Adversarial Attacks on Deep Learning-Based Splice Localization EXIF Self-Consistency Model

Ang Yi Zhe

May 31, 2021

Contents

1 Introduction

This project explores the influence of adversarial attacks on splice-detection/localization algorithms, more specifically on the EXIF Self-Consistency (EXIF-SC) model by Huh et al.[5]

For resources on learning more about media forensics and counter-forensics, I have found the following survey papers to be useful: [13][11].

Here is a summary of the realized deliverables of this project:

1. Re-implemented the **EXIF-SC model in the PyTorch Framework** for ease of any subsequent experimentation, and evaluated it on various datasets to ensure the reproducibility of the reported results as much as possible.

2. Implemented the **training algorithm of the EXIF-SC model**. However, the correctness of the implementation has not been properly validated.

3. Implemented **adversarial attack algorithms on the EXIF-SC model** and demonstrated their effectiveness in reducing the performance of the model's predictions.

The codebase containing all the above-mentioned implementations can be found here.

1.1 Datasets

All the datasets utilized in the experiments are implemented as PyTorch Dataset classes in this folder. Initializing the dataset automatically downloads and preprocesses the data. This folder compiles the metadata of the datasets.

More recent and larger-scale datasets that can serve as robust evaluation datasets but not utilized in this project include:

- DEFACTO Dataset[9], which collects over 200,000 images with realistic manipulations, including splicings, copy-moves, removals and face-morphing.

- Wild Web Tampered Image Dataset[14], contains 80 cases of real forgeries, each with multiple versions of the same image, totalling up to 13,577 unique images labelled with ground truth binary masks.

2 Implementing EXIF-SC

The authors of EXIF-SC have released an official implementation of their model, coded up using TensorFlow 1.X. The released code only contains the "inference" portion of their model and the final trained weights, lacking:

- The evaluation pipeline and implementations of metrics used to arrive at the results reported in the paper.
- The training algorithm.
- The training dataset used to attain the final weights.

In short, the released code only exposes the "forward-pass" of the model, and only allows users to perform inference on their own test examples.

As accurately as possible with the available information at hand, I have attempted to port over the implementation to PyTorch.

Dataset	Splice Localization, AP ↑		Splice Detection, cIOU ↑	
	Original Paper	**Re-implementation**	**Original Paper**	**Re-implementation**
RT[8]	0.55	0.54	0.54	0.54
Columbia[10]	0.98	0.95	0.85	0.67
In-The-Wild[5]	NA	NA	0.58	0.64
Hays[4]	NA	NA	0.65	0.54

Table 1: Comparison between results from Original Paper and Re-implementation of EXIF-SC. The two main metrics reported are average precision (AP) for detection, and class-balanced IOU (cIOU) for localization.

2.1 Model Architecture

Since the model weights are already provided, and most of the post-processing code is in NumPy, the bulk of the work was dedicated to porting the network architecture building code from TensorFlow over to PyTorch.

Luckily, an open-source framework, MMdnn, already provides utilities for generating the PyTorch model building code for the "TensorFlow-slim ResNet50" network used in the official implementation.

Thereafter, I just had to

1. Examine the model weights in the TensorFlow checkpoint provided and make some modifications (for e.g. reshaping, removing unnecessary weights) in order to load them into PyTorch layers.
2. Save the weights into a NumPy format.
3. And finally make some minor modifications to the default model building code provided by MMdnn (as per the EXIF-SC network architecture) in order to load those weights.

A rough overview of the porting process can be found in this notebook, and the model code is located in this folder.

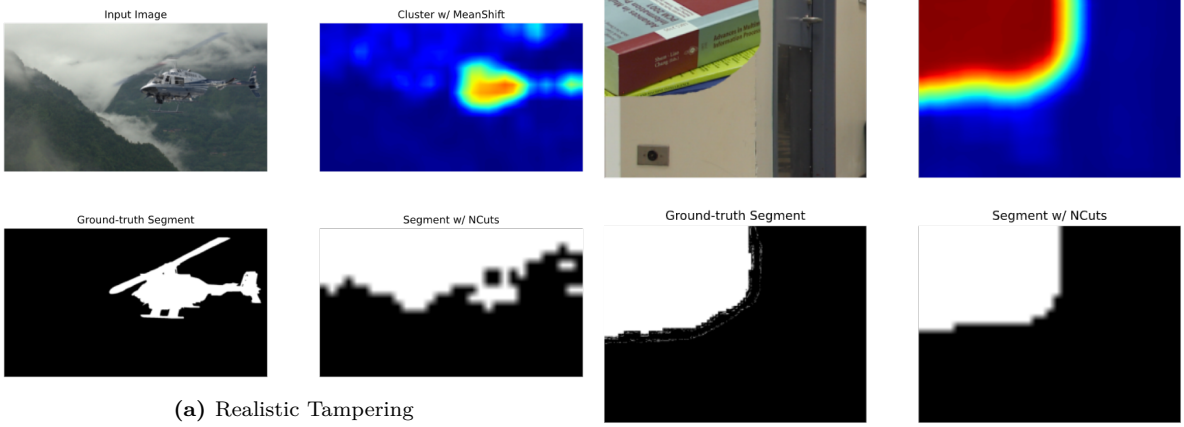
2.1.1 Evaluation

Evaluation of the re-implemented model was compared to the reported results in the paper, with a few caveats:

- There are likely to be some hidden hyperparameter choices in the evaluation pipeline that I have not replicated exactly.
- In the paper, every dataset in split in half into train / test sets. Since the exact split is unknown, all my experiments simply evaluate on the entire dataset.
- In order to compute class-balanced IOU (cIOU), I resize all the ground-truth and prediction maps into the same size in order to compute the optimal threshold and corresponding IOU score for each image in a vectorized and efficient manner.

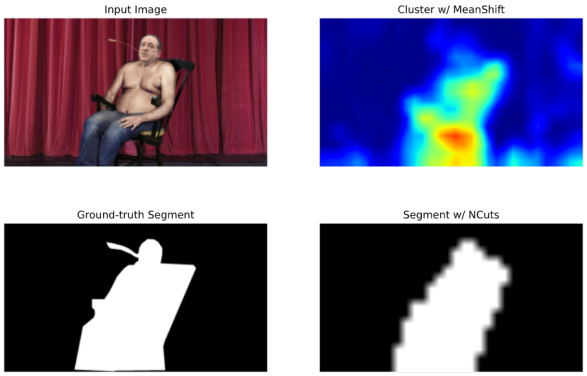
Table 1 summarizes the differences between the results from the paper and my re-implementation. Figure 1 shows some example predictions from the re-implemented model on the respective datasets.

While the results cannot be perfectly reproduced due to the caveats listed above, the obtained metrics do seem comparable, and the qualitative examples in Figure 1 also seem to show that the re-implemented model does work as intended.

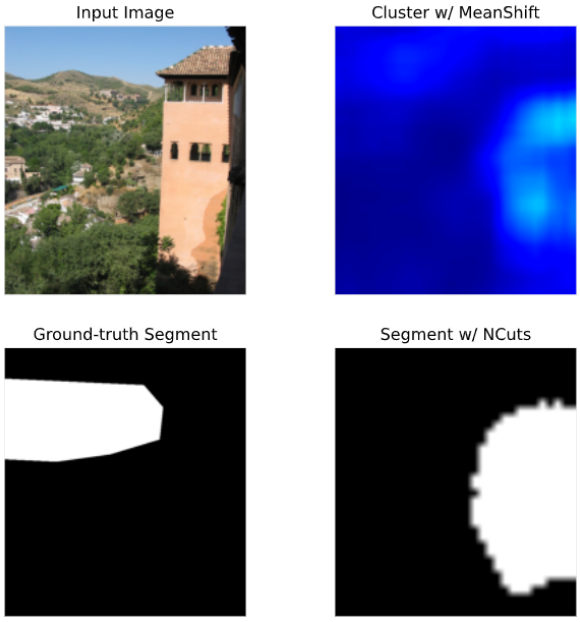


(a) Realistic Tampering

(b) Columbia



(c) In-The-Wild



(d) Hays

Figure 1: Example Predictions from Re-implemented EXIF-SC Model. An example is shown from each dataset.

2.2 Training Algorithm

In order to implement the training algorithm, I used the MIRFLICKR25000[6] dataset which contains images and their associated EXIF attribute information as a sample training dataset. The training code is located in this folder.

A sketch of the Siamese network training algorithm follows:

2.2.1 First Stage

The first stage trains the network to predict EXIF attribute consistency (multi-label classification) from a pair of image patches.

The sampling process for a training batch is as follows:

1. We'll select a specific EXIF attribute value. To do this, we'll first randomly sample an EXIF attribute, and then randomly sample a value from it.

2. The first half of the batch will be consistent, i.e. pairs will both have that specific attribute value. We randomly sample from the set of images with that attribute value.

3. The second half of the batch will be inconsistent, i.e. the first image will have that specific value, but the second image to be compared with will have a different value. We sample from the rest of the images to form those second images.

2.2.2 Second Stage

The second stage attaches a MLP on top of the EXIF attribute predictions and trains it to predict whether an image pair comes from the same image or not (binary classification).

The rest of the network weights are frozen, and only this MLP is trained.

A training batch is constructed by ensuring that the first half of the batch is consistent (pairs are patches that come from the same image), and the second half of the batch is inconsistent (pairs are patches that come from different images).

2.2.3 Implementation Notes

Overall, the original paper was not entirely clear about the exact technical details of the training algorithm, and hence some design choices were made simply based on what made sense:

- The EXIF attributes to predict are chosen dynamically based on the dataset, for e.g. the top 80 EXIF attributes with the least missing values.
- Binary cross-entropy is used as the loss function.
- If an attribute is missing for either image, it is immediately assigned a target of 0.
- The incorporation of post-processing consistency attributes is yet to be implemented.

I had only tested to make sure that the training algorithm is able to run without any bugs, and had not evaluated it on any testing data to validate the correctness of the implementation.

3 Adversarial Attacks on EXIF-SC

The adversarial attack algorithms are implemented in this folder.

3.1 White-box Attacks

Since the EXIF-SC model isn't actually an end-to-end deep learning algorithm (it has non-differentiable post-processing steps), we can't take gradients of the final output w.r.t. the input image and apply the typical white-box adversarial attacks directly. Nevertheless, we can still take gradients within the neural network portions of the model, and manipulate the network's output in order to influence how the downstream post-processing steps produces the final output to our liking.

This was what was explored in [12], where the authors performed adversarial attack experiments on various splice localization models that produces features from a neural network as a intermediary step (which includes the EXIF-SC model).

Since the EXIF-SC model produces its final localization prediction based on how "different" (with respect to the predicted EXIF attributes) the image's patches are, all we have to do is to make each image patch feature output by the neural network to be similar to one another.

In other words, we'll find a perturbation to the input image such that all image patch features output by the neural network is as close as possible to each other.

A sketch of the algorithm proposed by [12] is as follows:

1. Identify all the authentic patches, i.e. image patches that don't overlap with the spliced regions, and compute their mean feature vector. This will be our target feature for every image patch.
2. We'll solve an optimization procedure that finds an image perturbation that causes all patch features to be close to the target feature, using gradient descent:
   (a) Take the L2 loss between each patch feature and the target feature, and take the sum over all patches; this will be the loss function to optimize.
   (b) Take the gradient of the loss w.r.t. to the input image, and normalize the gradient by its infinity norm before applying the gradient update on the image.
   (c) Clip the pixels of the input image to be between $[0, 255]$.
3. Repeat the gradient descent procedure for a specified number of iterations.
4. Finally, choose the perturbed image that attains the lowest loss value, round its pixel values to be integers, and we attain our final perturbed image.

The above algorithm was implemented as a baseline, which I'll refer to as Adv-Mean.

One possible downside of this algorithm is that the localization maps that it produces may seem unnatural. As shown in the prediction examples in Figure 2, given a sufficiently large step-size or number of gradient descent iterations, it is able to force all patch features to be the same, resulting in a completely uniform localization map. However, the EXIF-SC model isn't perfect, and often produces chunks of "noise" in the localization map.

Hence, rather than fixing the same target feature for each image patch, we can attempt to imitate the feature distribution of the authentic patches instead. Namely, we can choose our target features as follows:

1. Identify all the authentic patches, and their corresponding feature vectors.
2. For each non-authentic patch, we'll set their target feature to be a random (uniform) sample from the set of authentic feature vectors.

This might result in more "natural-looking" localization maps that still conceal the actual spliced region. I'll refer to this variant as Adv-Sample.

	Columbia[10]				DSO-1[1]			
Metrics	**Clean**	**JPEG**	**Adv Mean**	**Adv Sample**	**Clean**	**JPEG**	**Adv Mean**	**Adv Sample**
F1 ↑	0.8703	0.6397	0.7014	**0.5067**	0.9473	**0.9253**	0.9263	0.9281
MCC ↑	0.6971	0.2417	**0.0004**	0.0081	0.3650	0.1209	**0.0221**	0.0541
mAP ↑	0.8958	0.6084	0.6984	**0.3832**	0.9652	0.9195	0.9313	**0.9129**
AUC ↑	0.9697	0.8476	0.8773	**0.7213**	0.8439	**0.6774**	0.7303	0.6877
cIOU ↑	0.8490	0.6528	0.7194	**0.5363**	0.5038	0.5124	0.5263	**0.5041**

Table 2: Evaluation Results of Adversarial Attacks on EXIF-SC Model. The images from the Columbia and DSO-1 datasets were modified for the attack. The lower the metrics are, the more potent the attack.

3.2 Black-box Attacks

A black-box attack is one that does not require access to the specific model in question or its parameters, outputs or gradients. A simple approach to disrupt useful forensic traces on images is lossy / JPEG compression. This will possibly work well on models such as EXIF-SC that rely on traces such as device fingerprints in order to make its prediction.

Other approaches that may be worth exploring include:

- Performing the attack on a surrogate detector, expecting that the attack will also work against the real detector. This was explored in [12] where they demonstrated that adversarial examples generated from similar detectors do still generalize well when applied onto the EXIF-SC model.
- Generative models like GANs have shown to be able to create images that can falsify camera model traces, or more generally imitate the distribution of a training dataset to counterfeit forensic traces. This can fool detectors that rely on such traces for its prediction [7][2][3].

3.3 Evaluation

The efficacy of Adv-Mean, Adv-Sample and JPEG Compression was experimented on the EXIF-SC model using an evaluation scheme similar to [12], with these differences:

- For the MCC metric that requires binary decision maps instead of score maps, a default threshold value of 0.5 was chosen (most papers search for the optimal threshold value and report the corresponding highest metric score instead).
- The F1 score was computed by finding the optimal threshold.

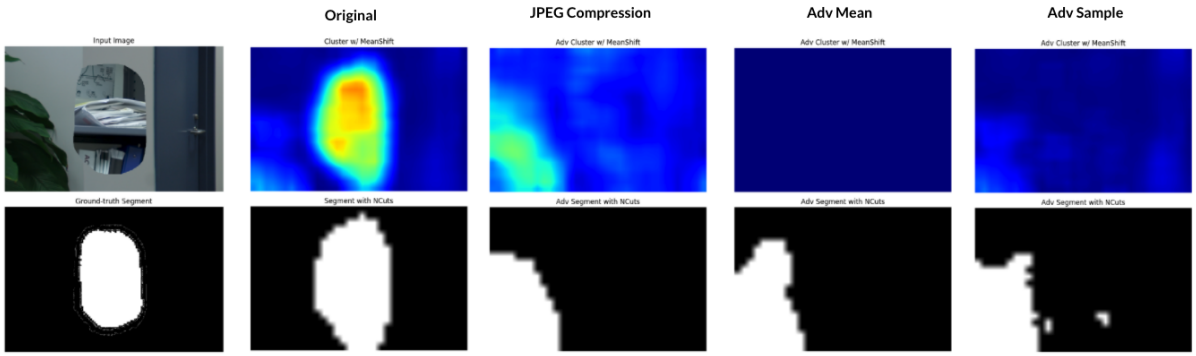
For Adv-Mean and Adv-Sample, a step size of 10,000 was used, with 50 iterations during optimization.

The quantitative results are reported in Table 2. All the different types of attacks (JPEG compression, Adv-Mean and Adv-Sample) do result in a noticeable decrease in test performance. In general, Adv-Sample has the most potent attack, followed by JPEG compression and then Adv-Mean.

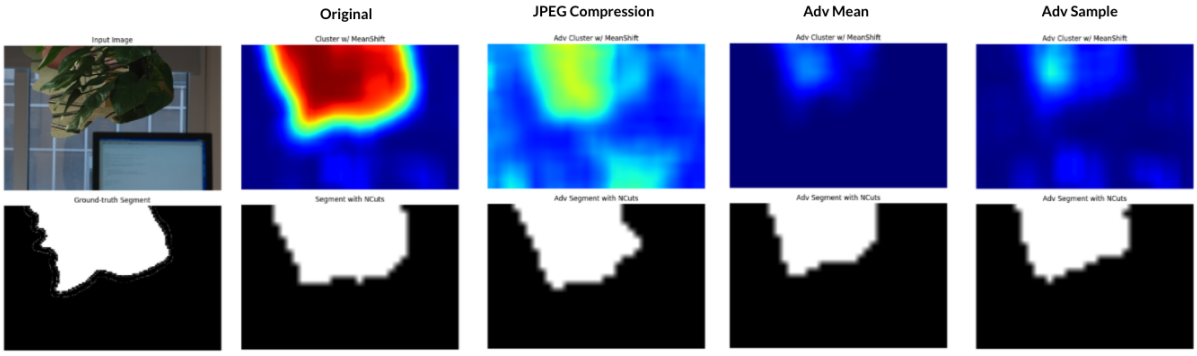
Some qualitative results are also shown in Figure 2. We can see how Adv-Mean results in a "smooth" localization map, predicting that the entire image is completely authentic, while Adv-Sample still leaves behind some traces of prediction noise. JPEG compression also does seem to easily disrupt the predictions of the model.

References

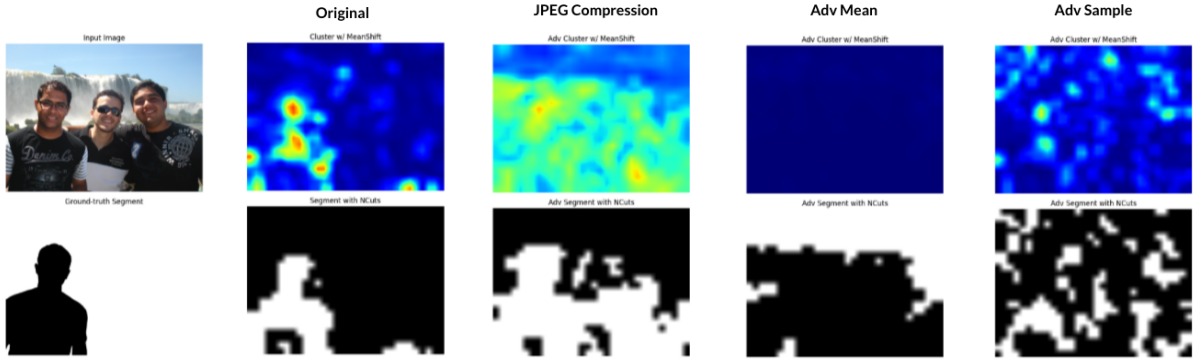
[1] T. Carvalho et al. "Exposing Digital Image Forgeries by Illumination Color Classification". In: *IEEE Transactions on Information Forensics and Security* 8 (2013), pp. 1182–1194.



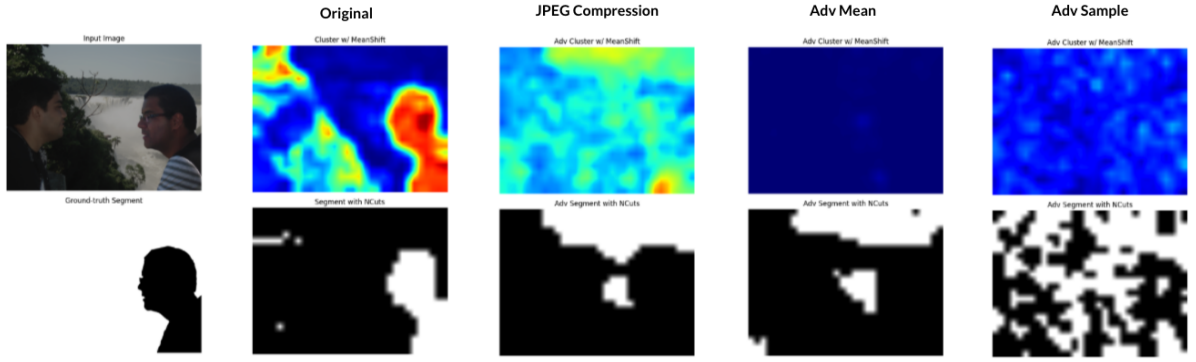
(a) Example 1 from Columbia Dataset



(b) Example 2 from Columbia Dataset



(c) Example 1 from DSO-1 Dataset



(d) Example 2 from DSO-1 Dataset

Figure 2: Example Predictions from Adversarial Attacks on EXIF-SC. From left to right: input image and ground-truth localization, original prediction on clean image, and then predictions on images attacked with JPEG Compression, Adv-Mean and Adv-Sample. Examples of the perturbed images are left out as they contain barely perceptible differences.

[2] C. Chen, Xinwei Zhao, and Matthew C. Stamm. "Generative Adversarial Attacks Against Deep-Learning-Based Camera Model Identification". In: *IEEE Transactions on Information Forensics and Security* (2019), pp. 1–1.

[3] D. Cozzolino et al. "SpoC: Spoofing Camera Fingerprints". In: *ArXiv* abs/1911.12069 (2019).

[4] James Hays and Alexei A. Efros. "Scene completion using millions of photographs". In: *Communications of the ACM* 51 (2008), pp. 87–94.

[5] Minyoung Huh et al. "Fighting Fake News: Image Splice Detection via Learned Self-Consistency". In: *ECCV*. 2018.

[6] M. Huiskes and M. Lew. "The MIR flickr retrieval evaluation". In: *MIR '08*. 2008.

[7] Dongkyu Kim et al. "Median Filtered Image Restoration and Anti-Forensics Using Adversarial Networks". In: *IEEE Signal Processing Letters* 25 (2018), pp. 278–282.

[8] Pawel Korus and J. Huang. "Evaluation of random field models in multi-modal unsupervised tampering localization". In: *2016 IEEE International Workshop on Information Forensics and Security (WIFS)* (2016), pp. 1–6.

[9] Gaël Mahfoudi et al. "DEFACTO: Image and Face Manipulation Dataset". In: *2019 27th European Signal Processing Conference (EUSIPCO)* (2019), pp. 1–5.

[10] T. Ng and S. Chang. "A Data Set of Authentic and Spliced Image Blocks". In: 2004.

[11] Ehsan Nowroozi et al. "A survey of machine learning techniques in adversarial image forensics". In: *Computers and Security* 100 (Oct. 2021). ISSN: 01674048. DOI: 10.1016/j.cose.2020.102092. arXiv: 2010.09680. URL: http://arxiv.org/abs/2010.09680.

[12] Andras Rozsa, Terrance E. Boult, and Zheng Zhong. "Adversarial attack on deep learning-based splice localization". In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* 2020-June (Apr. 2020), pp. 2757–2765. ISSN: 21607516. DOI: 10.1109/CVPRW50498.2020.00332. arXiv: 2004.08443. URL: http://arxiv.org/abs/2004.08443.

[13] Luisa Verdoliva. "Media Forensics and DeepFakes: An Overview". In: *IEEE Journal on Selected Topics in Signal Processing* 14.5 (Jan. 2020), pp. 910–932. ISSN: 19410484. DOI: 10.1109/JSTSP.2020.3002101. arXiv: 2001.06564. URL: http://arxiv.org/abs/2001.06564.

[14] Markos Zampoglou, Symeon Papadopoulos, and Yiannis Kompatsiaris. "Detecting image splicing in the wild (WEB)". In: *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 2015, pp. 1–6. DOI: 10.1109/ICMEW.2015.7169839.