Applying Universal Adversarial Perturbations to Vision Model Architectures

Andy Chen[1], **Luke Miga**[1], **Luke Yang**[1], **Nathan Yap**[1]

[1]University of Michigan
{anche, ldcmiga, lukeyang, nyap}@umich.edu

Abstract

Universal Adversarial Perturbations (UAPs) (Dosovitskiy et al. 2020a) are subtle changes that can be applied to images to cause them to be wrongly classified by image classifiers with high probability. In their original context, UAPs were shown to be extremely effective in fooling traditional deep neural network (DNN) architectures. However, recent advancements in the Vision Transformer (Moosavi-Dezfooli et al. 2016a) architecture have resulted in a new popular architecture whose effectiveness against UAPs was not tested in the original paper. As such, we first define and implement an analogue to the original UAP architecture on modern frameworks and libraries, then analyze the effectiveness of these generated perturbations on Vision Transformer models. In addition, we compare the results of our findings to that of the original UAP paper in order to contextualize our results, and see that UAPs remain quite robust in fooling Vision Transformers across a variety of datasets. We then analyze the impact that our extension can bring on a larger scale to understand the profound impact upon society that these UAPs can have.

Introduction

Image classification is a fundamental problem in safety-critical domains like autonomous driving systems, facial recognition and surveillance, and quality control. In these settings, vulnerabilities in classification models are particularly important to understand. One way to evaluate worst-case performance is to simulate an adversarial setting in which an adversary tries to make a possibly small or constrained change to an image to try to cause the image to be classified incorrectly.

The paper (Moosavi-Dezfooli et al. 2016b) showcases the vulnerabilities of deep neural networks (DNNs) for image classification in an adversarial setting. A *universal adversarial perturbation* (UAP) is a single vector that, when added to an image classifier's inputs, causes the classifier to mislabel a significant proportion of inputs. Such a vector is *universal* in that it disrupts the classifier on many images. Additionally, the perturbation may be very small—for instance, constrained in its L_2 norm—so as to make the adversarial task more difficult. The UAP paper showed that even under a very small norm bound (so that any changes made to an image are almost imperceptible by the human eye), there can still be small adversarial "noise" filters such that, when added to an image, causes that image to be misclassified by an image classifier.

In addition to replicating the DeepFool algorithm from the landmark paper, we hope to utilize the more efficient generation methods proposed by a recent paper (Shafahi et al. 2020b) to generate UAPs.

Our replication project, situated within the image classification subfield of computer vision, asks three related research questions:

1. Can we produce image-agnostic UAPs for DNNs using the DeepFool algorithm proposed by (Moosavi-Dezfooli et al. 2016b) utilizing optimizations from (Shafahi et al. 2020b)?
2. How well do these UAPs fool image classifiers?
3. How can our observations be theoretically justified?
4. How can we apply UAPs to Vision Transformer Models (Dosovitskiy et al. 2020b)?

For (1) and (2), we plan to attack DNNs pre-trained on the ImageNet data (Russakovsky et al. 2015), including the VGG family of networks (Simonyan and Zisserman 2015) and the ILSVRC-winning ResNet family of networks (He et al. 2016). These networks are precisely the ones used in (Moosavi-Dezfooli et al. 2016b). For (3), we plan to offer pen-and-paper proofs (or heuristics) explaining the successes and failures of DeepFool and our replication efforts. Any supporting experimentation would use toy DNNs or the networks mentioned already, so would contribute only marginally to our computational and timeline overhead.

Our extension lies in the application of UAPs to Vision Transformers (ViTs) (4), as we hope to apply perturbations to contemporary ViT architectures (ViT-B, ViT-L) to identify whether UAPs have a similar effect on ViTs. In addition, we identify trends and patterns present in generated perturbations that reveal aspects about ViT architecture and training methods. Due to the novel use of positional encodings and patch-wise granularity used by ViT, we hypothesize that UAPs will utilize these underlying architectures within pre-trained ViT models to improve fooling rates.

Related Work

Research in the domain of adversarial networks has often sought to provide justifications as to their efficacy within image classification models. In a 2016 paper (Tanay and Griffin

2016), the authors proposed a rationalization for the effectiveness of adversarial perturbations on classification models. Whereas previous discussions have proposed that perturbations leverage hidden high-dimensional artifacts learned by models in a sort of "accidental steganography" on part of the classifier, this paper argues that adversarial learning is a result of decision boundaries lying in a higher dimensional space compared to the submanifold of an image dataset. Allowing this boundary to be tilted relative to training data and allowing for adversarial examples to be learned within these subspaces. Within our extension, we hope to explore how this applies to ViT and theoretically justifying the efficacy of our adversarial examples generated for ViT architectures.

There has also been research in the field of detecting UAPs as well, (Metzen et al. 2017) found that a generalizable detector can be effectively trained to detect adversarial perturbations within an input image. This paper makes the observation that a UAP will generate vectors which push an image towards a higher-dimensional decision boundary bordering a known data-manifold within a pretrained model. By generating a detector for UAPs, the authors were able to propose this rationalization of UAP generation. Providing basis for their effectiveness when applied to ViT architecture, as it is reasonable to assume that the DeepFool algorithm will be able to determine similar perturbations which push data over proximal data-manifolds within pretrained ViTs. There may also be a different given the large datasets and large amount of learnable parameters used to train ViT models, we may see that UAPs perform better or worse based on the complex data-manifolds present in these models.

Various defenses against adversarial attacks (both universal and per-image) have been found to be easily broken by newer adversaries. A recent paper (Shafahi et al. 2020a) discusses whether these adversarial attacks are inevitable with the current age of image classifiers. Several methods for defending against these attacks were proposed, such as adding an "unknown" classification, such that perturbations do not cause false classifications in all cases. The paper also proposes that feature squeezing through the application of autoencoders can help to reduce the leverage that adversarial networks have on models. We will explore whether or not these methods would be applicable within ViTs to protect against adversarial attacks.

Background

We following the definitions, metrics, and principles defined below within this replication project:

Datasets: We trained our perturbations on the ImageNet dataset (Russakovsky et al. 2015). Consisting of approximately 30,000 images with 1000 classes.

Metrics: we define the *fooling rate* as the proportion of misclassified images within the dataset with a perturbation applied.

Iterative v.s. SGD the iterative DeepFool algorithm is utilized in the landmark paper (Moosavi-Dezfooli et al. 2016b) designed to find the smallest adversarial perturbation by iteratively calculating the vector which pushes an image over the nearest decision boundary of a classifier. In contrast, the SGD-based DeepFool algorithm incorporates stochastic gradient descent (SGD) to approximate the adversarial perturbation. While the iterative version aims for precision and minimal perturbation, the SGD-based version trades off some precision for computational efficiency.

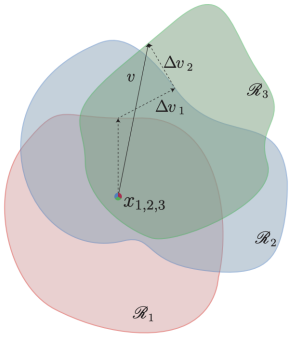


Figure 1: Conceptual representation of UAP optimizing to decision boundaries from (Moosavi-Dezfooli et al. 2016b). The high-dimensional analogue to this in the ViT setting should be the same due to continuity of the patch transformation.

Theoretical Justification for UAPs on ViTs

In the context of applying the UAP concept to Vision Transformers, it is important to note here that there is no reason why we should not except the same concept of decision boundaries between labels to hold in the ViT context. From a surface standpoint (that is, prior to transformation from image patches into vector embeddings for the transformer architecture), we have the same notion of boundaries existing in the patch space between decisions. This is due to the fact that the transformation from patches to transformer input embeddings is continuous (Dosovitskiy et al. 2020b). While the dimensionality of the vector embedding for the vision transformer may be quite high, this does not modify the existing one-to-one correspondence between a "boundary word" in the transformer and a "boundary patch" in the original image.

As a result, one way to think of perturbations per patch in the ViT context is in correspondence to perturbations per pixel in the original DNN setting for a UAP. One heuristic to see whether this holds is to check whether the UAPs obtained in the ViT setting have rough internal boundaries corresponding to patch boundaries, much in the same way that UAPs in the DNN setting have pixel boundaries. In the latter context, patterns across pixels in a DNN UAP may be interpreted to have correspondences with features of the internal network architecture, whereas one should expect the patch delineation in the ViT case to be the most prominent and noticeable feature.

Computational Resources: Due to the relatively manageable computational requirements of generating UAPs, we were able to run training on a MacOS M1 chip over the course of 14 hours for the larger datasets. This computational budget allowed us to replicate the general trends found in the original UAP paper (Moosavi-Dezfooli et al. 2016b)

	GoogLeNet	ResNet-152	VGG16	VGG19
Random	3.4	2.2	5.9	6.6
GoogLeNet	17.81	8.97	21.05	20.47
ResNet-152	14.39	66.59	20.15	19.18
VGG16	14.13	8.20	25.38	21.03
VGG19	15.85	12.03	27.24	29.04

Table 1: Fooling Rate (%) for DNNs
Rows indicate the architecture for which UAPs are generated. Columns indicate the architecture for which fooling rates are reported (as percentages). "Random" is a random vector with ℓ_2-norm 30.

	GoogLeNet	ResNet-152	VGG16	VGG19
None	70.3	82.7	70.6	70.7
GoogLeNet	62.6	81.3	66.8	67.0
ResNet-152	68.2	31.1	62.7	64.9
VGG16	68.9	80.7	60.4	67.8
VGG19	65.9	80.4	65.1	63.6

Table 2: Validation Rates for DNNs
Rows indicate the architecture for which UAPs are generated. Columns indicate the architecture for which top-1 validation rates are reported (as percentages). The "None" row gives unperturbed validation rates.

and run our experimentation with ViTs.

Methodology, Results, and Discussion

In this section, we replicate the original results on UAPs for DNNs (Moosavi-Dezfooli et al. 2016b). This involves utilizing the DeepFool algorithm presented in the original UAP paper and applying it to various models to generate perturbations. We then calculate the fooling rate of these perturbations when applied to various model architectures.

Methodology In a first experiment, we reproduce Table 2 of (Moosavi-Dezfooli et al. 2016b). We use their Algorithm 1 to generate universal adversarial perturbations for four DNNs: GoogLeNet (also called Inception v1), ResNet-152, VGG16, and VGG19. We report the *fooling rate*, or percentage of images that change labels when perturbed by a given UAP, in Table 1. (Moosavi-Dezfooli et al. 2016b) introduced fooling rates as the main way to evaluate UAPs, but we believe this is a misspecification of objectives: more relevant in practice is the effect of a perturbation on overall classification accuracy, which we report in Table 2.

All DNNs used weights pre-trained on the ImageNet1K subset of ImageNet. None were fine-tuned. We sampled training and validation data from the same subset uniformly at random. Due to computational limitations, we generally validated on 10000 images from the validation set (rather than all 50000).

The key parameters to Algorithm 1 are $|X|$, the size of the training set; p, an element of $[1, \infty) \cup \{\infty\}$; and ξ, the maximum ℓ_p-norm of the generated UAP. Algorithm 1 stops when the empirical fooling rate on the training data exceeds a threshold value $1-\delta$ or after a maximum number of epochs M.

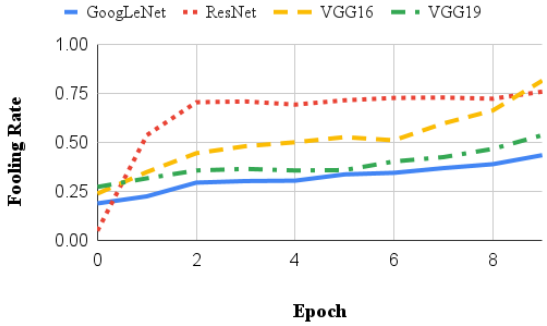


Figure 2: DNN Fooling Rates During Training
Fooling rates are computed over the entire training data.

Due to our limited computational budget, we set $|X| = 500$, considerably lower than the setting $|X| = 10000$ in the original paper. We set $\delta = 0.2$, but found this was never achieved in practice, so that Algorithm 1 always terminated after $M = 10$ epochs. We set $p = 2$ and $\xi = 30$. The average ℓ_2-norm of an image in the validation set is ≈ 450, so $\xi = 30$ ensures the generated perturbations are sufficiently small as to be easily ignored by humans. This setting is approximately commensurate with the original paper, where $\xi = 2000$ and the average ℓ_2-norm is $\approx 5 \times 10^4$.

In a second experiment, we consider the effect of increasing $|X|$. We consider $|X| \in \{500, 5000, 10000\}$ for the GoogLeNet architecture with all other settings as above. The resulting fooling rates comprise Table 3; the resulting UAPs are Figure 3.

Our implementation of Algorithm 1 is based on, but independent from, the code accompanying (Moosavi-Dezfooli et al. 2016b); therefore, it is doubly unlikely any results shared between our works are due to implementation mistakes.

Results and Discussion Our fooling rates reside around 15-25% (Table 1), considerably lower than the 40-90% reported by (Moosavi-Dezfooli et al. 2016b), but nonetheless far beyond the 2-7% achieved by random perturbations of comparable size. Although our fooling rates are lower in absolute terms, there is a moderate correlation ($r^2 = 0.546$) between our values and those in the original work.

We attribute the drop in fooling rates primarily to training on one-twentieth of the original paper's training data. This is borne out by Figure 6 of (Moosavi-Dezfooli et al. 2016b), which reports a fooling rate of $\approx 30\%$ for perturbations trained with $|X| = 500$. It is also evidenced by the drop in fooling rate from training to testing (Figure 2, Table 1), indicating poor generalization from the small training set. Other potential causes include stopping too early—Figure 2 shows that some empirical fooling rates never stabilized—and a relatively small choice for ξ. Nevertheless, we are able to fool a large proportion of images while training on less than one image per object class.

ResNet appears unusually robust against perturbations generated on other architectures, but unusually susceptible

$\lvert X\rvert$	Fooling Rate
500	17.00
5000	40.40
10000	56.13

Table 3: Fooling Rates for Varying Choices of $|X|$
Fooling rates (as percentages) against GoogLeNet for UAPs trained against GoogLeNet with $\xi = 30$ and $p = 2$.

to its own perturbations. We cannot account for this. Otherwise, fooling rates drop only slightly when a UAP trained on one architecture is applied to another, supporting (Moosavi-Dezfooli et al. 2016b) claim that UAPs are "cross-model universal." Validation rates uphold these conclusions, and by this metric, ResNet is only midly more robust against cross-model perturbations than GoogLeNet. It is reassuring that fooling rates correlate inversely with validation rates ($r^2 = 0.784$) and that validation rates drop by 2-10% after applying UAPs. Although fooling rates are arguably the wrong metric for UAPs, they are still a good proxy metric. Following the literature, we report only fooling rates going forward.

Figure 3 shows (i) that our UAPs exhibit the same superficial characteristics as in the paper, namely small black patches and swirls of color, and (ii) that perturbed images are still clearly interpretable by humans. In particular, the earlier theoretical prediction in the Background section that the distinction between patches in the original ViT setup should correspond to real features on the ViT turned out to be true: upon closer inspection, every ViT UAP has abrupt pattern changes at regular square-sized blocks, which correspond to the boundary lines of patches in the ViT architecture.

From Table 3, we see that increasing $|X|$ does indeed increase fooling rate, albeit not to the level of (Moosavi-Dezfooli et al. 2016b). Thus, choosing small values for $|X|$ does not entirely explain our lower fooling rates. Note that due to computational limitations, training on the $|X| = 10000$ image set was stopped after six epochs instead of the usual 10.

In (Moosavi-Dezfooli et al. 2016b), it is posited that perturbed images tend to be classified as certain dominant labels. We observe the same phenomenon: applying the $|X| = 10000$ GoogLeNet perturbation causes approximately 46.5% of the validation set to be classified as marmosets despite all 1000 classes being equally distributed over this set.

We still have minor concerns about replicability. First, a potentially relevant parameter, which we call C, was omitted from the original paper. Given a classifier k, training sample x, and classes $[n] = \{1, \ldots, n\}$, the heart of Algorithm 1 is to use DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016) to compute the minimum perturbation Δx sending x to some decision boundary. That is,

$$\Delta x = \arg\min_z \|z\|_2 \text{ s.t. } k(x+z) \in [n] \setminus \{k(x)\}$$

In every implementation we inspected, the following was computed instead:

$$\Delta x = \arg\min_z \|z\|_2 \text{ s.t. } k(x+z) \in \{\ell_1, \ldots, \ell_C\} \setminus k(x)$$

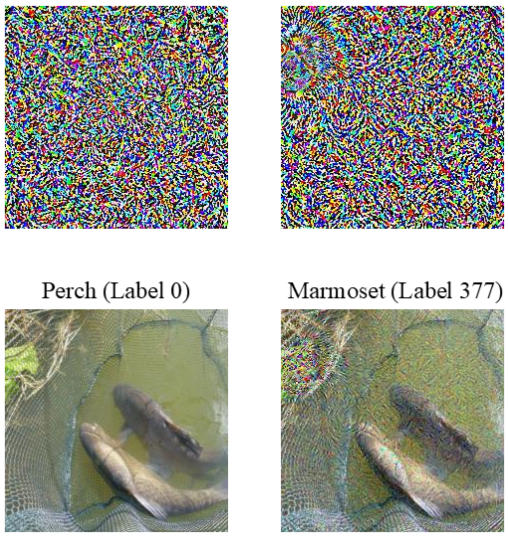


Figure 3: UAPs and Perturbed Images for GoogLeNet
Top Left: UAP trained on GoogLeNet with $|X| = 500$.
Top Right: UAP trained on GoogLeNet with $|X| = 10000$.
Bottom Left: Unperturbed image from validation set.
Bottom Right: Image perturbed by top right UAP.

where $\ell_1, \ldots, \ell_m$ are the C likeliest classes for x according to k (so $k(x) = \ell_1$). As presented in the paper, Algorithm 1 is the special case $C = n$. The typical range for C is 5-10, which results in huge computational savings when n is large. We found no analysis of the effect of C on the efficacy of Algorithm 1, but our own brief investigation, wherein we set $C \in \{1, 5, 10, 20, 100\}$, suggests that Algorithm 1 achieves lower fooling rates when C is large (and in particular, when $C = n$). To produce Tables 1 and 2, we set $C = 5$.

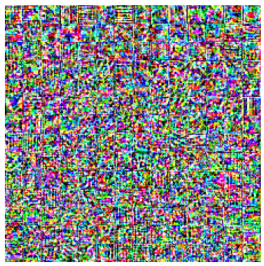
A major ambiguity in the original paper is whether perturbations are applied before or after image preprocessing (image rotation, normalization, blurring, etc.). The accompanying implementation post-applies perturbations, and so do we. However, this goes against the intuition that perturbations come from an adversary with no influence over the classification process. Also, Figure 3 in (Moosavi-Dezfooli et al. 2016b) shows perturbed images that are not normalized, implying that perturbations are pre-applied. Applying perturbations before rescaling and normalization would explain why we report an average ℓ_2-norm of 450 compared to the original 5×10^4 despite sampling from the same dataset. This should be clarified by the authors.

Paper Extension: Generating UAPs for ViTs

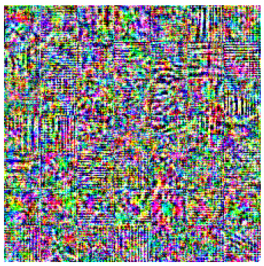
We will now analyze the application of UAP-generated perturbations on multiple instances of vision transformer models, analyzing the comparative fooling rates across a variety of factors including training set size (number of pure images) as well as differing pretrained ViT models used in the process of creating the perturbed images.

Model	Layers	Hidden Size D	Heads	Params
ViT-Base	12	768	12	86M
ViT-Large	24	1024	16	307M
ViT-Huge	32	1280	16	632M

Table 4: Details of Vision Transformer model variants



(a) Perturbation trained on ViT-B/16



(b) Perturbation trained on ViT-B/32

Figure 4: Generated Perturbations for ViT Architectures

Comparing Pretrained ViT Models The first experiment we wanted to consider was the differentiation between perturbed image performance across a variety of models. Specifically, Vision Transformer pretrained models come with specified parameters following the format *ViT-{B/L/H}/{Patch Size}*, where ViT-B/16 represents a vision transformer pretrained on the base configuration with 16x16 patch size. A detailed breakdown of the exact parameters of each model configuration can be seen in Table 4. We wanted to analyze if the differing architectures between these models would result in any visual differences in the resulting perturbed images, or differentiation between fooling rates. As such, we trained the perturbed images across the following pretrained models: ViT-B/16, ViT-B/32, and ViT-L/32. We fixed the training set to use our default ImageNet subset of 32,500 images on 1000 classes defined in the Dataset section. For this experiment, we utilized a validation subset of 10,000 images taken from the ImageNet validation dataset in order to reduce validation time due to computational limitations. Figure 4 shows the resulting perturbations

Figure 5: Example of correctly classified (left) and Perturbed (right) images

	ViT-B/16	ViT-B/32	ViT-L/32
Fooling Rate	99.84%	99.82%	99.84%

Table 5: Vision Transformer Fooling Rates
Rates are validated on a 5,000 image subset of ImageNet.

Training Set	Fooling Rate
1k images	98.78%
6.5k images	99.21%
13k images	99.22%
32k images	99.82%

Table 6: ViT-B/32 Model performance on various dataset sizes

corresponding to ViT models trained on different parameters. Observe the distinct image patch sizes found in each perturbation where training patch size clearly translates to the produced perturbated image. For instance, ViT-B/16 and ViT-L/16, with 16x16 patch size, have notably smaller visible image patches in comparison to the ViT-B/32 and ViT-L/32 models which utilize 32x32 pixel patch sizes. In addition, after applying these Perturbations upon the validation images, we can observe in Figure 5 that the resulting Perturbed Images are visually identical to the original images while resulting in each corresponding model to misclassify the image label. In other words, the applied perturbation filter is essentially invisible to the naked eye once applied. It is also important to note the resulting fooling rate of each of the pretrained ViT models. Observe Table 5, where we can see that for the validation image set of 10,000 images, we achieved incredibly high fooling rates of around 99% for each perturbation. Compared to our UAP findings, these resulting values are very surprising are are likely indicative of the effectiveness of image perturbations of ViT models.

Effect of Training Dataset Size Next, we wanted to experiment with different training dataset sizes in order to evaluate the difficulty of training effective Universal Perturbations. For this, we fixed the model configuration to be the pretrained ViT-B/32, as this was the fastest model to train and test on out of the aforementioned models we tested on. We then created various partitions of the ImageNet dataset each with a number of images scaling from 1,000 to the full 32,500 from our dataset. Table 6 displays the results for our testing experiments. There is indeed a positive correlation where larger number of images result in better fooling rates of our model. However, what stood out to us was that even smaller subsets of only 1,000 images could achieve extremely effective results of over 98% on the validation set. Compared to the original findings (Moosavi-Dezfooli et al. 2016b) where low training dataset size lead to relatively poor fooling rates, the Vision Transformer architecture seemingly allows for extremely effective perturbations. Although more comprehensive testing on multiple models and varying parameters is needed to fully arrive at a conclusion, it appears that these ViT models can generate extremely effective perturbations on small samples of training data.

Model	Fooling Rate
ViT-B/16	99.78%
ViT-B/32	**99.82%**
AlexNet	98.20%
VGG-16	98.51%
ResNet-152	26.24%
GoogleNet	99.02%

Table 7: Comparing ViT-B/32 Perturbation performance across a variety of models

Cross-Model Universality of ViT Perturbations Finally, we evaluated the results of ViT perturbations across a variety of different models. For this, we fixed the UAP perturbation computed upon the ViT-B/32 model with 32,500 training images. We wanted to compare both the effects of different patch sizes, as well as completely different models. To do so, we first tested the perturbation upon the ViT-B/16 model, with 16x16 patch sizes in comparison to the 32x32 patch size of the training model. In this case, the resulting fooling rate was not shown to very much, however the differing patch size model performed slightly worse in comparison to the native model. We also wanted to analyze the performance of this perturbation on a variety of models. While most models performed relatively well, within 1% fooling rate of the original ViT-B/16 rate, the ResNet-152 model stood out immensely, with only a 26.24% fooling rate. Compared to the cross-model findings from (Moosavi-Dezfooli et al. 2016b), GoogleNet and VGG-16 performed significantly better relative to the native models, while ResNet had a notable decrease in performance. All results are listed in Table 7.

Given additional resources, comprehensively evaluation of the effectiveness of each computed perturbation upon each model would have resulted in more conclusive results, however the goal of this replication was to prove the ability to adapt universal perturbations to ViT algorithm and further research can be performed to uncover additional findings.

Extension Results: After evaluating the fooling rate of UAPs applied to various ViT-B/32 we found that across all training sets the fooling rate of the DeepFool algorithm was extremely high across all conditions (see Table 5). This may be a result of vulnerabilities within the vision transformer architecture, which make them especially susceptible to perturbations.

Extension Discussion: Our results show a definite trend that vision transformer architectures remain significantly more susceptible to UAPs compared to DNN architectures. Thereby indicating an underlying vulnerability in the ViT architecture that allows a UAP to consistently perturb classifications. This may lie in the patch-based representations learned by ViT models, which can be clearly seen in the UAPs generated in Figure 4. Given the importance of positional encodings in ViT architectures for patch comprehension (Dosovitskiy et al. 2020b), and the fact that ViT models utilize a fixed sinusoidal position embedding function, it is possible that the DeepFool algorithm is able to learn these embeddings to induce misclassifications.

In addition, the number of learnable parameters for ViT architectures is significantly larger than conventional DNN models (Dosovitskiy et al. 2020b). This could result in a higher fooling potential given the high dimensionality of the decision boudnaries learned by ViT with a large number of parameters (Tanay and Griffin 2016), thereby giving potential for the high fooling rates seen in Table 5.

However, there were many caveats that lied within our research due to computational limits and short timelines. One area in which we could test the extension further is expansion upon differing perturbation norms. All of our perturbations were trained upon a norm $\xi = 10$, while changing the norm of perturbation is shown to significantly affect the resulting fooling rates (Moosavi-Dezfooli et al. 2016b). The fixed perturbation norm we utilized may have resulted in the high fooling rates we achieved on ViT architecture, so further analysis of different norms would certainly result in more comprehensive results. In essence, these roadblocks heavily limited the range of our findings and further computational resources would have bolstered the extent of our results.

Conclusions

Throughout our replication, we found that the DeepFool algorithm was able to generate perturbations with a significant fooling rate on various DNN architectures for image classification. In addition, we showed that these perturbations are also universal, and have transferrable fooling capabilities that extend beyond their original models. In our extension we explored the effect of UAPs on pre-trained vision transformer models. These tests showed that ViT models are especially susceptible to UAP attacks, with a high fooling rate. Future work could involve empirically determining the representations that UAPs learn when fooling ViT models, as the unique positional encodings and patches used in the architecture could have unique contributions to the fooling potential of DeepFool on ViT architectures.

Societal Impact

By understanding the vulnerabilities that vision models have due to Universal Adversarial Perturbations, we can better defend against attacks for image classifications. The DeepFool algorithm provides proof that malicious attackers have the opportunity to apply generalizable perturbations to consistently fool models, opening up the field for research in UAP detection and defense methods. We also explored the effect of UAPs on Vision Transformer models, which are currently state-of-the-art and are widely used by generative AI services and various computer vision applications. Furthermore, Vision Transformers currently have widespread usage in critical industries such as digital health applications (Al-Hammuri et al. 2023), as well as autonomous driving (Lai-Dang 2024). As exposure to UAPs may lead to disastrous results in these use cases, it is necessary that further research is done to secure these models from misclassification due to perturbed images. By showing that ViTs are susceptible to these adversarial attacks, we can better understand the vulnerabilities in these widely used systems.

References

Al-Hammuri, K.; Gebali, F.; Kanan, A.; and Chelvan, I. T. 2023. Vision transformer architecture and applications in digital health: a tutorial and survey. *Vis. Comput. Ind. Biomed. Art*, 6(1): 14.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020a. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020b. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lai-Dang, Q.-V. 2024. A Survey of Vision Transformers in Autonomous Driving: Current Trends and Future Directions. arXiv:2403.07542.

Metzen, J. H.; Genewein, T.; Fischer, V.; and Bischoff, B. 2017. On Detecting Adversarial Perturbations. arXiv:1702.04267.

Moosavi-Dezfooli, S.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2016a. Universal adversarial perturbations. *CoRR*, abs/1610.08401.

Moosavi-Dezfooli, S.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2016b. Universal adversarial perturbations. *CoRR*, abs/1610.08401.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. DeepFool: a simple and accurate method to fool deep neural networks. arXiv:1511.04599.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3): 211–252.

Shafahi, A.; Huang, W. R.; Studer, C.; Feizi, S.; and Goldstein, T. 2020a. Are adversarial examples inevitable? arXiv:1809.02104.

Shafahi, A.; Najibi, M.; Xu, Z.; Dickerson, J.; Davis, L. S.; and Goldstein, T. 2020b. Universal Adversarial Training. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 5636–5643.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tanay, T.; and Griffin, L. 2016. A Boundary Tilting Persepective on the Phenomenon of Adversarial Examples. arXiv:1608.07690.

Individual Contributions

Andy Chen

Conducted research on previous papers related to UAPs. Wrote the Abstract and Extension Results sections. Assisted in formatting and grammar checking throughout the paper.

Luke Miga

Code and data for replicating (Moosavi-Dezfooli et al. 2016b). Associated figures. Portions of "Methodology, Results, and Discussion" section.

Luke Yang

Adapted UAP model from only using toy dataset to using imagenet data. Trained ViT-B/16 and ViT-B/32 models to generate appropriate UAPs. Performed validation testing on ViT performance locally. Wrote portions of ViT replication portion in final paper.

Nathan Yap

Wrote Introduction, Related Work, Conclusion, Societal Impacts, and Extension Discussion sections. Created the initial script to generate UAPs for the toy dataset. Conducted research on related works and added additional citations.

Appendix

Additional examples of Perturbed Images on ViT-B/16 model and perturbation

black and gold garden spider

cradle

water bottle

pillow

Maltese dog

bib

daisy

African chameleon

broom

cradle