# Exploring the Impact of Model Architecture on Adversarial Transferability in Facial Recognition

**Xander Coomes**
Department of Computer Science
University of California, Santa Barbara
Goleta, CA 93106
xsc@ucsb.edu

**Derek Kirschbaum**
Department of Computer Science
University of California, Santa Barbara
Goleta, CA 93106
dgkirschbaum@ucsb.edu

**Agam Makkar**
Department of Computer Science
University of California, Santa Barbara
Goleta, CA 93106
agam@ucsb.edu

## Abstract

**Given the ubiquity of facial recognition software used in our day-to-day lives, it seems as though every major corporation knows what we look like. In this paper, we explore different types of adversarial attacks that could fool current facial recognition models, and examine the transferability of attacks between different types of machine learning architectures. Specifically, we attack 5 types of machine learning models, and test the classification performance of 6 different models when given the adversarially perturbed images as input. We find that models built upon similar underlying architectures are more likely to fool one another than models with differing architectures. Additionally, attacks generated by simpler architectures exhibit greater transferability than attacks generated by more complex models. We also provide insight into the practical applications of our findings and discuss how we could use these results in future work to design methods to fool powerful facial recognition systems.**

## 1 Introduction

Adversarial perturbations are imperceptible alterations to images that can cause machine learning models to misclassify their input. They can be used in many contexts, such as protecting privacy by preventing companies like Apple from recognizing faces in photos, to adding digital watermarks to images, or, in a more malicious scenario, to cause self-driving cars to misidentify road signs. In prior work, Papernot et al. [2016] generated adversarial attacks by training a source model such that its decision boundary mimicked that of the target model. However, given a highly accurate target model such as Apple's facial recognition system, and given our limited computational resources, this approach becomes infeasible.

In our paper, we explore transferability of adversarial perturbations in contexts where it may be difficult to accurately mimic the decision boundary of a target model. Transfer attacks involve training a source model to generate a perturbation, and then applying the perturbation to a target model. Transfer attacks are often implemented where access to the target model is limited, which is common given that many models such as Apple's facial recognition, ChatGPT, or IBM Watson are not open source software.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Our hypothesis was that the attacks generated by larger capacity embeddings models would have a high transferability success rate to small capacity models, even with relatively small perturbations. In contrast, we thought that perturbations generated by smaller capacity models would be ineffective at transferring to the larger capacity models. These beliefs were based on the idea that larger embeddings models tend to gather deeper, richer information about a face, potentially generating more effective perturbations. Further, we thought the embeddings models would be more robust, and less susceptible to perturbations generated by other models. In terms of architectural differences between source and target, we also believed that models with similar architectures would transfer better to one another.

## 2 Related Work

In Goodfellow et al. [2015], the Fast Gradient Sign Method attack (FGSM), the simplest method of generating perturbations, was first introduced. This was one of the first major works detailing key insights about adversarial examples. One fascinating property was that while these perturbations are imperceptible to the human eye, they are still able to fool machine learning models. Because an image may contain thousands of pixels, changing many of them, each by a very small margin, can result in a misclassification.

This, among other papers, sparked research into developing more effective perturbations. Xi et al. [2024] details advancements in perturbation development, including iterative methods such as Iterative FGSM, among others. Other attack methods such as Auto PGD by Croce and Hein [2020], and the P-FGVM attack by Chatzikyriakidis et al. [2019] were also developed to increase effectiveness against increasingly robust models.

Our work specifically focused on transfer attacks, where perturbations are generated using one model (the source) and tested on another (the target). Papernot et al. [2016] demonstrates that when the same model architecture is trained on different data (disjoint subsets of the entire training set), transferability effectiveness is relatively high. This is called intra-technique transferability. Cross-technique transferability, the capacity to fool a target model while generating a perturbation with a source model of a different architecture, was also tested. They demonstrated the success of cross-technique transferability, and also proposed ways to improve black box transferability when given oracle access to the target model. One such method (Jacobian Based Dataset Augmentation) involves fixing the source model's architecture, and then refining the source model's training set in order for it to achieve a very similar decision boundary and loss landscape to the target model.

While much of the literature on adversarial perturbations has focused on image classification in general, less research has been performed with regards to facial recognition and classification. Deb et al. [2019] introduces common models such as ArcFace and FaceNet, and common datasets such as Casia, and VGG, that are frequently used when studying facial recognition. We make use of these models and datasets throughout our experiments, while additionally exploring both intra-technique and cross-technique transferability in an effort to apply techniques from general image classification to the field of facial recognition.

## 3 Background

### 3.1 Definitions

*Perturbation*: Intentionally crafted noise that is added to an image.

*Adversarial attack*: A method that generates and adds perturbations to an image in an attempt to cause a machine learning model to misclassify the image.

*White-box attack*: An adversarial attack that is performed on a model that the attacker has complete knowledge of (i.e. hyperparameters, architecture, training methods)

*Black-box attack*: An adversarial attack that is performed on a model that the attacker has no internal knowledge of (i.e. the attacker can only feed inputs and read outputs from the model)

*Epsilon*: The maximum change per pixel allowed during a given attack.

*Embeddings Model*: An embeddings model is a deep neural network that has been pre-trained on a large database of facial images and labels to generate an embeddings vector of the features detected

by the model. Different images of the same person will generate similar embedding vectors that are near each other in the model's embedding space, but images of different people will generate vectors that are quite different in the embedding space.

*Cosine Similarity Score*: A numerical value between -1 and 1 that describes the similarity between two embedding vectors. A score of 1 indicates the vectors are identical while 0 indicates the vectors are orthogonal and -1 indicates the vectors are pointing in opposite directions. For our project, we considered any score above 0.5 to indicate that the two embedding vectors represented different images of the same person.

$$cossim(x, y) = \frac{x \cdot y}{|x||y|} \tag{1}$$

### 3.2 Attack Definitions

*Fast Gradient Sign Method Attack*[Goodfellow et al., 2015]: The FGSM method takes the gradient of the model's loss function $L$ with respect to the input image $x$. $\theta$ represents the model's parameters, and y is the ground truth label.

$$X_{\text{adv}} = x + \epsilon \operatorname{sign}\big(\nabla_x L(\theta, x, y)\big) \tag{2}$$

*Projected Gradient Descent Attack*[Madry et al., 2019]: PGD is an iterative attack that applies the FGSM attack repeatedly while constraining the total perturbation to be within a predefined range. At each iteration, the attack scales the gradient by $\alpha$, the step size, and then projects the perturbation back to be within an $\epsilon$-ball of the original image $\Pi$. The PGD attack starts from a randomly perturbed point from within the epsilon ball of the original image.

$$x'_{t+1} = \Pi_\epsilon\Big(x'_t + \alpha \cdot \operatorname{sign}\big(\nabla_x L(\Theta, x'_t, y)\big)\Big) \tag{3}$$

*Universal Attack*[Moosavi-Dezfooli et al., 2016] :The universal attack generates a single perturbation vector that can be applied to all inputs. It first initializes a perturbation vector, $v = \overrightarrow{0}$, and then loops through all inputs that are incorrectly classified by the model. At each iteration it updates its perturbation vector adding the gradient of the loss function multiplied by alpha(the step size), and then projecting the perturbation back to be within an e-ball of the original perturbation vector, $v$.

$$v_{t+1} = \Pi_\epsilon\Big(v_t + \alpha \operatorname{sign}\big(\nabla_x L(\Theta,\, x + v_t,\, y)\big)\Big) \quad \text{if } \big(x + v_t\big) = y. \tag{4}$$

## 4 Methodology

### 4.1 Models

1. **Linear:** The linear model uses four fully connected layers. Since we are performing a classification task, we use a cross entropy loss function to train the model.
2. **Convolutional Neural Network (SimpleCNN)**: We designed a CNN with two convolutional layers and two pooling layers, followed by four fully connected layers. The CNN also uses cross entropy loss because we are performing a classification task.
3. **InceptionResNet_v1**: This is a pre-trained embeddings model built using the Inception-ResNet_v1 architecture. When fed an input image, this model generates a feature vector mapping the image to a 512 dimension feature space. We utilized two different models with this architecture, one trained on the VGGFace2 dataset(3.3M faces and over 9000 identities, and the other trained on the Casia-WebFace dataset (453,453 images and 10,575 identities) [Sandberg, 2016].
4. **ArcFace (buffalo_l)**: ArcFace is built from the InsightFace python library and has one default model, buffalo_l, which we used throughout our project. It also generates a 512 dimension feature vector and was trained on a variety of datasets, including MS1M, VGG2, and Casia-WebFace [Guo and Deng, 2018].
5. **Vision Transformer (ViT)**: We used Google's "vit-base-patch16-224-in21k" model that was pre-trained on ImageNet-21k (14M images, 21,843 classes). It processes input in 16x16 patches, and outputs a 768 dimension feature vector [Hugging Face, 2025].

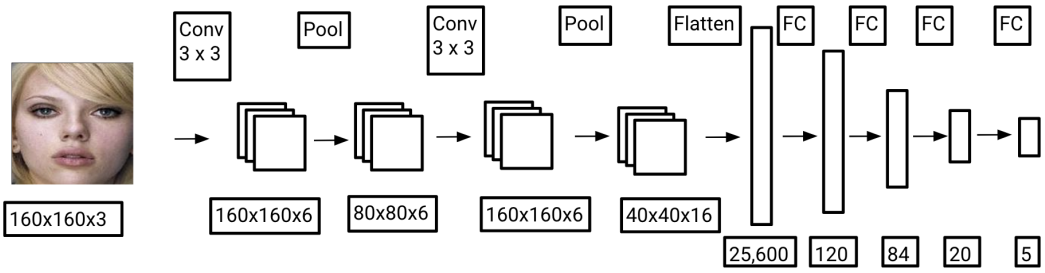


Figure 1: SimpleCNN Architecture

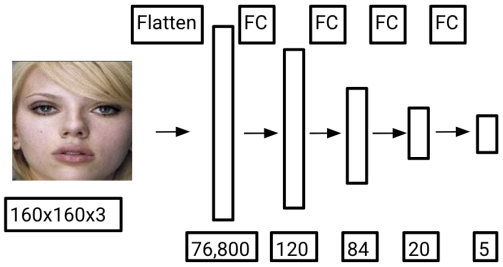


Figure 2: Linear Architecture

### 4.2 Dataset

Throughout this project we trained and tested our model on a collection of 1,913 images featuring five different celebrities: Angelina Jolie, Brad Pitt, Megan Fox, Scarlett Johansson, and Tom Hanks. From public databases on Kaggle, as well as images available online, we collected about 380 images of each celebrity. We then used MTCNN, a widely-used face detection framework, to detect and extract the faces in our photos. This allowed us to crop the images to tightly bound and align each face in the center of the image to ensure consistent inputs throughout all of our training and testing. We normalized the RGB images to the range $[0, 1]$and cropped to a dimension of $3 \times 160 \times 160$. We then split the entire dataset into training, validation, and testing sets using a $70/15/15$ split, respectively.

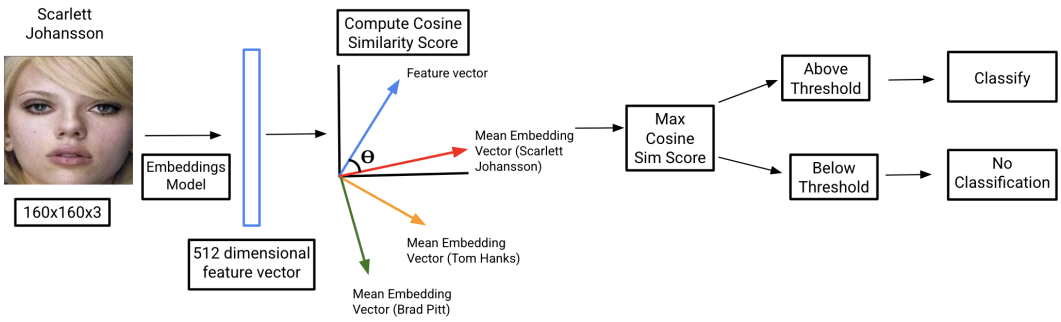


Figure 3: Embedding workflow

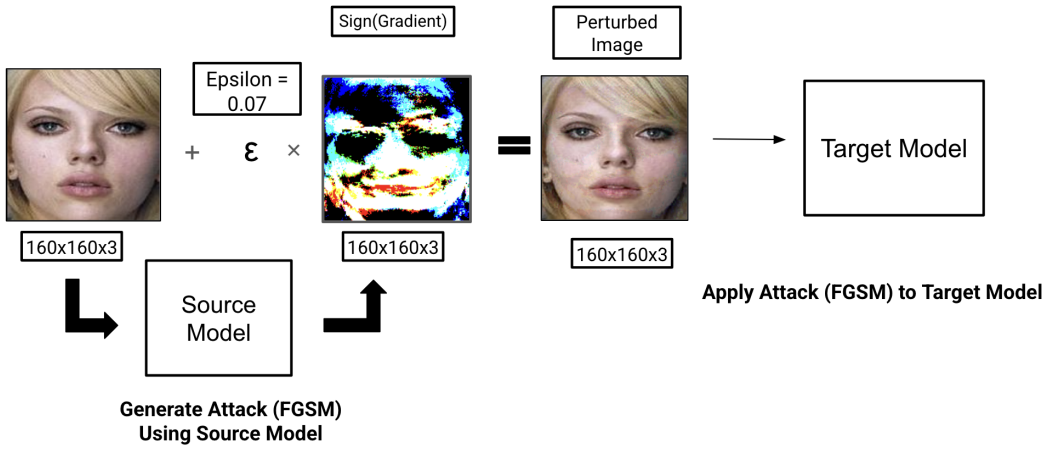


Figure 4: Perturbation workflow (FGSM attack)

Table 1: Model Capacity

| Model | Model Type | Parameters |
|---|---|---|
| SimpleCNN | CNN | $\sim 3M$ |
| Linear | Fully Connected DNN | $\sim 9M$ |
| InceptionResNet_v1 | CNN | $\sim 54M$ |
| ArcFace (buffalo_l) | CNN | $\sim 65M$ |
| Vision Transformer (ViT) | Vision Transformer | $\sim 86M$ |

### 4.3 Perturbation Workflow

## 5 Experiments

### 5.1 Training Procedure

Throughout our testing we relied upon 6 machine learning models which we defined in Section 4.1: Linear, SimpleCNN, InceptionResNet_v1(VGG), InceptionResNet_v1(Casia), ArcFace, and Vision Transformer.

For the Linear and SimpleCNN models, we used the 1,339 images in our training sets to train the models. We used an epoch of 5, batch size of 128, learning rate of 0.001, weight decay of 0.001, and a dropout of 0 for both the Linear and CNN models. Our optimizer was Adam, and our criterion was cross entropy loss. To classify images using these Linear and SimpleCNN models we take the softmax of the model's final layer, and check if the model's confidence in a given class exceeds a chosen threshold (50%).

The embeddings models were pre-trained, so we separated our training data into the five celebrity classes, generated feature vectors for all of the images in each class, and then computed a mean embedding vector for each of the five classes. To classify images using these embeddings models, we required the cosine similarity score between the feature vector and the mean embedding vector of a given class to be greater than or equal to a chosen threshold determined empirically (0.5) as seen in Figure 3.

### 5.2 Testing Perturbations

To compute baseline accuracies, we fed the 288 images from our training set into each model and computed the accuracy based on the number of correct predictions divided by the total size of the training set.

Then, for each of the 6 models, we performed three types of white-box attacks: FGSM, PGD, and the Universal attack. We passed in the 288 images in our testing dataset to each model, then performed

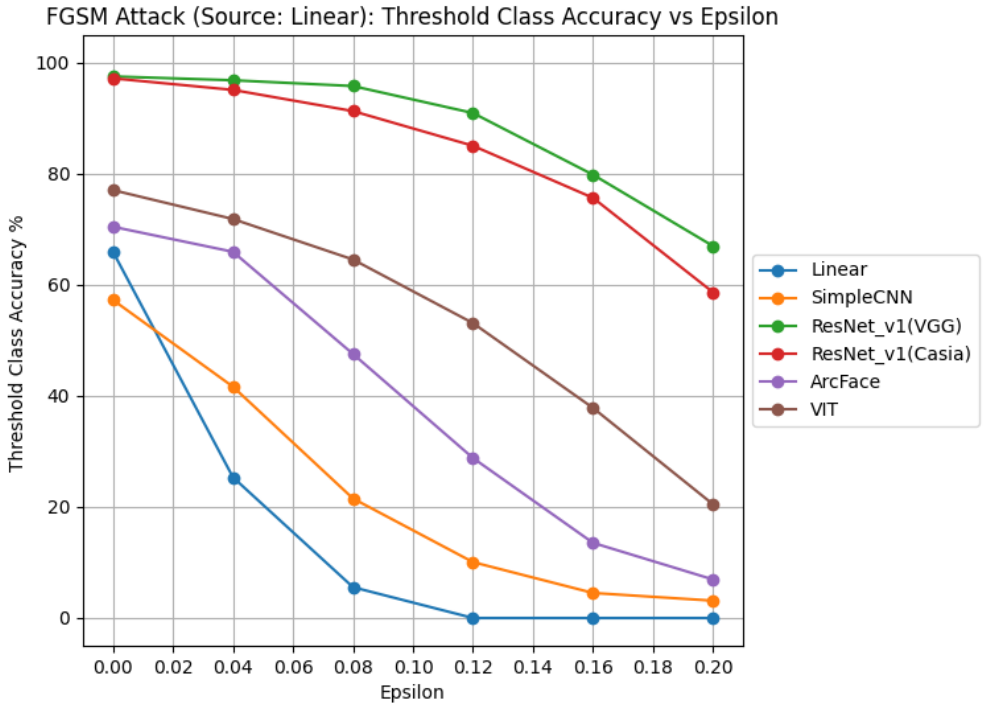


Figure 5: FGSM Attack (Source: Linear) Threshold Accuracy vs Epsilon

each type of attack to generate 288 perturbed images as shown in Figure 4. We then fed these perturbed images into each of the 6 models and computed new accuracies to test if the perturbations generated by a given model caused decreases in accuracy when transferred to the other models.

We tested epsilon values ranging from 0.0 to 0.20, increasing by a step size of 0.04 to scale the size of the perturbations. To ensure that decreases in model accuracies were truly a result of the specific attacks, we also applied uniformly random noise scaled by the value of epsilon to all of our test images, to see how the models performed on images with specific perturbation patterns as opposed to images with uniform noise.

## 6 Results and Analysis

### 6.1 Results

| Source / Target | Linear | SimpleCNN | ResNet(VGG) | ResNet(Casia) | ViT | ArcFace |
|---|---|---|---|---|---|---|
| Linear | -100.0% | -82.6% | -6.2% | -11.4% | -31.9% | -58.8% |
| SimpleCNN | -53.2% | -92.2% | -2.3% | -4.9% | -26.1% | -26.1% |
| ResNet(VGG) | -0.6% | -2.4% | -100.0% | -96.4% | -59.5% | -86.6% |
| ResNet(Casia) | -1.1% | -1.2% | -91.4% | -100.0% | -51.0% | -79.7% |
| ViT | 0.0% | -3.0% | -6.6% | -10.3% | -99.1% | -22.6% |
| Noise | -0.2% | 1.1% | 0.0% | -1.4% | -9.7% | -6.4% |

Table 2: % Decrease in Accuracy when transferring adversarial perturbations from each source model (rows) to each target model (columns) for FGSM attack with $\epsilon = 0.12$.

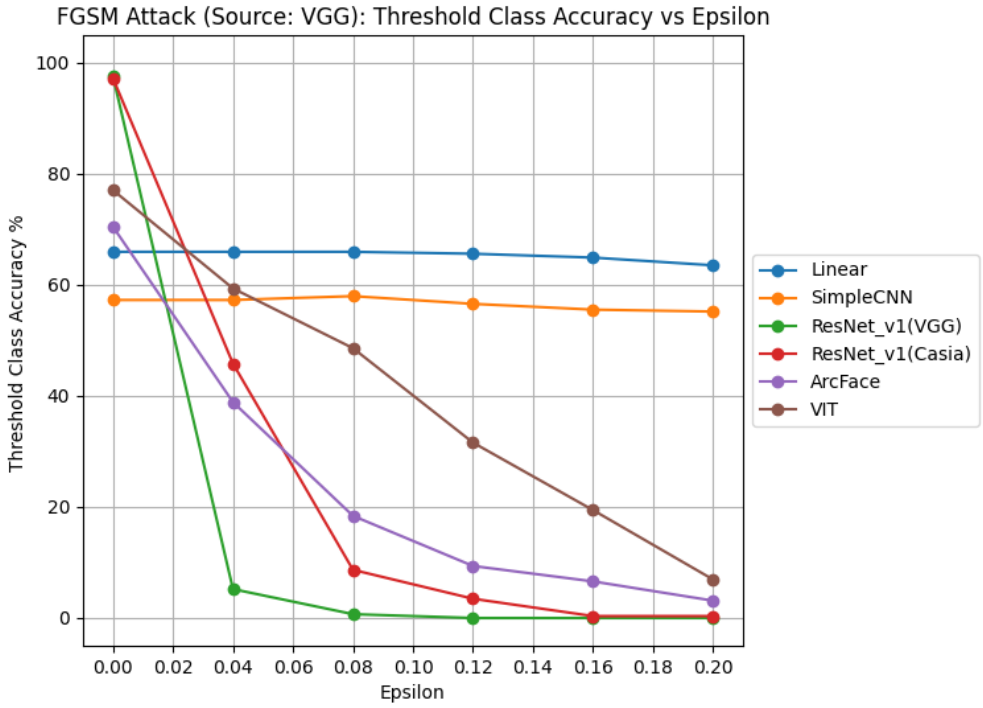


Figure 6: FGSM Attack (Source: VGG) Threshold Accuracy vs Epsilon

### 6.2 Analysis

Throughout our testing, we observed two key trends:

#### 6.2.1 Architectural Similarity Results in Better Transferability:

As demonstrated in Hwang et al. [2024] and Alvarez et al. [2021], we observed that adversarial attacks generated by similar model architectures transfer better to one another. The two ResNet_v1 models that we tested (which both consisted of the same DNN architecture, but were trained on two independent datasets) exhibited the greatest mutual attack transferability. FGSM perturbations generated by ResNet_v1(VGG) caused a 96.4% decrease in the accuracy of ResNet_v1(Casia), and perturbations from ResNet_v1(Casia) caused a 91.4% decrease in accuracy in ResNet_v1(VGG) as seen in Table 2. This trend also holds for the Linear and SimpleCNN models. While their architectures differ slightly, they both have 4 fully connected layers and consist of less than 10M total parameters as seen in Table 1, making them relatively shallow and simple compared to the other models we tested. Linear perturbations led to an 82.6% decrease in the accuracy of the SimpleCNN, and SimpleCNN perturbations led to a 53.2% decrease in the accuracy of the Linear model. Both of these values are nearly double the percent decreases caused by perturbations from any of the other models, which further emphasizes the enhanced transferability between similar architectures.

Because models with similar architectures are likely to function similarly and share a similar decision boundary, it follows that they are susceptible to similar types of perturbations. This is evident in a visual comparison of perturbations generated by the various models as seen in Figure 7. Visually, the perturbations from the Linear and SimpleCNN models appear similar, while the perturbations from the two ResNet_v1 models also share similar features.

With the goal of improving transferability in mind, we can conclude that mimicking the target model's architecture will likely result in increased effectiveness of black box attacks. As architectural similarity increases, the margin separating black box and white box attacks shrinks, leading to strong results when generating and applying adversarial perturbations.

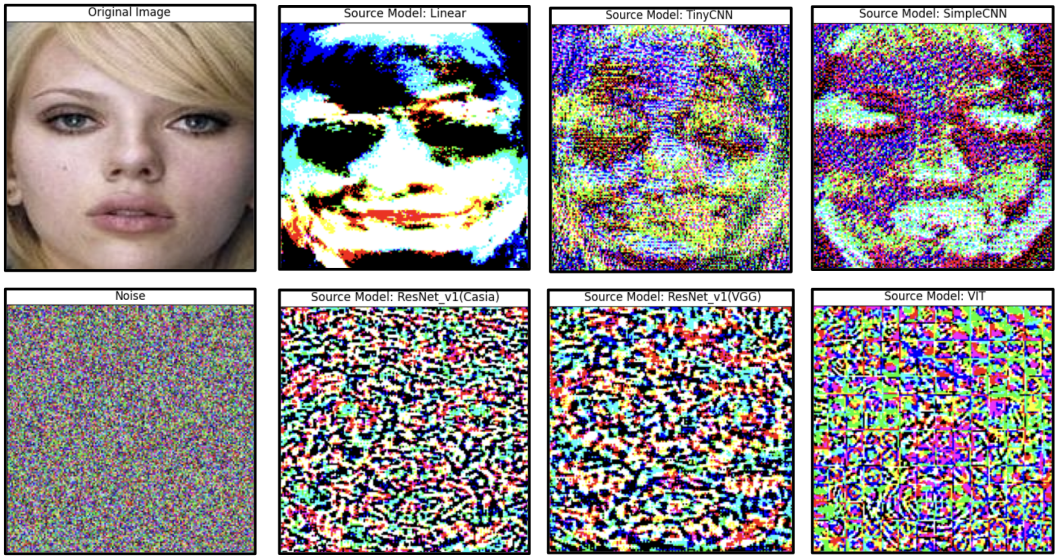


Figure 7: Perturbations visualized

### 6.2.2 Simplicity Results in Better Transferability

We found that across all three of the attacks we tested (FGSM, PGD, Universal), perturbations generated by simpler models transferred more effectively than perturbations generated by models of greater complexity. All 3 types of attacks resulted in very similar results, so for simplicity we choose to only discuss FGSM here, and we discuss PGD and Universal results in the Appendix (See Section A.2).

As shown in 5, an FGSM attack generated by the Linear model led to the following percent decreases in model accuracy: Linear: -100.0%, SimpleCNN: -82.6%, ResNet_v1(VGG): -6.2%, ResNet_v1(Casia): -11.4%, ArcFace: -58.8%, ViT: -31.9%. The SimpleCNN also results in similar decreases in accuracy for all 6 of the models we tested. These results demonstrate a key finding: attacks generated by the simpler models transfer well to other equally or more complex models.

In contrast, attacks generated from models with more complex architectures did not transfer well to simpler models. For an FGSM attack generated from ResNet_v1(VGG), we observed ResNet_v1(VGG), ResNet_v1(Casia), ArcFace, and the ViT all decreased by $\geq 59.5\%$, while the Linear and SimpleCNN models only decreased marginally, by $\leq 2.4\%$. Furthermore, for an attack generated by the ViT model, we observed smaller decreases: all models other than the ViT only showed decreases of $\leq 22.6\%$ as seen in 6. Here, we notice that while perturbations generated from our models of intermediate complexity succeed in transferring to similar (ResNet_v1(VGG), ResNet_v1(Casia), ArcFace) and more complex (ViT) models, they failed to have a significant effect on the simpler Linear and SimpleCNN models. Similarly, perturbations generated by our most complex model, the ViT, had little effect on the simple models, and only caused small declines in the accuracies of the intermediate complexity models.

One reason for these results (specifically when observing gradient based attacks) is that simpler models have more basic loss landscapes. As seen in Figure 7, due to the simple loss landscape of the Linear model, almost all of the pixels in the eye region share the same gradient, which is in the "black" direction. This effectively serves to hide the "eye" feature from the model. In contrast, complex models, which detect more complex features, have more complicated loss landscapes. This is also apparent in Figure 7, where the gradients generated by the ResNet_v1 and ViT models appear more similar to random noise, and human-recognizable features like the eyes, nose, and mouth are no longer clearly defined.

While the trend in machine learning has been to "go deeper", our evidence suggests that simpler models may be more effective at generating generalizable adversarial perturbations.

### 6.3 Sources of Error/Limitations

#### 6.3.1 Dataset Differences

In our experiments, we trained our simple CNN on a dataset of 1,913 images with 5 identities from our custom celebrity dataset. However, the pretrained models we used (such as ResNet_v1 and ArcFace) were originally trained on 10,000 or more images, and we then created a simple cosine similarity classifier on top of the base embeddings model. Thus, the differences in transfer accuracy that we attributed to model architecture could also be a partial byproduct of the original dataset differences.

#### 6.3.2 Metrics Used

For the embeddings models, we classified images using a cosine similarity threshold of 0.5, while for the Linear model and SimpleCNN, we decided to require the model to be over 50% confident in its prediction label. Since there is no direct equivalence between cosine similarity and prediction confidence, these metrics are not equally rigorous and may have induced some experimental error into our results.

## 7 Discussion and Future Work

In our experiments, we limited ourselves to Linear Models, CNNs, and DNNs (the embeddings models). In the future, it seems valuable to explore different architectures and employ ensemble methods to broaden the scope of our findings. We also intend to experiment with different types of attacks, including the Carlini and Wagner Attack by Carlini and Wagner [2017], MI-FGSM by Dong et al. [2017] , and DeepFool by Moosavi-Dezfooli et al. [2015], to see if varying attack methods would improve transferability. With further refinement to our techniques, more data, and additional time, we could better observe how perturbed images impact consumer technology. For instance, would Apple Photos be able to accurately classify a perturbed image of Brad Pitt? Currently, we've achieved misidentification using an FGSM attack with $\epsilon = 0.20$, but we believe there are better methods that could produce more imperceptible perturbations. Based on our experimental findings, we would create a simpler model with architectural features similar to that of Apple's model and generate perturbations using an FGSM attack. In accordance with our results (simpler models transfer better to more complex models, and similar architectures transfer better to each other) this would have the best chance of fooling the Apple Photos classification model. With more testing, we could refine and verify this hypothesis.

## 8 Broader Impact

Machine learning models are increasingly becoming integrated into our day-to-day lives. Image classification has numerous applications including photo organization, facial identification security systems, self-driving cars, and medical imaging. However, these models are still susceptible to adversarial inputs, which could lead to harmful consequences. A self-driving car misreading a stop sign for a speed limit sign or a cancerous tumor being misdiagnosed as benign, could be disastrous. Thus, testing against such examples is crucial to creating production-ready models that can be used by the general public. The broader impact of our work lies in understanding how to generate better and more transferable perturbations, which can then inform the development of better adversarial training methods. Moreover, improving the robustness of machine learning models ensures that it is much more difficult for adversaries to exploit weaknesses. For instance, adding perturbations to social media posts to bypass content moderation filters could rapidly spread misinformation. By researching and aiming to improve the field of adversarial security, we can aid in eliminating these frightening possibilities and build safer, more reliable models.

## 9 Collaboration Statement

**Derek Kirschbaum**

Coding: Implemented FGSM and PGD attacks, implemented ArcFace and ViT models, generated testing figures and images with perturbations, Paper: abstract, experiments, methodology, results and analysis, future work

**Xander Coomes**

Coding: OOP design for Perturbations and ML models, implemented simple cnn & linear models, plotting libraries & utils, Paper: results, related works, figures, intro, sources of error Organization: meeting planning, project direction

**Agam Makkar**

Coding: Initial testing with InceptionResNet_v1, ArcFace, and cosine similarities, researched and implemented the universal attack technique. Paper: Abstract, Introduction, Discussion and Future Work, Broader Impact, Result and Analysis, Figures

## 10 Code

Code can be found at the following Github Repository

## References


Nicolas Papernot, Patrick D. McDaniel, and Ian J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR*, abs/1605.07277, 2016. URL http://arxiv.org/abs/1605.07277.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. URL https://arxiv.org/abs/1412.6572.

Hailong Xi, Le Ru, Jiwei Tian, Bo Lu, Shiguang Hu, and Wenfei Wang. Adversarial attacks: Key challenges for security defense in the age of intelligence. In *2024 4th International Conference on Artificial Intelligence, Robotics, and Communication (ICAIRC)*, pages 41–46, 2024. doi: 10.1109/ICAIRC64177.2024.10900089.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *CoRR*, abs/2003.01690, 2020. URL https://arxiv.org/abs/2003.01690.

Efstathios Chatzikyriakidis, Christos Papaioannidis, and Ioannis Pitas. Adversarial face de-identification. pages 684–688, 09 2019. doi: 10.1109/ICIP.2019.8803803.

Debayan Deb, Jianbang Zhang, and Anil K. Jain. Advfaces: Adversarial face synthesis. *CoRR*, abs/1908.05008, 2019. URL http://arxiv.org/abs/1908.05008.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019. URL https://arxiv.org/abs/1706.06083.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. URL http://arxiv.org/abs/1610.08401.

David Sandberg. facenet: A unified embedding for face recognition and clustering. https://github.com/davidsandberg/facenet/blob/master/LICENSE.md, 2016. Accessed: 2025-06-06.

Jia Guo and Jiankang Deng. InsightFace: State-of-the-art 2d and 3d face analysis project. https://github.com/deepinsight/insightface, 2018. Accessed: 2025-06-06.

Hugging Face. google/vit-base-patch16-224-in21k: Vision transformer pre-trained on imagenet-21k. https://huggingface.co/google/vit-base-patch16-224-in21k, 2025. Accessed: 2025-06-06.

Jaehui Hwang, Dongyoon Han, Byeongho Heo, Song Park, Sanghyuk Chun, and Jong-Seok Lee. Similarity of neural architectures using adversarial attack transferability, 2024. URL https://arxiv.org/abs/2210.11407.

Enrique Alvarez, Rafael Alvarez, and Miguel Cazorla. Studying the transferability of non-targeted adversarial attacks. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2021. doi: 10.1109/IJCNN52387.2021.9534138.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017. URL https://arxiv.org/abs/1608.04644.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Discovering adversarial examples with momentum. *CoRR*, abs/1710.06081, 2017. URL http://arxiv.org/abs/1710.06081.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *CoRR*, abs/1511.04599, 2015. URL http://arxiv.org/abs/1511.04599.

# 11 Appendix

## 11.1 Apple Photos Experiment

One of our initial goals for the project was to design an imperceptible perturbation that could trick the Apple Photos Embedding Model into either misclassifying a face or failing to identify a face in the photo. After labeling a select number of Brad Pitt images, the apple photos app was able to detect photos containing the celebrity. Testing perturbations quickly proved to be difficult, as the Apple the testing interface was difficult(one needs to send and receive photos to a phone, not to mention a large time delay, where we needed to wait overnight). Nonetheless, we proceeded to feed it a handful of perturbed images from each of the models. Based on our very limited testing, the ResNet_v1(Casia), ResNet_v1(VGG) and ViT models were most effective, and, given that Apple likely implements a very deep embedding model, which aligns with our finding that similar architectures seem to transfer better, as seen in Figure 8.

## 11.2 PGD and Universal Attacks

Performing both the PGD and Universal attacks resulted in very similar general trends to the FGSM attack. We chose to highlight the FGSM attack because it resulted in the greatest percent decreases in accuracy across all models. We suspect that due to the more complex nature of the PGD and Universal attacks, the attacks are overfitting to the loss landscapes of the specific models they are being applied to, resulting in decreased transferability when tested on the other models in our experiment.

## 11.3 Additional Definitions

- *Targeted-attack:* A targeted attack involves an adversary generating a perturbation designed for the model to label the perturbed image as a "target class"
- *Obfuscation-attack:* An obfuscation attack, or non-targeted attack is designed to simply result in a misclassification.
- *Gradient-based attack:* Attacks using basic gradient computation, FGSM, PGD, Universal, pros: computational efficiency, ease of implementation, cons: perceptible perturbations, model dependency[Xi et al., 2024]
- *GAN-based attack:* Attacks using Generative Adversarial Networks, pros: Natural Generation, High degree of freedom, efficient training, cons: stability issues, model uncontrollability, model collapse [Xi et al., 2024]

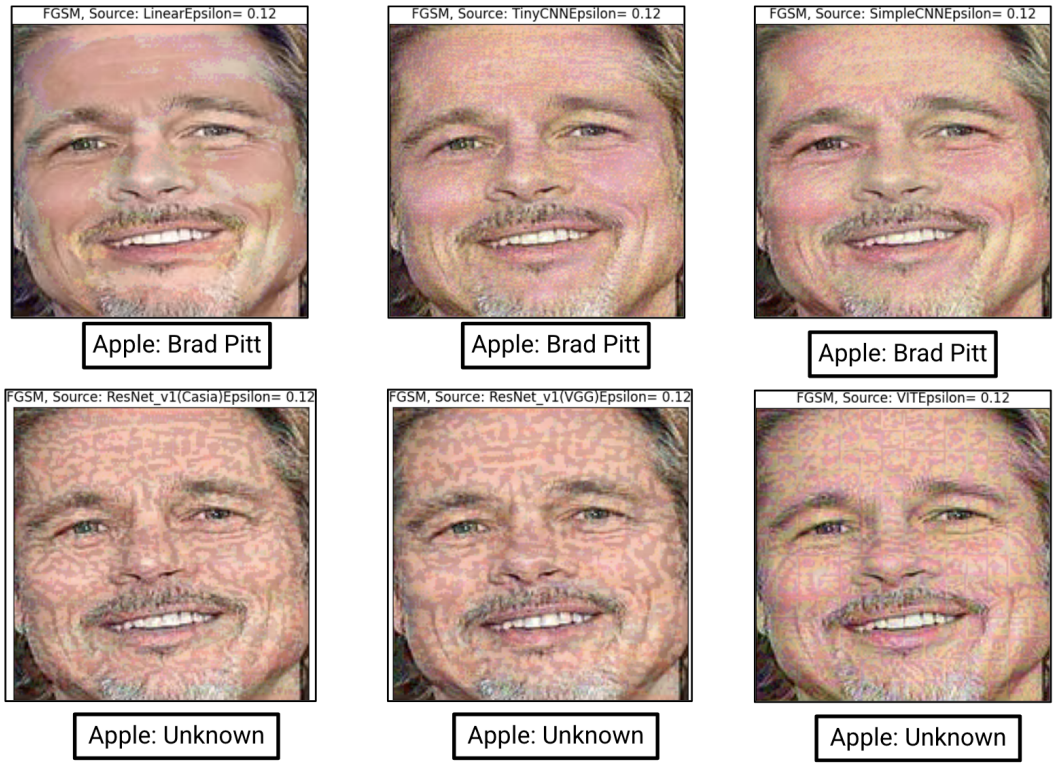


Figure 8: Testing apple photos with FGSM

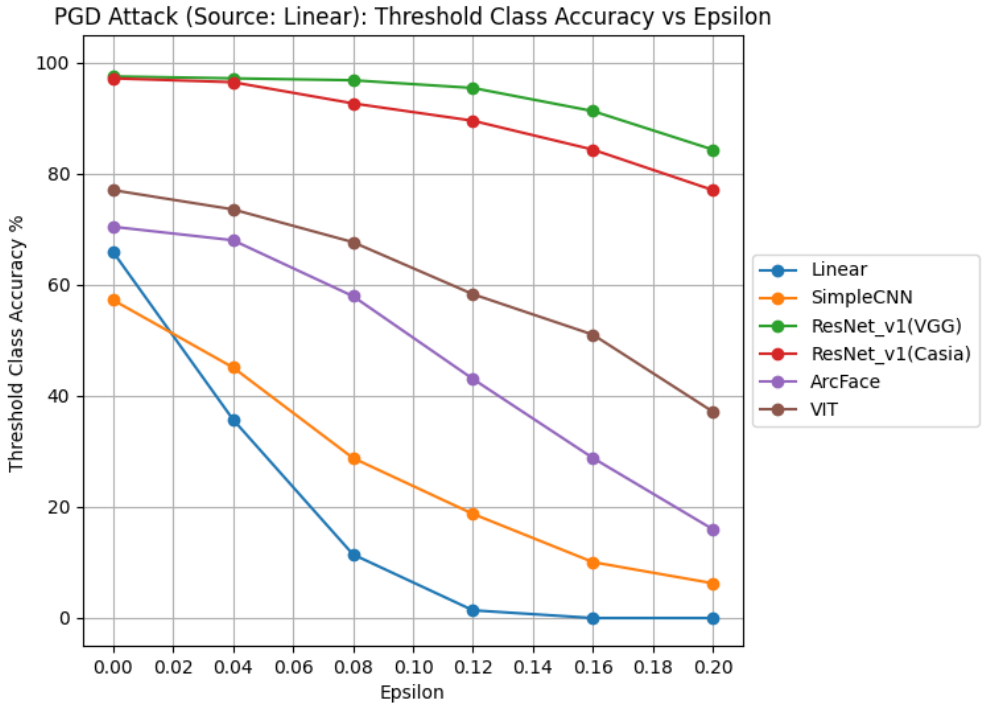


Figure 9: PGD Threshold accuracy vs epsilon (Source:Linear)

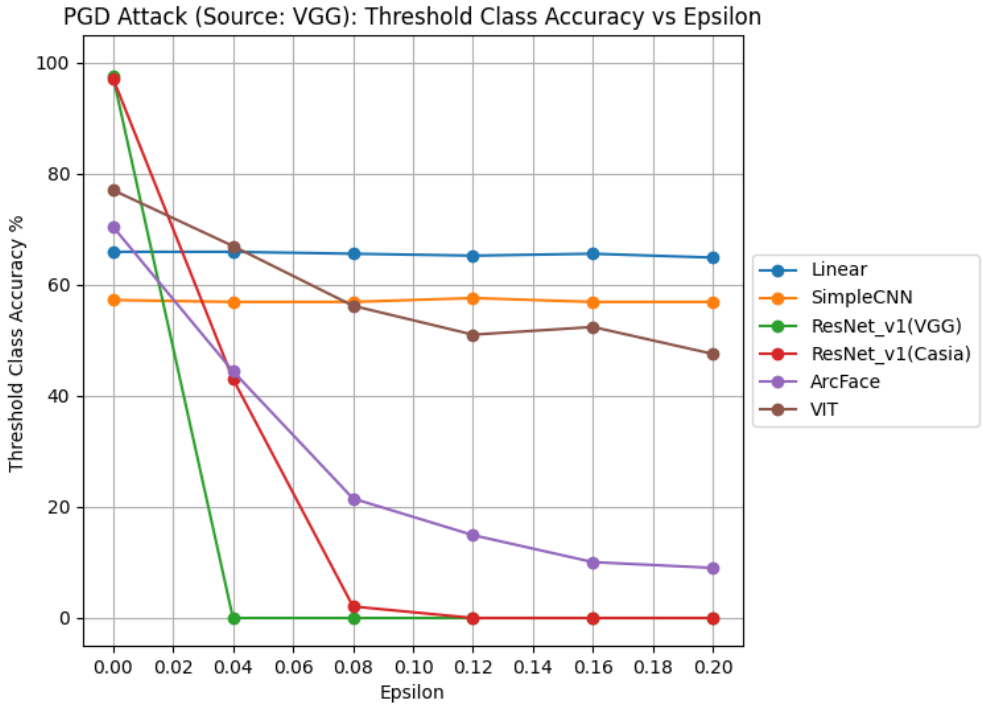


Figure 10: PGD Threshold accuracy vs epsilon (Source: VGG)

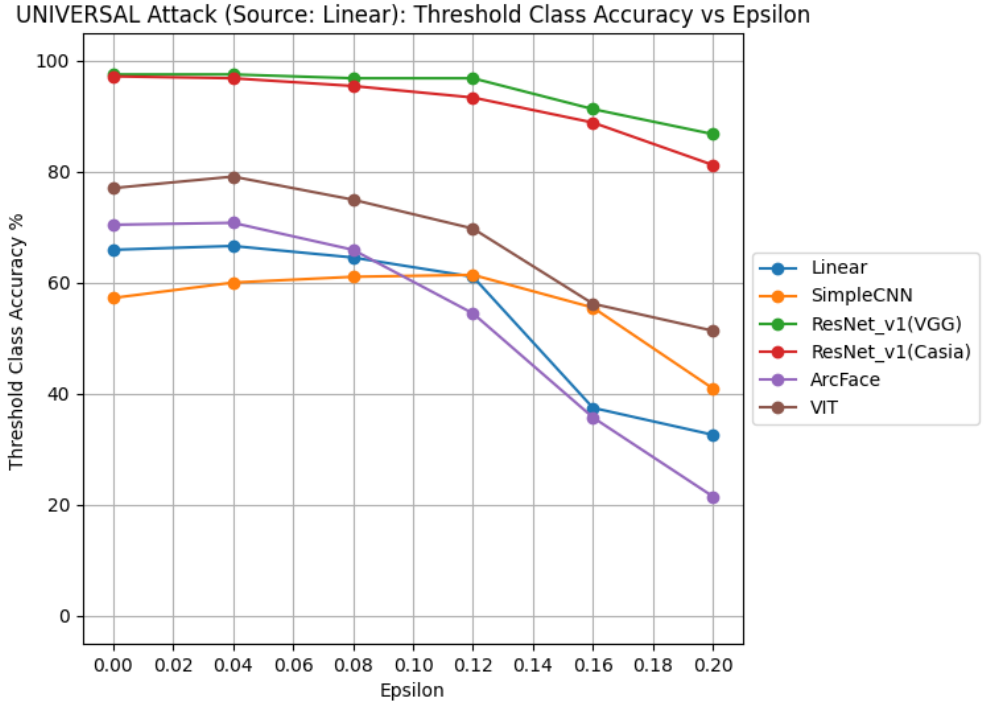


Figure 11: Universal threshold accuracy vs epsilon (Source:Linear)

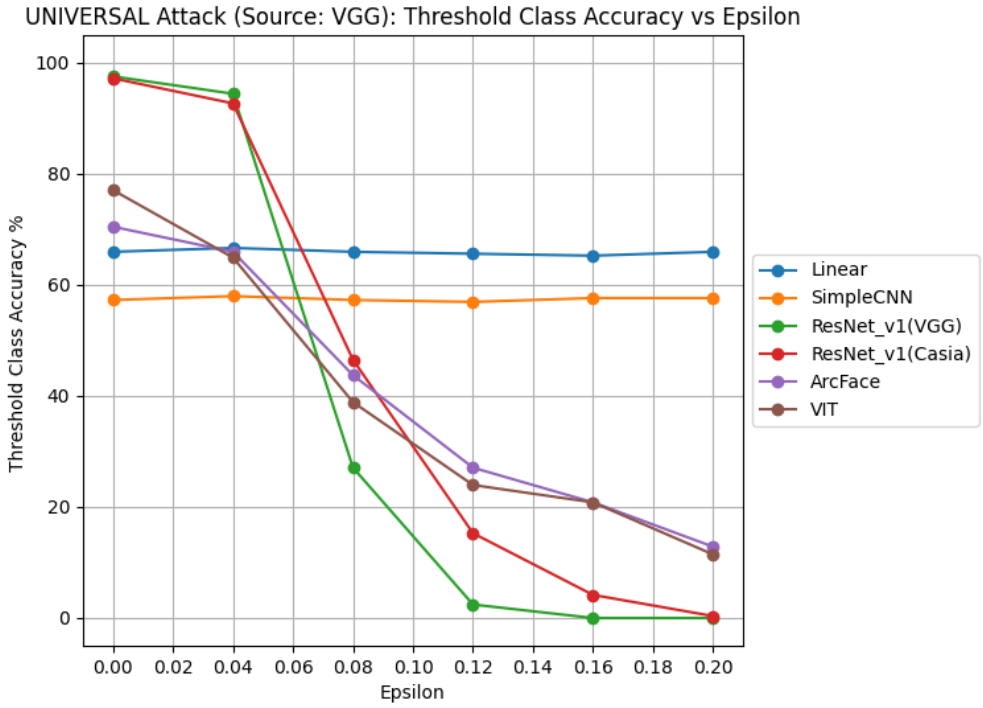


Figure 12: Universal threshold accuracy vs epsilon (Source: VGG)

### 11.4 Additional Sources of Error

#### 11.4.1 Accuracy of the Models

We attributed the transfer accuracy difference to the model architecture, but one source of error may be that the model accuracies (between simple models - linear and cnn) and model accuracies of complex architectures were very different.

#### 11.4.2 Experimental Error

There could very well be mistakes in our code which cause the results that we are obtaining. While we tried to control for these errors through testing accuracy on simple "noise" perturbations, among other things, this is a possibility.

#### 11.4.3 Embeddings Model

While we wished to examine the difference in transferability between architectures such as a transformer model, large CNN, small CNN, and a DNN, for all of the pretrained models, we imposed a cosine similarity metric on top, which added a similar structural component to all of the pretrained models. For this reason, the discrepancies of transferability we observed may have been a function of this embedding layer we constructed rather than the underlying model architecture.

#### 11.4.4 Independence of Data

Scraping so the data might not be a completely independent train and test set, as there may be some duplicate images in the train set and the test set.