APD: Boosting Adversarial Transferability via Perturbation Dropout [1]

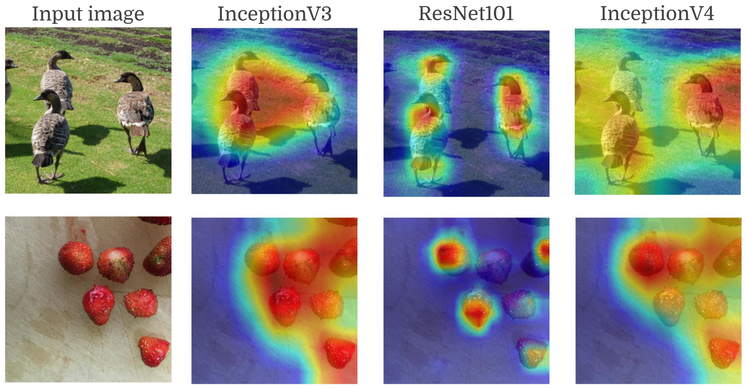
Poster by Germain Vivier-Ardisson and Guillaume Sallé

SORBONNE UNIVERSITÉ

Introduction

Convolutional neural networks (CNNs) are complex models that are known to lack robustness against **adversarial attacks:** small **gradient-based perturbations** that aim at flipping the model's output by imperceptibly modifying its input image x. In a **white-box setting**, the attacker has access to the model's weights, and can thus compute gradients with respect to its input. However, crafting an adversarial attack is much more difficult in a **black-box setting**, where the weights of the model are inaccessible.

Adversarial transfer aims at attacking a black-box, target CNN model M^* by crafting a white-box attack x^{adv} on a known, source model M_0 and feeding it to M^*, hoping that the induced perturbations will also mislead the target model. However, adversarial perturbations computed from a source model highly depend on the **attention regions** of the model for the considered input. Thus, differences between attention regions of M_0 and M^* may lead to poor adversarial transferability of x^{adv}, as the perturbations could fall into low attention regions of M^*.



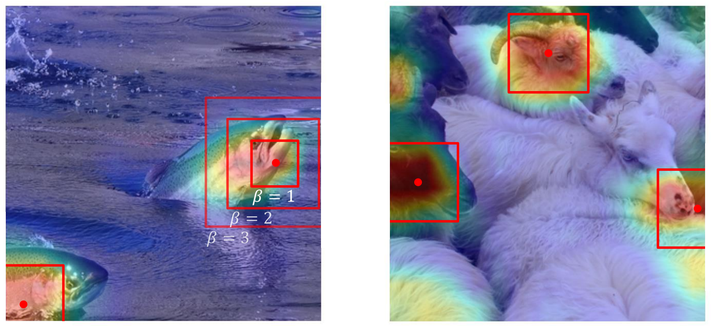
Attention regions as obtained with GradCAM++

APD aims at boosting the transferability of adversarial attacks by making them more model-agnostic. To do so, it uses **Class Activation Maps** (CAM) to locate midpoints of the attention regions of M_0, and prevents the adversarial perturbations from different regions to co-adapt, by **dropping them out** during the optimization process. The method is inspired by dropout training of neural networks, that aims at decoupling the weights to achieve better generalization, which translates to better transferability of the attack in our setting.

APD: Methodology

Selecting Dropped-Out Regions

The first step of APD consists in selecting the attention regions that will be dropped-out during the optimization process. It does so with **GradCAM++**, a method which uses a weighted combination of the positive partial derivatives of the last convolutional feature maps of M_0 with respect to the label as weights to generate an activation map. Once the CAM is computed, APD chooses the dropped-out regions as $n \leq 3$ squares, centered on local activation maxima.



Examples of dropped-out regions selected by APD with ResNet101

Averaging Gradients

The perturbation dropout scheme appears inside the gradient computation step of the adversarial attack. Let $\nabla J(x_t^{adv}, \text{ytr}^{\text{ue}})$ be the gradient of the loss function of the classifier M_0 with respect to the adversarial attack at time t of the optimization process, x_t^{adv}; and $Clip_\epsilon^x$ the projection on the $||.||_\infty$ ball of radius ϵ centered on the initial input image x. Instead of updating x_t^{adv} as

$$x_{t+1}^{adv} = \text{Clip}_\epsilon^x[x_t^{adv} + \alpha \times sign(\nabla J(x_t^{adv}, y^{true})],$$

which is the Fast Gradient Signed Method (FGSM [2]) iteration, APD with same baseline updates the adversarial attack as

$$x_{t+1}^{adv} = \text{Clip}_\epsilon^x[x_t^{adv} + \alpha \times sign(\nabla J^{drop})]$$

With $\nabla J^{drop} = \frac{1}{n \times m} \sum_{j=1}^{n} \sum_{k=1}^{m} \nabla J(x_{tjk}^{drop}, y^{true})$, where n is the number of selected local maxima and x_{tjk}^{drop} is the image obtained by dropping out the adversarial perturbations on the square of semi side-length $\beta \times k$ centered at the j –th local maximum. β and m are hyperparameters. Thus, ∇J^{drop} is the average of the gradients computed over the attack with growing dropped-out regions.

Numerical Experiments

We re-implement APD with MI-FGSM (FGSM with momentum iterations [4]) as chosen baseline. We provide comparisons with MI-FGSM without perturbation dropout. We use InceptionV3 (IncV3), IncV4, Inc-ResNet-V2 and ResNet101 as both white-box source models and black-box target models. We also use adversarially-trained IncV3 as a bonus target model.

Source \ Target	Inc-V3	Inc-V4	IncRes-V2	ResNet101	AdvInc-V3
Inc-V3	A: 100 / 98.3 MI: 100 / **100**	A: **67.2** / 38.2 MI: 50 / **84.3**	A: **67.4** / 32.5 MI: 47.8 / **99.9**	A: **57.5** / 17.1 MI: 38.1 / **62.4**	A: **22.7** / 36.6 MI: 14.4 / **60.4**
Inc-V4	A: **80.7** / 63.8 MI: 67.2 / **85.4**	A: 100 / 100 MI: 100 / 100	A: **70.7** / 53.6 MI: 50.5 / **99.8**	A: **65.2** / 34.6 MI: 45.4 / **58.8**	A: **22.3** / 39.3 MI: 12.9 / **59.4**
IncRes-V2	A: **82.7** / 64.7 MI: 66.2 / **82.1**	A: **77.1** / 64 MI: 57.3 / **70.7**	A: 99.8 / 99.9 MI: 99.9 / **100**	A: **67.5** / 38.7 MI: 46.1 / **47.3**	A: **25.9** / 45 MI: 14.2 / **57.1**
ResNet101	A: **79** / 60.4 MI: 67.6 / **79.1**	A: **72.8** / 58.1 MI: 61.8 / **80.1**	A: **73.1** / 51.2 MI: 59.8 / **95.7**	A: 99.6 / 99.7 MI: **99.9** / **100**	A: **28.6** / 43.8 MI: 18.6 / **60.1**

Attack Success rates (%). Announced and obtained results.
"MI" is MI-FGSM and "A" is MI-FGSM+APD. Bold denotes best method.

Discussions

Our results with APD are way below what is announced in the paper. Moreover, our results without APD (MI-FGSM, with the same hyperparameters as when used in the article) are way better than those announced, and empirically constantly better than with APD.
This questions both the experimental methodology employed by the authors and the effectiveness of their approach.

References

[1] APD: Boosting Adversarial Transferability via Perturbation Dropout, Wang et. al (2024)

[2] Explaining and Harnessing Adversarial Examples, Goodfellow et. al (2015)

[3] Improving neural networks by preventing co-adaptation of feature detectors, Hinton et. al (2012)

[4] Boosting Adversarial Attacks with Momentum, Dong et. al (2018)