PRAT: PRofiling Adversarial aTtacks
(Supplementary Material)

1. Overview

We include the following details in this supplementary,

1. **Motivation:** elaborates the motivation and applications of PRAT.
2. **AID creation:** discusses the details and methodology opted while creating the dataset
3. **Signature Extractor:** detailed working of the signature extractor, GLoF block and the T2I blocks.
4. **Experiments & Analysis:** includes further analysis on the performance of the proposed method, dictionary based indexing for novel attack identification.

2. Motivation

One principal application of PRAT is to provide information about the attacker which can be used to develop a defense beyond the abilities of the attacker, this will also nullify the possibility of an adaptive attack, hence making the target system more robust. For example, any existing large-scale vision model can be analyzed under PRAT problem to identify attackers and devise a defense technique(possibly by including more of those attacked samples in the adversarial training) to improve it. In most real-world scenarios the deep model would be presented with an image without any knowledge of the attacker. We agree that additional details like targeted/untargeted, black box/white box could possibly help the ML practitioner to build a better defense system, but for this work, we take a step to identify a new problem, provide extensive research analysis, and develop GLOF based model to address it. As part of future work, we expect to release a much larger version of AID with a much wider problem statement while the current version of AID still remains the baseline.

3. AID Creation

Benign images for the creation of adversarial samples are obtained from the validation set of ImageNet2012[6]. The validation set is split into two parts of size 45k and 5k samples corresponding to the train and test partitions of AID. We consider 13 different adversarial attacks targeting 3 different networks. For the train set, we randomly choose 4k images per attack per network and execute the attack strategies. Similarly, for the test set, we consider 800 images per attack per network. All the images are resized and cropped to a standard size of 224x224x3. Generated perturbations along with the information of the original benign sample used to generate the sample are saved to the disk. We summarize the details of AID in Table 1.

Parameter	**Details**
Dataset size	187.2k
Training samples	156k
Testing samples	31.2k
Per network train set	52k
Per network test set	10.4k
Total attacks	13
Toolchain families	3
Target networks	3
$\|\|\boldsymbol{\rho}\|\|_\infty$ range	$\{1, 16\}$
$\|\|\boldsymbol{\rho}\|\|_2$ range	$\{1, 10\}$

Table 1: AID statistics: AID includes several variants of adversarial perturbations generated by launching popular attacks on different networks using varied perturbation norms.

The attacks/families and the corresponding hyper-parameters have been chosen in accordance with the existing literature. To make the PRAT problem more interesting and challenging we have also included perturbations with budget as small as {1/255}. While there is always scope for including more attacks, we have chosen AID to consider maximum possible variations. In future, versions of AID could include PRAT problem focusing on other domains such as videos etc.

4. Signature Extractor

In this section, we give a detailed overview of the signature extractor highlighting the working of the GLoF block.

The signature extractor serves the purpose of disentangling the perturbation for the adversarial sample. Several GLoF blocks in series help reconstructing the benign sample which in turn generates the signature.

Consider an input adversarial image $\tilde{\boldsymbol{I}} \in \mathbb{R}^{H \times W \times 3}$ (H, W correspond to image height and width and 3 corresponds to the RGB channels). The standard GLoF block receives a series of token embeddings and a 2D feature map of the image. The input adversarial image $\tilde{\boldsymbol{I}}$ is reshaped into a series of 2D patches $\tilde{\boldsymbol{I}}_{\boldsymbol{p}} \in \mathbb{R}^{N \times P^2 \times 3}$, where P is the height and width of each patch and N is the number of patches/tokens $N = HW/P^2$. The patches are flattened along the feature dimension $\tilde{\boldsymbol{I}}_{\boldsymbol{p}} \in \mathbb{R}^{N \times (P^2 \cdot 3)}$. The attention arm along the GLoF blocks expects a constant embedding dimension, hence the patches are projected onto the embedding space of dimension $\boldsymbol{D_1}$. As proposed in [2], adding position embeddings $\boldsymbol{E} \in \mathbb{R}^{N \times D_1}$ to the patch embeddings help in retaining the relative position of the patches in the 2D space. The resulting patch embedding is termed $\boldsymbol{T_0} \in \mathbb{R}^{N \times D_1}$ (0 referring to the initial feature level and N referring to the number of patches).

$$\boldsymbol{T_0} = \tilde{\boldsymbol{I}}_{\boldsymbol{p}} + \boldsymbol{E}; \quad \tilde{I}_p, E \in \mathbb{R}^{N \times D_1} \tag{1}$$

Alongside, the input image is projected to an embedding dimension $\boldsymbol{D_2}$, by applying a 3×3 Conv with $\boldsymbol{D_2}$ features. We term these features $\boldsymbol{Z_0} \in \mathbb{R}^{H \times W \times D_2}$ (0 referring to the initial feature level). Features extracted from previous level($l-1$) are passed on to the next GLoF block.

$$\boldsymbol{T_l}, \boldsymbol{Z_l} = GLoF(\boldsymbol{T_{l-1}}, \boldsymbol{Z_{l-1}}); \quad l = 1...L \tag{2}$$

Where L is the number of GLoF blocks. The output of the final GLoF block corresponding to the convolutional arm $\boldsymbol{Z_l}$ is transformed to RGB space by applying a 3×3 Conv with 3 feature maps resulting in the rectified image $\boldsymbol{I_r} \in \mathbb{R}^{H \times W \times 3}$. Finally, to extract the signature from the rectified image, difference of the rectified and the original image is considered $\tilde{\boldsymbol{\rho}} = \tilde{\boldsymbol{I}} - \boldsymbol{I_r}$.

4.1. GLoF Module

Global-LOcal Feature extractor(GLoF) module combines CNN's ability to extract local features with vision transformer's global connectivity.

The GLoF block at any level receives the local and global features from the previous level. The GLoF block at level l receives the image features $\boldsymbol{Z_{l-1}} \in \mathbb{R}^{H \times W \times D_2}$ and the embedded tokens $\boldsymbol{T_{l-1}} \in \mathbb{R}^{N \times D_1}$ where $\boldsymbol{T_{l-1}} = \{\boldsymbol{T^1_{l-1}}, \boldsymbol{T^2_{l-1}}, ..., \boldsymbol{T^N_{l-1}}\}$ (N being the number of tokens) as inputs.

Global features: Embedded tokens are fed to attention mechanism. Tokens are interpreted as Query $\boldsymbol{Q} \in \mathbb{R}^{N \times D_1}$, key $\boldsymbol{K} \in \mathbb{R}^{N \times D_1}$ and value $\boldsymbol{V} \in \mathbb{R}^{N \times D_1}$. Attention is calculated by measuring the weighted sum over all values as,

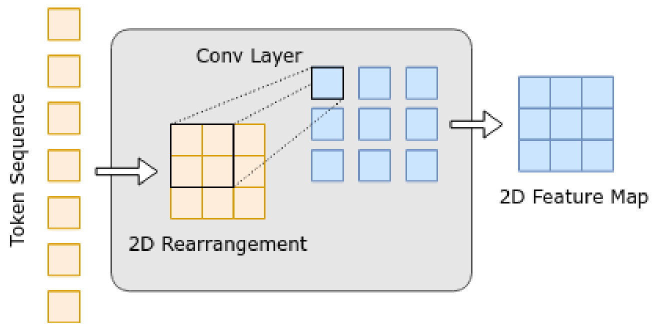

Figure 1: **T2I Block Overview:** Input token sequence is transformed to a 2D representation. Convolutional layers are applied to obtain the 2D feature map.

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\boldsymbol{Q}\boldsymbol{K}^{\boldsymbol{T}}/\sqrt{d})\boldsymbol{V} \tag{3}$$

Multi-head attention performs the attention *num_heads* times in parallel, concatenates and linearly projects it. A residual skip connection connects the tokens with the attention map.

$$\boldsymbol{T^a_l} = Add(Attention(Norm(\boldsymbol{T_{l-1}})), \boldsymbol{T_{l-1}}) \tag{4}$$

$$\boldsymbol{T^f_l} = GELU(\boldsymbol{T^a_l}\boldsymbol{W_1} + b_1)\boldsymbol{W_2} + b_2 \tag{5}$$

where $\boldsymbol{W_1} \in \mathbb{R}^{D_1 \times \mathfrak{D}}$ and $\boldsymbol{W_2} \in \mathbb{R}^{\mathfrak{D} \times D_1}$ are weights of the two linear layers. The first linear layer projects the token to a higher dimension $\mathfrak{D}$ for better learning. $b_1 \in \mathbb{R}^{\mathfrak{D}}$ and $b_2 \in \mathbb{R}^{D_1}$ are the bias terms for the two linear layers.

Local features: Embedded 2D image features from the previous layer $\boldsymbol{Z_{l-1}}$ are fed to a standard ResNet block with convolutional, batch norm and activation layers.

$$\boldsymbol{Z^{c_1}_l} = Act(Norm(Conv(\boldsymbol{Z_{l-1}}))) \tag{6}$$

$$\boldsymbol{Z^{c_2}_l} = Act(Norm(Conv(\boldsymbol{Z^{c_1}_l}))) \tag{7}$$

$$\boldsymbol{Z^{c}_l} = Add(\boldsymbol{Z_{l-1}}, \boldsymbol{Z^{c_2}_l}) \tag{8}$$

where Conv, Norm and Act refer to convolutional layer, normalization layer and non-linear activation.

T2I Block: T2I block transforms the patches to a 2D feature map which allows merging with convolutional feature maps. Fig.1 shows the overview of a T2I block.

$$\boldsymbol{T^h_l} = \boldsymbol{T2I}(\boldsymbol{T^f_l}) \tag{9}$$

$$\boldsymbol{T^{h'}_l} = Transform(\boldsymbol{T^f_l}); \quad \boldsymbol{T^h} \in \mathbb{R}^{H \times W \times d'} \tag{10}$$

$$\boldsymbol{T^{h'}_l} = Act(Norm(Conv(\boldsymbol{T^{h'}_l}))) \tag{11}$$

$$\boldsymbol{T^{h}_l} = Act(Norm(Conv(\boldsymbol{T^{h'}_l}))) \tag{12}$$

where Transform function rearranges the series of tokens to form a 2D feature map $T_l^{h'} \in \mathbb{R}^{H \times W \times d'}$. Conv., Norm and Act layers are applied over $T_l^{h'}$ resulting in $T_l^h \in \mathbb{R}^{H \times W \times D_2}$ which is merged with the features from the convolution arm of the GLoF block.

$$\boldsymbol{Z_l} = Add(\boldsymbol{Z_l^c}, \boldsymbol{T_l^h}) \tag{13}$$

The merged features Z_l in conjunction with tokens Z_l are passed on to consecutive GLoF blocks.

Signature: To extract the signature $\tilde{\boldsymbol{\rho}}$, the output of the final GLoF block is subtracted from the input adversarial image.

$$\tilde{\boldsymbol{\rho}} = \boldsymbol{Z_l} - \tilde{\boldsymbol{I}} \tag{14}$$

The generated signature is fused with the input adversarial image and fed to the attack classifier.

5. Experiments & Analysis

We explore how **different attack classifiers** affect the overall performance. As seen in Table 3, the signature extractor paired with DenseNet121 [4] yields the best results. It can be observed that the attack classifiers paired with signature extractor (Table 3) performs significantly better compared to training stand alone classifiers. This supports the claim that extracting input-specific signature form the adversarial input to identify the attack is a better strategy.

We analyze class wise scores and the confusion matrix of the predictions from the proposed approach in Fig **??** and Table 2. From the confusion matrix, we observe the common trend of relatively high scores for all decision based attacks except for boundary attack. With scores close to 1, these attacks have distinctive patterns which are being easily identified by the signature extractor. Boundary attack do not always have specific patterns because of the way they are generated. Starting from a point that is already adversarial, boundary attack performs a random walk on the decision boundary minimizing the amount of perturbation. Similarly, universal attacks generate discernible patterns making it easier for detection. Major confusion occurs in the gradient based attacks among NewtonFool, DeepFool and CW attack. These attacks being highly powerful, are targeted on generated nearly imperceptible perturbations specific to the input image, making it difficult for the method to identify and distinguish. Similar trends observed in the confusion matrix can be seen in Table 2.

5.1. Detecting clean vs. perturbed

While the core idea of PRAT is to profile the attacker given an adversarial image, it is likely for the signature extractor to be tested with clean images in real world scenarios. We devise an experiment to analyze the performance of the signature extractor in distinguishing clean from perturbed images. We consider a subset of AID containing adversarial images and similar size set of clean images. The signature extractor is trained to extract signatures highlighting the patterns in adversarial images. Extracted signatures are used to train a binary classifier that identifies clean and perturbed images. We use a standard ResNet50[3] as the binary classifier in this case. The end to end pipeline yields a **100%** accuracy in distinguishing perturbed images from clean images. This can act as a preliminary step, and if a perturbation is detected, it can be passed to the attack classifier for identifying the specific attack category.

label	Attack Method	Precision	Recall	F1-score
0	PGD	0.90	0.82	0.86
1	BIM	0.95	0.85	0.89
2	FGSM	0.86	0.90	0.88
3	DeepFool	0.49	0.51	0.50
4	NewtonFool	0.59	0.64	0.61
5	CW	0.48	0.46	0.47
6	Additive Gaussian	1.00	1.00	1.00
7	Gaussian Blur	0.90	0.91	0.90
8	Salt&Pepper	0.97	0.94	0.95
9	Contrast Reduction	0.92	0.97	0.94
10	Boundary	0.49	0.53	0.51
11	UAN	1.00	1.00	1.00
12	UAP	0.95	0.88	0.91

Table 2: Classification report(Precision, Recall, F1-score) of the proposed network on AID.

Method	Accuracy
SigExt. +	
ResNet50[3]	73.80%
ResNet101[3]	74.74%
ResNet152[3]	73.25%
DenseNet121[4]	**80.14%**
DenseNet169[4]	78.15%
DenseNet201[4]	76.69%
InceptionV3[7]	70.38%

Table 3: Performance of different attack classifiers with Signature Extractor

5.2. Success rate vs. Identifiability

While the stronger attacks like PGD have a 100% fooling rate, the black box attacks have a success rate of at least 65% for the samples considered in AID. We also study the identifiability vs success rate for the FGSM class and find that our technique achieves 74.9% accuracy for an epsilon as low as 2 and 94.5% for an epsilon of value 16. We observe an upward increasing trend as the epsilon increases indicating an increasing level of perceptibility of the patterns.

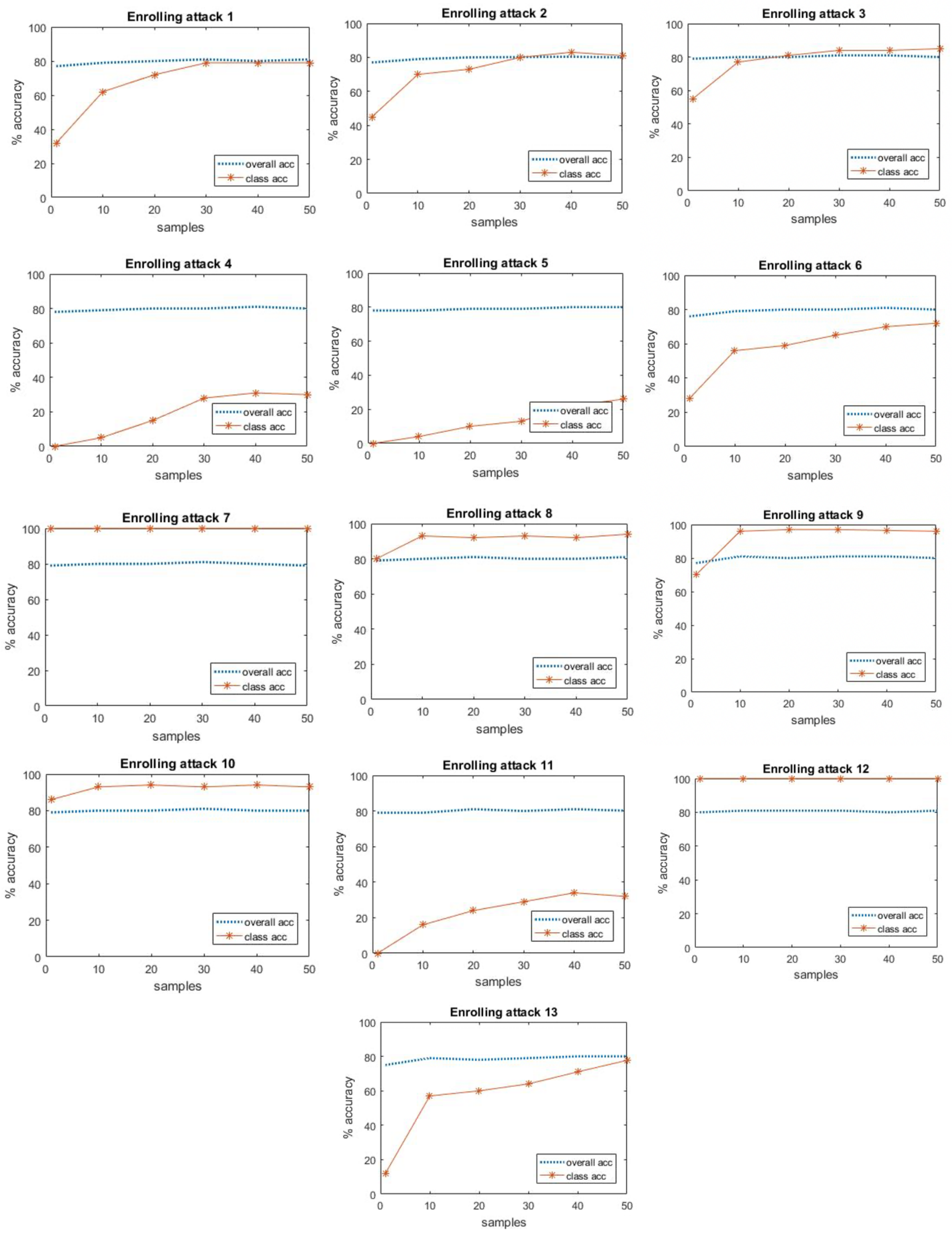

Figure 2: Results of considering individual classes as the unknown class

5.3. Enrolling novel classes

With the fast moving field of adversarial machine learning, it is highly likely for the signature extractor to come across novel unseen attacks. While, it is difficult to retrain the signature extractor and the attack classifier each time a new attack is added to the system, we employ a dictionary based toolchain indexing scheme to enrol novel attack classes with limited data.

We use a simple indexing scheme which can work as an addition to the existing signature extraction approach. The extracted signature for the adversarial images is of size 24x224x3. Since, storing and indexing such large images requires large amounts of memory and computations, we project the signature to a 512-dimensional using the model activations. These are extracted from the penultimate layer of a standard DenseNet-121 network. To index these compressed signatures into a dictionary, it is required to assign the correct toolchain to the sample. To solve this problem, we adapt a sparse and collaborative representation based classifier[1]. This classifier expects training data, which is our dictionary and the test sample that we need to index in the dictionary. The produced label is the label of the adversarial attack in our case, which is identifiable because our dictionary is structured. We use Orthogonal Matching Pursuit(OMP)[5] algorithm to compute sparse codes for a given sample over a fixed dictionary. It tends to assign large coefficient values in the sparse codes corresponding to the dictionary elements that are closely correlated to the test samples. The algorithm does not make any assumption about the dictionary itself. Hence, it does not restrict us from enrolling new attacks (or their families) to the dictionary.

To enroll a novel attack, we adopt a similar strategy. The main challenge for the indexing scheme is to register the novel attack with limited data. Hence, the indexing challenge gets translated into maintaining reasonable classification performance with one or very few samples for the unknown class. For analysis, we sequentially consider each of the thirteen attacks as the ‘unknown attack’ and note the performance of indexing scheme with varied number of samples available from the known attacks. We consider a subset of 50 samples per class from AID for the experiment. Starting with enrolling as little as a single sample for the class, we analyze the performance when we have 10, 20, 30, 40 and 50 samples for a newly enrolled class. The corresponding plots are shown in Fig 2.

From the plots, it can be observed for PGD, BIM and FGSM the indexing technique achieves accuracies greater than 60% with just 10 training samples. With as low as a single training sample, accuracy is consistently above 30%. For relatively simple classes like Gaussian Blur and UAP, we were able to maintain 100%. For particularly challenging classes like NewtonFool, CW and UAN, more samples resulted in better performance. These results demonstrate that the degradation in performance of our indexing scheme in the case of fewer training samples is graceful, to the extent that average accuracy across all classes with a single training sample is 46%. Hence, we can claim that the scheme has the ability of enrolling new attack effectively with as little as a single sample for most of the unknown attacks.

References

[1] Naveed Akhtar, Faisal Shafait, and Ajmal Mian. Efficient classification with sparsity augmented collaborative representation. *Pattern Recognition*, 65:136–145, 2017. 5

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3

[5] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE, 1993. 5

[6] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 3