Supplementary Information: Probing for Artifacts

March 14, 2020

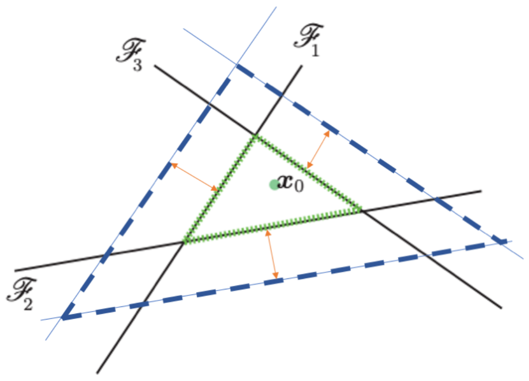

Figure 1: Shifted level curves from [1]. Dashed blue are solutions to $\mathscr{F}_k = \{x : f_k(\boldsymbol{x}) - f_4(\boldsymbol{x}) - c(i) = 0\}$ where $c(i)$ is a positive constant we call *confidence* to be consistent with the language of Carlini and Wagner. Dashed green lines are the original solution set to $\mathscr{F}_k = \{x : f_k(\boldsymbol{x}) - f_4(\boldsymbol{x}) = 0\}$ prior to introducing a constant. Orange arrows are not necessarily equal length.

1 Modifications to DeepFool

In this section, we refer the reader to [1] for details of notation and methodology. A multi-label classifier perturbation solution set examined by [1] (Equation 6) is:

$$\text{argmin } ||\boldsymbol{r}||_2 \text{ s.t. } \exists k : \boldsymbol{w}_k^T(\boldsymbol{x_0}+\boldsymbol{r})+b_k \geq \boldsymbol{w}_{k(\boldsymbol{x_0})}^T(\boldsymbol{x_0}+\boldsymbol{r}) + b_{k(\boldsymbol{x_0})}, \quad (1)$$

so that a perturbation is a valid solution to the proposed DeepFool if the classifier has an evasive label change. Please refer to[1] for details. The issue is that the true arg minimum perturbation, that is to say the exact solution, is usually very close to the decision boundary. DeepFool is an iterative algorithm and [1] suggest using an overshoot on the last iteration to avoid boundaries. We found that approach worked well in floating-point precision but not as well as unsigned byte RGB images.

We modify the original DeepFool equation to be solutions of the form:

$$\text{argmin } ||\boldsymbol{r}||_2 \text{ s.t. } \exists k : \boldsymbol{w}_k^T(\boldsymbol{x_0}+\boldsymbol{r})+b_k \geq \boldsymbol{w}_{k(\boldsymbol{x_0})}^T(\boldsymbol{x_0}+\boldsymbol{r}) + b_{k(\boldsymbol{x_0})} + c_i, \quad (2)$$

where notation is as in [1]; however, we add a constant c_i to the criteria for a valid solution to the argument minimum. We do not assume that there exists an $\boldsymbol{r}$ that is a solution to this criteria, and to get around that limitation, we gradually decrease c_i in an upper bound M to 0 until a solution is found. At $c_i = 0$ this algorithm is the original DeepFool. Compared to the original "overshoot on the last iteration," this method tries to be explicit about how far past a decision boundary an overshoot should be by building it into the solution set.

To make the algorithm run efficiently, we decay c_i in the DeepFool optimization loop. Decaying c_i inside of the DeepFool optimization loop makes solutions heuristics and not necessarily the true L_2 minimizing perturbation, but we have observed that it works well in practice. The schedule decay is a function with $c_0 = c(0) = M$ and some future iteration k such that for all $i > k$, $c(i) = 0$. We scaled $c(i)$ linearly from 0.10 to 0.0 over the 100 max iterations of DeepFool.

The modified Algorithm 2 from [1] is recorded here as

Procedure 1 with logit$(\boldsymbol{x}, k(\boldsymbol{x}))$ being $f(\boldsymbol{x})$ for the argmax of an image $\boldsymbol{x}$. One important operational note on this modified Procedure 1 comes from the lines 4 and 6 with and without $-c(i)$ present in the algorithm. When $-c(i)$ is present, the absolute value of the arg min is minimized for classes that are near the "further away boundary," and when it is absent, the nearest decision boundary will do just fine. The net effect is the algorithm heads towards appropriate solutions to the modified acceptance criteria. The modified acceptance criteria is gradually relaxed in importance as the algorithm proceeds to guarantee the stopping criteria can be achieved.

Our application of DeepFool was to create evasive perturbations to Imagenet model input. Images begin on disk as JPEG. They are loaded into memory as RGB unsigned bytes: b. They are resized to model resolution: $r = \text{resize}(b)$. They are transformed to model input: $x = \text{preprocess}(r)$. After transformation DeepFool is applied to create $x_a = g(x)$, where $g(x)$ is the adversarial evasion, and then the whole process is inverted, omitting resize, reapplied, and adversarial status is rechecked. $r_a = \text{preprocess}^{-1}(x_a)$, the inverse of the model preprocessing, still as float32. $b_a = \text{cast to ubyte}(r_a)$. We were interested in b_a that remained adversarial after the inverse and the cast to ubyte, so then we proceeded by re-examining the adversarial status of $x'_a = g(\text{preprocess}(b_a))$. We found two pecularities with DeepFool. First often $|b_a - \text{cast to ubyte}(r)| = 0$, meaning that the inverted adversarial image was exactly the same as the resized image as unsigned bytes ($L_1 = 0$), and x'_a failed to remain evasive. It is worth noting that the cleverhans implementation of DeepFool is fine, the x_a float32 perturbed image is usually truly evasive, but only just barely, and it the perturbation does not often remain evasive after casting to 24-bit RGB.

2 Adversary Mean Distortions

The average RGB perturbation in successful adversarial evasions is presented in Table 2. The smallest distortions in this data set is on Resnet50 DeepFool with an average per pixel channel change of 0.024. On the same model, CWL2 created distortions 66x larger on average. The probe model also failed to detect Resnet50-DeepFool at a recall rates larger than a controlled FPR. This is unlikely to be a coincidence. 0.024 is a very small average distortion.

Also, on average RGB L_1-norm of evasive distortions are larger on 10-class variants. This also makes intuitive sense because more work has to be done by the adversary to reach a reduced set of decision boundaries.

3 FC2 Ablation

We studied the impact of adding a second fully connected (FC) layer prior to a softmax output. As indicated in we were unable to show any advantage for the second layer. The only surprising thing about that is the 2 FC layer model has more capacity than the 1 FC layer model. Though, if it were only a matter of adding more parameters to models, then the largest model would always be state-of-the-art, and experience teaches that is just not true. FC2 remains an important contrast to adding MobileNet-v2 to the model because in that case results were markedly improved for having more capacity.

References

[1] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

Procedure 1 Modified DeepFool

```
Input: Image $\boldsymbol{x_0}$, classifier $f$ and scheduled decay $c(i)$.
Output: Perturbation $\hat{\boldsymbol{r}}$.
1: while $k(\boldsymbol{x_i}) = k(\boldsymbol{x_0})$ or logit$(\boldsymbol{x_i}, k(\boldsymbol{x_i})) \leq$ logit$(\boldsymbol{x_0}, k(\boldsymbol{x_0})) + c(i)$ do
2:     for $k(\boldsymbol{x}) \neq k(\boldsymbol{x_0})$ do
3:         $\boldsymbol{w}'_k \leftarrow \Delta f_k(\boldsymbol{x_i}) - \Delta f_{k(\boldsymbol{x_0})}(\boldsymbol{x_i})$
4:         $f'_k \leftarrow f_k(\boldsymbol{x_i}) - f_{k(\boldsymbol{x_0})}(\boldsymbol{x_i}) - c(i)$
5:     end for
6:     $l \leftarrow \text{arg min}_{k \neq k(\boldsymbol{x_0})} \frac{|f'_k|}{||\boldsymbol{w}'_k||_2}$
7:     $\boldsymbol{r_i} \leftarrow \frac{|f'_k|}{||\boldsymbol{w}'_k||_2^2} \boldsymbol{w}'_l$
8:     $\boldsymbol{x_{i+1}} \leftarrow \boldsymbol{x_i} + \boldsymbol{r_i}$
9:     $i \leftarrow i + 1$
10: end while
```

DeepFool Algorithm	Succ. Mean L_1	Succ. Std. Dev. L_1	Tried Evasion	24-bit Successful Evasion	24-bit Success Rate
Baseline cleverhans	0.64	-	993	2	< 0.01
Modified $M = 0.01$	0.11	0.11	995	903	0.91
Modified $M = 0.05$	0.11	0.11	991	932	0.94
Modified $M = 0.15$	0.16	0.16	994	991	> 0.99

Table 1: DeepFool succcess rates after inverting the preprocess, casting the adversarial perturbation to unsigned byte, and then reapplying model preprocessing. Images are 1000 random sampled Imagenet images from 10-classes. Each row had its own random sample. Tried is less than 1000 images because some images were discarded due to being greyscale or 4-color channels. The successes for unmodified reference cleverhans were 2 out of 993. The successses of the modified cleverhans were 91% to 100% depending on initial value of δ. All applications of DeepFool utilized 100 iterations and an overshoot on the last iteration of 1.05.

Algorithm	Model	Successful Evasion	Success Rate	Mean L_1
FGSM	xception	7937	0.772	1.959
FGSM	vgg16	10010	0.973	1.963
FGSM	slice-10 vgg16	6104	0.593	1.969
FGSM	resnet50	9850	0.958	1.961
FGSM	inception-resnet-v2	5919	0.575	1.959
FGSM	slice-10 inception-v3	2903	0.282	1.963
FGSM	inception-v3	7994	0.777	1.960
DeepFool	xception	10286	1.000	0.100
DeepFool	vgg16	10286	1.000	0.054
DeepFool	slice-10 vgg16	10286	1.000	0.349
DeepFool	resnet50	10286	1.000	0.024
DeepFool	inception-resnet-v2	10286	1.000	0.183
DeepFool	slice-10 inception-v3	10286	1.000	0.226
DeepFool	inception-v3	10286	1.000	0.089
CWL2	xception	10286	1.000	1.594
CWL2	vgg16	10245	0.996	1.559
CWL2	slice-10 vgg16	8555	0.832	2.459
CWL2	resnet50	10092	0.981	1.602
CWL2	inception-resnet-v2	10286	1.000	1.840
CWL2	slice-10 inception-v3	10286	1.000	2.690
CWL2	inception-v3	10286	1.000	1.655

Table 2: Success rates and mean perturbation sizes for select training adversarial images. Number training images attempted by each algorithm and model combination (row) is 10286. An adversary is considered successful only if the pre-processing of the 24-bit RGB is evasive to the former top-1 label. FGSM is parameterized with $\epsilon = 2.0$. DeepFool is modified with confidence parameter $M = 0.10$ decaying over 100 iterations, and Carlini-Wagner L2 has confidence parameter 0.10 with a learning rate of 0.001. Mean L_1-norm is on unsigned bytes (24-bit) RGB images. It is a mean over successful evasions of mean distortion per image. Slice-10 is the 10-class Imagenet models created by slicing logits prior to softmax.

model	FPR	FGSM $\epsilon = 2.0$	CWL2 conf.=0.1	DeepFool conf.= 0.1
xception-probe-16-FC1	0.01	0.92	0.43	0.13
xception-probe-16-FC2	0.01	0.91	0.43	0.12
xception-probe-32-mobilenet-v2	0.01	0.92	0.46	0.18
xception-probe-16-FC1	0.05	0.99	0.56	0.32
xception-probe-16-FC2	0.05	0.97	0.55	0.29
xception-probe-32-mobilenet-v2	0.05	0.98	0.58	0.40
xception-probe-16-FC1	0.10	1.00	0.65	0.45
xception-probe-16-FC2	0.10	0.99	0.63	0.43
xception-probe-32-mobilenet-v2	0.10	0.99	0.65	0.50
xception-probe-16-FC1	0.20	1.00	0.75	0.60
xception-probe-16-FC2	0.20	1.00	0.73	0.59
xception-probe-32-mobilenet-v2	0.20	1.00	0.75	0.65

Table 3: Ablation results for Xception probe model. The FC2 classifier has no advantage over FC1 in many scenarios. The $K = 32$ probe into a mobilenet is more powerful than the lower capacity models for difficult to detect perturbations such as DeepFool which on average were an order of magnitude smaller than CWL2. DeepFool averages less than a 1 unit RGB change per pixel. This table justifies the claim that a confidence parameterized DeepFool is very difficult to detect. It uses an order of magnitude less L_1 difference to accomplish evasion compared to CWL2.

Name	Sequential Operations on r	Description

Model-Probe-K-FC	Dense(512, activation=ReLU)
	BatchNormalization()
	DropOut(rate=0.50)
	Dense(2, activation=ReLU)
	Softmax()
		Minimal probing into a dense classifier as in Figure **??**.

Model -Probe-K-FC2	Dense(512, activation=ReLU)
	BatchNormalization()
	DropOut(rate=0.50)
	Dense(512, activation=ReLU)
	Dense(2, activation=ReLU)
	Softmax()
		Probing followed by 2 dense activations.

Model -Probe-32-MobileNet-v2	Reshape(N, K, 1)
	MobileNet-v2($\alpha = 1.0$)
	Dense(512, activation=ReLU)
	BatchNormalization()
	DropOut(rate=0.50)
	Dense(2, activation=ReLU)
	Softmax()
		32 unit probing followed by reshape into image like input into unlocked MobileNet-v2 for convolution artifact detection.

Table 4: Subsequent classifier construction on r probed features constructed by concatenating $f_i(v_i|\theta_i, K)$.

Name	Probed Layers (N)	Trainable Params	Non-Trainable Params	Total Params
xception-probe-8 FC	91	1605306	21108360	22713666
xception-probe-16 FC	91	2962194	21108360	24070554
xception-probe-16 FC_2	91	3224850	21108360	24333210
xception-probe-32-mobilenet-v2	91	7043842	21122632	28166474
resnet50-probe-32-mobilenet-v2	121	7252270	23861068	31113338
inception-v3-probe-32 mobilenet-v2	216	6676290	22042080	28718370

Table 5: Probe model counts of probed layers and parameters. The name contains the probe dimension K. For example, xception-probe-16-fc is a locked Xception model with probes with output dimension K=16 into a fully connected dense classifier.