\hat{\mathbf{K}}$, where $\hat{\mathbf{K}}$ is the upsampled version on $\mathbf{K}$ by linear interpolation

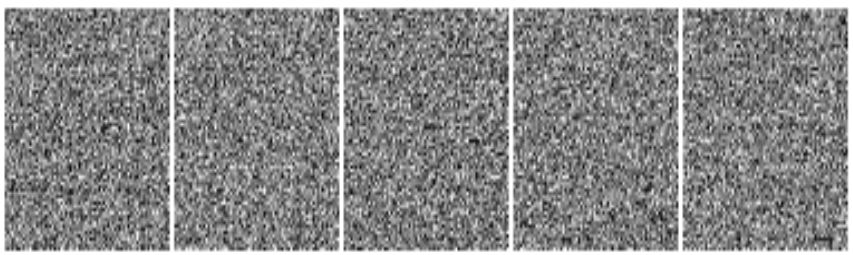


Figure: Examples of the trigger images

# 2. Watermark embedding

- The embedding of watermark was done by solving $\boldsymbol{\theta}^* = \arg\min_\theta \ell(\boldsymbol{\theta})$

## Loss function

$$\ell(\boldsymbol{\theta}) = \ell_{\mathrm{d}}(\boldsymbol{\theta}) + \lambda \ell_{\mathrm{w}}(\boldsymbol{\theta})$$

with:

- $\ell_{\mathrm{d}}(\boldsymbol{\theta})$: loss regarding the original denoising/SR performance

  $$\ell_{\mathrm{d}}(\boldsymbol{\theta}) = \frac{1}{2\mathrm{K}} \sum_{i=1}^{\mathbf{K}} \|(\mathrm{M}(\mathbf{X}_{\mathrm{i}}; \boldsymbol{\theta}) - \mathbf{Y}_{\mathrm{i}})\|_2^2$$

  $\mathbf{X}_{\mathrm{i}}$: i-th input image in the original training set; $\mathbf{Y}_{\mathrm{i}}$: the ground truth of $\mathbf{X}_{\mathrm{i}}$
- $\ell_{\mathrm{w}}(\boldsymbol{\theta})$: loss regarding watermark embedding

  $$\ell_{\mathrm{w}}(\boldsymbol{\theta}) = \|(\mathrm{M}(\mathbf{K}; \boldsymbol{\theta}) - \mathbf{S})\|_2^2$$
- $\lambda$: strength of embedding

- The training was stopped until $\ell_{\mathrm{w}}(\boldsymbol{\theta})$ is sufficiently small
- The verification key $\mathbf{S}$ was updated by $\mathrm{M}(\mathbf{K}; \boldsymbol{\theta}^*)$ after training the DNN

# 3. Watermark verification

Using *trigger image* $\mathbf{K} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ and *verification image* $\mathbf{S} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ ownership of a model $\mathrm{A}(\cdot)$ can be demonstrated

- Insert $\mathbf{K}$ into $\mathrm{A}(\cdot)$
- An image $\mathbf{S'} = \mathrm{A}(\mathbf{K})$ is obtaided
- The ownership is identified if the distance between $\mathbf{S'}$ and $\mathbf{S}$ is lower than some predefine threshold $\eta$. $\mathbf{S}$,$\mathbf{S'}$ have to normalized.

$$\mathrm{d}(\mathbf{S}, \mathbf{S'}) = \frac{1}{MN}\|\mathbf{S'} - \mathbf{S}\|_2 \leq \eta \quad , \qquad \mathrm{d}(\mathbf{S}, \mathbf{S'}) \in [0, 1] \tag{1}$$

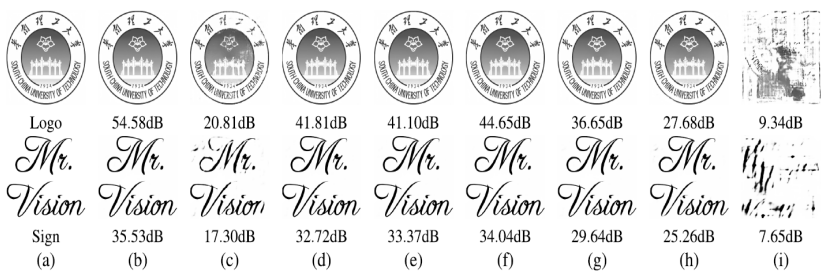
- $\eta = 6,07 * 10^{-3}$ was determinated by probabilistic approach.

# Auxiliary Copyright Visualizer

- An auxiliary module for **visualizing** the watermark information was proposed, because the verification image has no visual implication due to the need of randomness
- A generative DNN as a plug-and-play module $\mathbf{R}(\cdot;\phi) : \mathrm{R}^{\mathrm{M\times N}} \rightarrow \mathrm{R}^{\mathrm{P\times Q}}$ with parameter vector $\phi$ was proposed
- This module was trained to map the verification key $\mathbf{S}$ to a recognizable copyright image $\mathbf{I}$ by minimizing:

$$\mathbf{r}(\phi) = \|\mathbf{R}(\mathbf{S};\phi) - \mathbf{I}\|_2^2 \tag{2}$$

The module $\mathrm{R}$ can be activated by corresponding verification key, can be viewed as a memory of the copyright



# Training Configurations

- As for **Host models**, DnCNN [4] and RED [2] were used as image denoising DNNs and VDSR was used as image SR DNN (also more tests were done with DnCNN for SR).
  All experiments were implemented in Tensorflow.
- The **Watermarking** process (in embedding step) was done on trigger images with size **40x40** for denoising and **20x20** for SR.
  The model is trained with ADAM. Learning rate $10^{-3}$, epochs=8.
- The **Auxiliary Visualizer** is trained with ADAM having $lr = 10^{-3}$ and all default params. The copyright images are a logo and a sign image.

---

[4] Zhang et al. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising". 2017

[2] Mao, Shen, and Yang. *Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections*. 2016

# Analysis Methods for comparison

- The existing methods are applied to classification and can't be directly applied to our situation. Following two baselines were made in order to compare them to proposed method.
  - **Base-1** thinks of the output image of the DNN as a label of labels and then apply [1] for embedding and verification.
  - **Base-2** adds a random projection and softmax at the end of host model acting as a classifier, then applying [3] for watermarking. Projection matrix is sis randomly generated and not learned (avoid overfitting to watermarking process).
- Both baselines have class number set to 256. Using more than 40 images in Base-2 will cause large performance decrease in host model.
- Analyzing **Fidelity**, **Uniqueness**, **Robustness**

---

[1] Adi et al. "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring". 2018

[3] Rouhani, Chen, and Koushanfar. *DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models*. 2018

# Fidelity Analysis

- Fidelity Analysis is done by comparing original and watermarked model's performances.
- Tests are conduced on Img12, Img14 and BSD68, PolyU (with real noise removed) datasets.
- $\lambda = 0$ refers to unwatermarked model. Results on $1, 10^{-1}, 10^{-2}, 10^{-3}$ suggests that a PSNR gap $<= 0.05$dB is acceptable (imperceptible for humans).
- Degradation of denoising is acceptable when:
  - $\lambda <= 10^{-2}$ for DnCNN
  - $\lambda <= 10^{-3}$ on RED (more sensitive than DnCNN)
- Regarding SR similar results were concluded ($\lambda <= 10^{-3}$).
- Fidelity on proposed method is better than 2 baselines:
  - Base-1 shows poor fidelity with more than 0.22dB PSNR loss on DnCNN
  - Base-2 is worse

# Uniqueness Analysis

Only evaluating uniqueness of proposed method (two baselines inherite their ones from original version of classification).
In order to do this analysis, the tests are done on:

- Unwatermarked models
- Watermarked models
- Pairs of Watermarked models

# Uniqueness Analysis

- **Unwatermarked models**: in order not to encode any wm info, When creating trigger key and verification image pairs, it is desired that a randomly generated trigger key does not produce a verification key like one of the pairs calculated.
  When generating 200, 400, ... , 1000 trigger/verification image pairs and calculating distance $s$ btw verification images from pair and new ones, minimal distances are:

| **Task** | **Model** | $P$=200 | $P$=400 | $P$=600 | $P$=800 | $P$=1000 |
|---|---|---|---|---|---|---|
| Denoising | DnCNN | 1.11 | 1.11 | 1.11 | 1.11 | 1.11 |
| | RED | 1.17 | 1.17 | 1.17 | 1.17 | 1.17 |
| SR | DnCNN | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| | VDSR | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 |

- Such values are definitevly larger than $\eta = 6.07 * 10^{-3}$ (verification threshold).

# Uniqueness Analysis

- **Watermarked models**: the watermarked model does not have to encode different info with hasn't been encoded for.
  When having a model and its own pair of trigger/verification images, when comparing to multiple different trigger/verification keys from other models, the verification keys have to be very different (the distance $d$ ahs to be $> n$).
  As before, generating 200, 400, ... , 1000 trigger/verification image pairs, it is seen that increasing the pair numbers, minimal distace is more stable. Check out minimal distances with 1000 pairs:

| Task | Model | $P$=200 | $P$=400 | $P$=600 | $P$=800 | $P$=1000 |
|---|---|---|---|---|---|---|
| Denoising | DnCNN | 7.02 | 7.02 | 7.02 | 7.02 | 7.02 |
| | RED | 7.28 | 7.28 | 7.28 | 7.24 | 7.23 |
| SR | DnCNN | 7.42 | 7.38 | 7.38 | 7.38 | 7.38 |
| | VDSR | 6.97 | 6.97 | 6.97 | 6.97 | 6.97 |

- Such values are larger than $\eta = 6.07 * 10^{-3}$ (verification threshold).

# Uniqueness Analysis

- **Pairs of Watermarked models**: what about unboundnesses of DNNs leading to watermarking?. Two similar verification keys from different models could arise? (models having similar functions but different verification keys). This is not the case because initial verification key is very close to model's output based on learned parameters after embedding. Together with that the first and second pair of trigger keys are very different, so uniqueness is preserved.
- In order to do that a pair of trigger/verification images were created for the DnCNN and RED models and watermarking was embedded.Then the trigger keys are switched between the 2 models. Repeating 30 times, the resulting distance metric (averaged on 30 runs) resulted in:
  - $7,03 * 10^{-3}$ for DnCNN
  - $7,62 * 10^{-3}$ for RED
- Such values are larger than $\eta = 6.07 * 10^{-3}$ (verification threshold) demostrating that in proposed method watermark operation can well distinguish watermarked models (and so identify owners).
  Similar results were seen for SR-DnCNN and VSDR.

# Robustness Analysis

- The robustness of the proposed method was evaluated on three types of attacks: model **compression**, model **fine-tuning**, and **watermark overwriting**
- The results of the proposed method were reported from different aspects:
  1. **Denoising/SR Performance of Host Model in Terms of the Average PSNR on Img12/Img14**: this metric reflect the performance decrease of host model caused by the attack. An attack is meaningless if it causes too large performance decrease, as a model with low performance is useless in practice.
  2. **Watermark Distance**: distance $d$ calculated for watermark vertification. This determines whether the proposed methos can succeed ( this metric wasn't reported for the baselines).
  3. **Whether the Verification Succeeded**: "succeeded" or "failed" for the verification by the proposed method were reported

# Robustness Analysis

- In orderto compair the methods more than success/fail, we will consider following criteria:
  - **Normalized Correlation Coefficient (NCC) Between the Verification Key and the Triggered Output of the Verified Model**: Large NCC means that the triggered output of the watermarked model is similar to the verification key even under attacks. Thus, higher NCC implies stronger robustness.
  - **NCC-Based Verification**: The verification passes if the NCC is larger than a threshold that it was setted to 0.95 in the experiments.

# Robustness Analysis: Model compression

- The compression of the watermarked models was done by parameter pruning on each layer: sorting the model weights in the ascending order in terms of magnitude and then setting the top $p\%$ to zeros.

| Model | Pruning Rate | Our Result | | | Comparison to Baselines | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | Distance ($\times 10^{-3}$) | Verification $d > \eta$ | NCC | | | Verification (NCC>0.95) | | |
| | | | | | Base-1 | Base-2 | Ours | Base-1 | Base-2 | Ours |
| **DnCNN** | 0.25 | 29.77 | 2.09 | **Succeeded** | 0.9723 | 0.7501 | 0.9915 | **Succeeded** | Failed | **Succeeded** |
| | 0.22 | 29.42 | 2.82 | **Succeeded** | 0.9623 | 0.7390 | 0.9857 | **Succeeded** | Failed | **Succeeded** |
| | 0.35 | 29.22 | 4.48 | **Succeeded** | 0.9350 | 0.6385 | 0.9707 | Failed | Failed | **Succeeded** |
| | 0.40 | 28.39 | 6.36 | Failed | 0.9070 | 0.5490 | 0.9355 | Failed | Failed | Failed |
| **RED** | 0.08 | 26.26 | 2.41 | **Succeeded** | 0.9790 | 0.6926 | 0.9889 | **Succeeded** | Failed | **Succeeded** |
| | 0.10 | 23.85 | 3.83 | **Succeeded** | 0.9346 | 0.6549 | 0.9723 | Failed | Failed | **Succeeded** |
| | 0.15 | 19.49 | 5.98 | **Succeeded** | 0.8627 | 0.5485 | 0.9500 | Failed | Failed | Failed |
| | 0.20 | 23.18 | 7.93 | Failed | 0.8114 | 0.5481 | 0.8654 | Failed | Failed | Failed |
| **VDSR** | 0.25 | 30.04 | 2.21 | **Succeeded** | 0.9333 | 0.6944 | 0.9960 | Failed | Failed | **Succeeded** |
| | 0.22 | 29.89 | 3.75 | **Succeeded** | 0.8834 | 0.5477 | 0.9880 | Failed | Failed | **Succeeded** |
| | 0.35 | 29.48 | 5.01 | **Succeeded** | 0.8423 | 0.5258 | 0.9765 | Failed | Failed | **Succeeded** |
| | 0.40 | 28.94 | 6.58 | Failed | 0.8034 | 0.5250 | 0.9559 | Failed | Failed | **Succeeded** |
| **SR-DnCNN** | 0.25 | 29.88 | 2.76 | **Succeeded** | 0.9837 | 0.6826 | 0.9920 | **Succeeded** | Failed | **Succeeded** |
| | 0.22 | 29.78 | 3.86 | **Succeeded** | 0.9451 | 0.6418 | 0.9846 | Failed | Failed | **Succeeded** |
| | 0.35 | 29.47 | 5.99 | **Succeeded** | 0.8938 | 0.5476 | 0.9609 | Failed | Failed | **Succeeded** |
| | 0.40 | 29.07 | 6.73 | Failed | 0.8307 | 0.5258 | 0.9494 | Failed | Failed | Failed |

# Robustness Analysis: Model fine-tuning

- Two types of model fine-tuning were used:
  1. Use the **original** training data
  2. Use **new** data set: **KTH-TIPS** (for the denoising DnCNNs) and **DIV2K** (for the SR DNNs)
- The proposed method is robust to the fine-tuning attacks
- With the increase of epochs, the watermark becomes weaker but is still successfully detected, even with100-epoch fine-tuning
- For DnCNN and RED the fine-tuning on **new data** has a larger impact on the watermark than that on the original data, probabily because texture images have higher randomness than the natural images of the original data set, and therefore, the texture patches are closer to the randomly generated patches in the trigger image
- Base-2 is very sensitive to fine-tuning attacks again performed much worse than Base-1, for this reason methods for watermarking classification DNNs cannot be directly applied to image processing DNNs.

# Robustness Analysis: Model fine-tuning

| Model | Data | #Epochs | Our Result | | | Comparison to Baselines | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | PSNR | Distance ($\times 10^{-3}$) | Verification $d > \eta$ | NCC | | | Verification (NCC>0.95) | | |
| | | | | | | Base-1 | Base-2 | Ours | Base-1 | Base-2 | Ours |
| DnCNN | Original | 10 | 30.42 | 1.64 | Succeeded | 0.9988 | 0.5490 | 0.9953 | Succeeded | Failed | Succeeded |
| | | 25 | 30.41 | 1.66 | Succeeded | 0.9943 | 0.5478 | 0.9956 | Succeeded | Failed | Succeeded |
| | | 50 | 30.43 | 1.81 | Succeeded | 0.9949 | 0.5477 | 0.9959 | Succeeded | Failed | Succeeded |
| | | 75 | 30.44 | 2.38 | Succeeded | 0.9949 | 0.5268 | 0.9952 | Succeeded | Failed | Succeeded |
| | | 100 | 30.41 | 3.12. | Succeeded | 0.9955 | 0.5267 | 0.9938 | Succeeded | Failed | Succeeded |
| | Texture | 10 | 30.28 | 1.23 | Succeeded | 1.0000 | 0.6826 | 0.9979 | Succeeded | Failed | Succeeded |
| | | 25 | 30.15 | 1.76 | Succeeded | 0.9877 | 0.5477 | 0.9944 | Succeeded | Failed | Succeeded |
| | | 50 | 30.09 | 2.32 | Succeeded | 0.9801 | 0.5255 | 0.9914 | Succeeded | Failed | Succeeded |
| | | 75 | 30.11 | 3.81 | Succeeded | 0.9786 | 0.5250 | 0.9832 | Succeeded | Failed | Succeeded |
| | | 100 | 30.10 | 5.69 | Succeeded | 0.9574 | 0.5245 | 0.9580 | Succeeded | Failed | Succeeded |
| RED | Original | 10 | 30.28 | 2.08 | Succeeded | 1.0000 | 0.6111 | 0.9957 | Succeeded | Failed | Succeeded |
| | | 25 | 30.28 | 2.09 | Succeeded | 1.0000 | 0.5560 | 0.9932 | Succeeded | Failed | Succeeded |
| | | 50 | 30.30 | 2.08 | Succeeded | 0.9997 | 0.5490 | 0.9935 | Succeeded | Failed | Succeeded |
| | | 75 | 30.23 | 4.03 | Succeeded | 0.9997 | 0.5268 | 0.9915 | Succeeded | Failed | Succeeded |
| | | 100 | 30.26 | 5.62 | Succeeded | 0.9988 | 0.5253 | 0.9836 | Succeeded | Failed | Succeeded |
| | Texture | 10 | 29.90 | 1.39 | Succeeded | 0.9994 | 0.5773 | 0.9969 | Succeeded | Failed | Succeeded |
| | | 25 | 30.01 | 1.82 | Succeeded | 0.9860 | 0.5560 | 0.9936 | Succeeded | Failed | Succeeded |
| | | 50 | 30.09 | 2.00 | Succeeded | 0.9824 | 0.5549 | 0.9927 | Succeeded | Failed | Succeeded |
| | | 75 | 30.05 | 3.82 | Succeeded | 0.9791 | 0.5258 | 0.9877 | Succeeded | Failed | Succeeded |
| | | 100 | 30.02 | 5.83 | Succeeded | 0.9767 | 0.5244 | 0.9655 | Succeeded | Failed | Succeeded |
| VDSR | Original | 10 | 30.14 | 0.84 | Succeeded | 0.8630 | 0.8501 | 0.9993 | Failed | Failed | Succeeded |
| | | 25 | 30.10 | 1.06 | Succeeded | 0.7756 | 0.8056 | 0.9989 | Failed | Failed | Succeeded |
| | | 50 | 30.05 | 1.43 | Succeeded | 0.7594 | 0.5258 | 0.9979 | Failed | Failed | Succeeded |
| | | 75 | 30.00 | 1.88 | Succeeded | 0.7609 | 0.5255 | 0.9966 | Failed | Failed | Succeeded |
| | | 100 | 29.96 | 2.29 | Succeeded | 0.7606 | 0.5245 | 0.9948 | Failed | Failed | Succeeded |
| | DIV2K | 10 | 30.12 | 2.40 | Succeeded | 0.8152 | 0.8504 | 0.9947 | Failed | Failed | Succeeded |
| | | 25 | 30.13 | 3.04 | Succeeded | 0.7631 | 0.6926 | 0.9949 | Failed | Failed | Succeeded |
| | | 50 | 30.06 | 3.86 | Succeeded | 0.7599 | 0.6478 | 0.9889 | Failed | Failed | Succeeded |
| | | 75 | 30.07 | 4.36 | Succeeded | 0.7612 | 0.6268 | 0.9854 | Failed | Failed | Succeeded |
| | | 100 | 30.05 | 4.83 | Succeeded | 0.7598 | 0.6246 | 0.9815 | Failed | Failed | Succeeded |
| SR-DnCNN | Original | 10 | 30.11 | 1.68 | Succeeded | 1.0000 | 0.5476 | 0.9979 | Succeeded | Failed | Succeeded |
| | | 25 | 30.09 | 2.18 | Succeeded | 0.9967 | 0.5257 | 0.9961 | Succeeded | Failed | Succeeded |
| | | 50 | 30.03 | 2.72 | Succeeded | 0.9545 | 0.5250 | 0.9930 | Succeeded | Failed | Succeeded |
| | | 75 | 30.00 | 3.06 | Succeeded | 0.9137 | 0.5248 | 0.9906 | Failed | Failed | Succeeded |
| | | 100 | 29.96 | 3.45 | Succeeded | 0.8843 | 0.5245 | 0.9878 | Failed | Failed | Succeeded |
| | DIV2K | 10 | 30.17 | 5.20 | Succeeded | 0.9979 | 0.5490 | 0.9708 | Succeeded | Failed | Succeeded |
| | | 25 | 30.14 | 4.97 | Succeeded | 0.9611 | 0.5478 | 0.9738 | Succeeded | Failed | Succeeded |
| | | 50 | 30.08 | 4.82 | Succeeded | 0.9038 | 0.5267 | 0.9752 | Failed | Failed | Succeeded |
| | | 75 | 30.05 | 4.93 | Succeeded | 0.8842 | 0.5253 | 0.9741 | Failed | Failed | Succeeded |
| | | 100 | 30.00 | 4.97 | Succeeded | 0.8682 | 0.5244 | 0.9734 | Failed | Failed | Succeeded |

Figure: Results for Model fine-tuning

# Robustness Analysis: Watermark Overwrite

- Overwriting was done by **embedding a new trigger key** and its corresponding verification key into the watermarked model
- With **multiple new watermarks** written into the host model, the **original one can still be detected**. new trigger images are very likely to be far away from the original one due to the randomness in the generation
- The proposed method **succeded in all tested** models while both the baselines fail in all cases.

| Model | Our Result | | | Comparison to Baselines | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR | Distance ($\times 10^{-3}$) | Verification $d > \eta$ | NCC | | | Verification (NCC>0.95) | | |
| | | | | Base-1 | Base-2 | Ours | Base-1 | Base-2 | Ours |
| **DnCNN** | 30.41 | 3.66 | **Succeeded** | 0.8128 | 0.7957 | 0.9751 | Failed | Failed | **Succeeded** |
| **RED** | 30.27 | 3.47 | **Succeeded** | 0.8915 | 0.8689 | 0.9779 | Failed | Failed | **Succeeded** |
| **VDSR** | 30.10 | 3.07 | **Succeeded** | 0.7628 | 0.6581 | 0.9915 | Failed | Failed | **Succeeded** |
| **SR-DnCNN** | 30.13 | 3.19 | **Succeeded** | 0.7875 | 0.8833 | 0.9891 | Failed | Failed | **Succeeded** |

Figure: Results for Watermark overwrite

# Capacity

- In order to evaluate the **capacity**, trigger keys with different sizes are generated and embedded in host model (20x20, 40x40, 80x80, 160x160, 320x320, 480x480, 600x600).
- Denoising/SR performance is evaluated on test dataset
- Results shows that **larger trigger image size means less performance**.
- DnCNN (denoising) performance is not sensitive within 80x80, RED (SR) is within 80x80 (because RED is less redundant than DnCNN)
- For size 600x600, performance drops about 0.27dB
- Previous experiments had shown good detection results with 40x40
- It is hard to compare capacity between baselines and proposed method, but Base-1 was tested and DnCNN performance over size 600x600 was still much better than Base-1.

# Conclusions

- The proposed method was a black-box approach on watermarking a DNN model used for image processing
- Its effectiveness was demonstrated in the context of image denoising and SR
- Watermark embedding was done by modifying host DNN so as to degrade its performance on a specific image that has a statistically significant difference from the training data
- The results show that the proposed method meets the need for fidelity, uniqueness, and capacity, and it is robust to the attacks of model compression, model fine-tuning, and watermark over-writing. It is noted that the proposed watermarking technique can be easily generalized to the DNNs for other image processing tasks.

# References


Yossi Adi et al. "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring". In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1615–1631. ISBN: 978-1-939133-04-5. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/adi.

Xiao-Jiao Mao, Chunhua Shen, and Yu-Bin Yang. *Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections*. 2016.

Bita Darvish Rouhani, Huili Chen, and Farinaz Koushanfar. *DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models*. 2018. arXiv: 1804.00750 [cs.CR].

K. Zhang et al. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.