Eluding Mass Surveillance: Adversarial Attacks on Facial Recognition Models

Andrew Milich
Computer Science Department
Stanford University
amilich {*at*} stanford {*dot*} edu

Michael Karr
Computer Science Department
Stanford University
mkarr {*at*} stanford {*dot*} edu

Abstract*—Our project analyzes the sensitivity of a deep neural network (DNN) for facial recognition to adversarial input images. We began by modifying a transfer-learned DNN that performs facial recognition using weights from a pre-trained Inception ResNet v1 model. Then, we created methods for generating adversarial input images, such as adding random noise or obscuring facial landmarks (ears, eyes, nose, and mouth). Unsurprisingly, our results indicated that adding random noise to an image reduced model performance. In most cases, clustering random noise around facial landmarks further reduced model prediction accuracy, thereby suggesting that these landmarks play an important role in facial recognition. Finally, we tested whether adversarial training, or including perturbed input images in model training, could increase model accuracy on our adversarial dataset. This defense technique did not prove particularly effective. Thus, our results suggest that these black-box attack mechanisms effectively reduced the accuracy of facial recognition models.

I. Introduction

Recent academic literature has demonstrated that deep learning models for image classification are often highly sensitive to small perturbations in input images [5], [10]. Past experiments have demonstrated that a variety of attack mechanisms, from changing a single pixel [17] to recoloring an image in the direction of the gradient of the loss function (an attack known as FGSM, or the fast gradient sign method), can significantly impact test accuracy [5], [9]. We sought to study this problem in the context of facial recognition. Are deep learning models for facial recognition more sensitive or robust to single-pixel, random noise, or FGSM attacks?

We were particularly interested in this project due to its timely political relevance: Facial recognition has proven a cornerstone of new deep learning mass surveillance applications, and journalists, technology executives, and think tanks have recently argued that facial recognition should be regulated or controlled by the government [16].

Prior to this project, neither of our group members had any exposure to training or testing deep learning models. In completing this project, we hoped to make a novel and timely contribution to analyzing adversarial perturbations while learning how to perform deep learning research.

II. Literature review

Generating adversarial examples to confuse DNNs has become a fast-growing field within deep learning. Ian Goodfellow's 2014 paper "Explaining and Harnessing Adversarial Examples" outlined the fast gradient sign method (FGSM) for perturbing sample images; this paper also relied on random perturbations as control experiments for comparing performance. Since then, researchers have released open-source software for generating adversarial images, such as the library Cleverhans [3] and DeepFool [13]. Other studies have presented algorithms for generating adversarial examples from real-world or live images; one paper presents an "adversarial patch" that, when added to images, can confuse the output of DNNs [2].

As attack mechanisms have become increasingly sophisticated, other papers have proposed defenses. In 2018, the paper "PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples," written

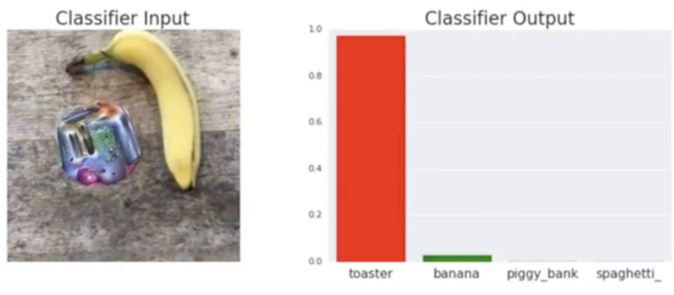

Fig. 1: On the left, the "adversarial patch" is placed next to a banana, causing classifier to predict (with high confidence) that the image contains a toaster [2].

by Yang Song and Stefano Ermon, proposed a method for using a generative model to detect and purify perturbations [15]. In our project, we decided to focus on black-box attacks where an attacker would not have access to the internal mechanics of a DNN [14]. This setting seemed more appropriate given our interest in the broader political ramifications of interfering with facial recognition.

III. Data

Our facial recognition model is trained on the Labeled Faces in the Wild (LFW) dataset, which contains over 13,000 images of over 5,000 individuals [8]. However, while some individuals are associated with only one or two photographs, others have far more training samples (for example, the LFW dataset contains 522 pictures of George W. Bush and 139 of Tony Blair). Given this discrepancy in training data, we restricted our model to train on individuals with more than 20 training samples. We also ran experiments where we trained our model on the same number of training samples per individual.

Fig. 2: Example photo from the LFW dataset.

In order to detect facial landmarks, we trained another DNN on a Kaggle dataset of 7,049 images with facial landmarks identified by (x, y) position.

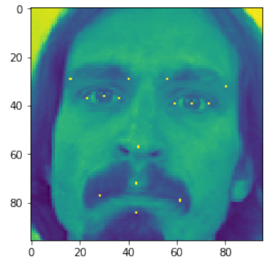

Fig. 3: Training sample from the Kaggle facial keypoints dataset [7].

IV. Facial recognition model

We began by developing a facial recognition model that could be used for testing adversarial inputs. We found that modifying a model developed by Cole Murray [12] provided a good fit for this project as we could easily modify training images, input dimensions, and classification parameters. In this section, we briefly describe how this model breaks down the facial recognition pipeline into three key steps: Preprocessing, learning, and classification [9].

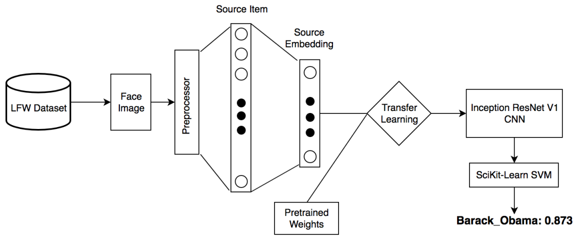

Fig. 4: Non-adversarial training flow.

In the preprocessing step, the model prepares images for facial recognition, which involves several steps. The first of which, segmentation, occurs by identifying the largest face present in any image, which is followed by the second step - cropping the image. The face identification step is conducted using Carnegie Mellon University's facial landmark predictor [1]. The image is subsequently rotated and aligned so that each image has its respective facial features in the same pixel-region of the image.

The learning step of the model takes the preprocessed image and uses it to update weights in order to classify individual faces. This is done by generating 128-dimensional embeddings for each face. In order to create these embeddings, we use the Inception ResNet v1 (a convolutional neural network that is similarly complex to Inception v3 but requires less computing power when using a batch size of 128). This was desirable to us since the ability to use Batch-Norm on the auxiliary classifiers was favorable to our facial recognition task primarily for the purpose of regularization, as we wanted to avoid overfitting, especially towards the end of our training process [4]. Since we did not have the resources nor the time to fully train the Inception model from scratch, we used a set of pre-trained weights for our model [6].

Classification is subsequently performed by using 128-dimensional image embeddings as inputs to the Scikit-learn SVM classifier. The classifier uses a linear kernel and outputs a probability for each person (i.e. each class) in our dataset. The model then chooses the highest probability class as its final prediction. These steps are shared with Cole Murray's facial recognition model; however, we have experimented with modifications to the SVM classifier (such as using a different kernel function) and modifications to the input and preprocessing steps (we tried switching from the CMU to the OpenCV facial cropper, which did not perform as well).

V. Attack methods

In Figure 5, we provide an overview of the two types of attacks used to generate adversarial examples.

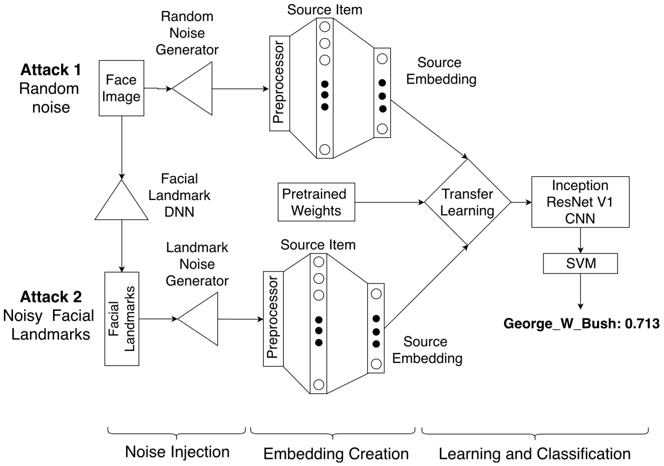

Fig. 5: Two different attack mechanisms were used: Random perturbations, and adding noise to facial landmarks.

A. *Random noise*

We wrote a script that adds random noise to images. We experimented with two types of noise: Gaussian Noise, which perturbs images at a given location based on sampling from a Gaussian distribution, and salt-and-pepper noise, which recolors randomly chosen pixels as white or black. Ultimately, we decided to use a modification of salt-and-pepper noise as our baseline attack mechanism; in this algorithm, pixels are randomly chosen and recolored as solid red, green, or blue. As outlined in Figure 5, this attack mechanism runs an image from our dataset through a random noise generator before using it as input to our classifier.

B. *Obscured facial landmarks*

This more sophisticated attack mechanism requires two steps: Identifying facial landmarks in an input image using a DNN, and then using random noise to perturb these landmarks. Below, we provide greater detail about each step in this process.

1) **Identifying facial landmarks**: We experimented with multiple DNNs to identify facial landmarks in the Kaggle facial keypoints dataset, including using 1D and 2D convolution layers. Ultimately, we saw the best performance (including reasonable training times) from a network that uses one max pooling layer, a flattening layer, two pairs of fully connected and dropout layers, and an additional fully connected layer. This model results in an average loss of 2.99 pixels from predicted to actual facial landmark locations. However, we also found that this network configuration - particularly our use of dropout layers - allowed our model to generalize well: It achieves relatively low test error on the Kaggle dataset as well as good results on images from the LFW dataset. We discuss the performance of our facial landmark DNN further in the Results section.
2) **Perturbing facial landmarks**: Our facial landmark identifying DNN outputs a list of points $p = [(x_1, y_1), \dots, (x_n, y_n)]$ that represent bounds for facial features; for example,

three points define each eyebrow and four points outline an individual's mouth. We then made a modification to our random noise generator in order to generate noise clustered around these points. The facial landmark noise generator used a multivariate Gaussian with a scaled identity covariance matrix. We generated the noise by sampling Gaussian noise from this distribution.

VI. Defense mechanisms

A. Adversarial training

Adversarial training - one technique for defending DNNs against perturbed inputs [10] - involves including perturbed images in the training set. To perform adversarial training on our model, we expanded our training set to include copies of each image with random perturbations and with obscured facial landmarks.

VII. Results

Our facial recognition model was able to achieve an overall accuracy of 94.6% across all classes. We used a 0.7-0.3 train test split on the LFW dataset, and the model was trained using the Google Cloud Compute Engine on a machine equipped with tensor processing units (TPUs). Below, we present model performance on adversarial inputs.

Class name	**Raw model**	**Noisy images**	**Obscured landmarks**
Bill Clinton	0.99	0.75	0.58
George W. Bush	0.98	0.91	0.88
John Negroponte	1.0	0.63	0.63
Hamid Karzai	1.0	0.67	0.50
Tony Blair	0.97	0.69	0.71

Table I: Model performance on raw images, noisy images, images and images with obscured landmarks. This model was trained on all training images and thus had imbalanced class sizes (such as George W. Bush, which contained 500+ images, and others that had only 20).

Class name	**Raw model**	**Noisy images**	**Obscured landmarks**
Bill Clinton	0.88	0.86	0.50
George W. Bush	0.92	0.41	0.15
John Negroponte	0.75	0.38	0.13
Hamid Karzai	1.0	1.0	0.50
Tony Blair	0.93	0.46	0.36

Table II: Model performance on raw images, noisy images, images and images with obscured landmarks. This model was trained on classes limited to 20 images each.

Class name	**Raw model**	**Noisy images**	**Obscured landmarks**
Bill Clinton	0.63	0.56	0.29
George W. Bush	0.98	0.88	0.60
John Negroponte	1.0	0.50	0.75
Hamid Karzai	0.83	0.58	0.60
Tony Blair	0.88	0.58	0.73

Table III: Adversarial training model performance on raw images, noisy images, and obscured landmarks. This model was trained on both raw and perturbed inputs.

Model name	**Average confidence**
Raw images	0.73
Noisy images	0.65
Obscured landmarks	0.51

Table IV: Average confidence of predictions on raw images, noisy images, images and images with obscured landmarks. This model was trained under the same conditions as reported in Table I.

VIII. Discussion

We begin by discussing the performance of our model trained on imbalanced classes - i.e. the results in Table I. As we initially hypothesized, adding random noise to input images reduced model accuracy. In some cases, such as John Negroponte, the decrease in performance when subjected to random noise is dramatic (37%); in others - such as George W. Bush - our facial recognition model did not demonstrate significantly lower accuracy for inputs with random noise. For the majority of classes in Table I, clustering noise around facial landmarks resulted in additional accuracy drops; while accuracy decreased notably in some cases (17% for Hamid Karzai and for Bill Clinton), there was not a significant decrease for others; accuracy remained the same for John Negroponte and actually increased 2% for Tony Blair. Initially, we believed that our model highly weighted features from facial landmarks, such as an individual's eyes, nose, and mouth; this would suggest that clustering noise around landmarks would reduce model accuracy. Although this hypothesis was validated in some cases, the effects were not as pronounced as initially anticipated.

One explanation for this attack's limited effectiveness is our use of different datasets to train and test our facial landmark recognizer. While the facial landmark DNN was trained on the Kaggle facial keypoint dataset [7], it is run on scaled images from the LFW dataset to generate inputs for our facial recognition model. Although this generally produced reasonable-looking output (see Figure 6), it may partially explain why our facial landmark attack did not achieve as dramatic results as expected.

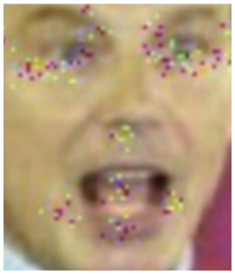

Fig. 6: An training image from the LFW datasert with noise clustered near facial landmarks.

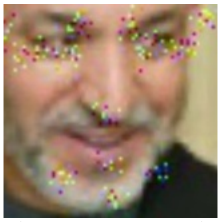

Fig. 7: Facial landmark recognition achieves limited success on Hamid Karzai, potentially due to his beard.

Another possible explanation for the lower-than-expected effectiveness of our facial landmark attack is limitations in the Kaggle facial keypoints training data. For example, individuals with beards in the LFW dataset (such as Hamid Karzai) often had facial landmarks classified incorrectly, as facial landmarks around their mouth were generally confused with the individual's beard (see Figure 7). Furthermore, since images are relatively small (160×160 in the LFW dataset and 96×96 in the Kaggle facial keypoints dataset), and were scaled down as inputs to our facial landmark recognizer, the OpenCV scaling algorithm used to input LFW images to our landmark DNN could have have had relatively minor effects on the accuracy of this step.

Given the relatively small drop in accuracy for George W. Bush when testing on obscured facial landmarks, we also hypothesized that classes with a higher number of training samples (such as George W. Bush) would not be as susceptible to attack. To test this hypothesis, we decided to retrain our model on only 20 images per class. Table II presents these results. Unsurprisingly, this retrained model achieved lower accuracy on raw images; this is likely due to our use of a much smaller training set for some individuals. Accuracy decreases even more dramatically in Table II for noisy images and obscured landmarks, thereby suggesting that limiting training data increases susceptibility to attack.

In Table III, we present the results of performing adversarial training on our model. Generally, this model resulted in unexpectedly low accuracy in some cases (such as classifying raw images for almost all classes) and surprisingly high accuracy in others (such as relatively high performance on data with obscured landmarks across). Our use of randomness in both attack mechanisms may explain these results: Inconsistency in random noise across training samples may have made it difficult for our model to highly weight features not associated with any perturbations. As a result, our adversarial training model yielded low accuracy on inputs with random noise but higher accuracy on perturbed landmarks.

Finally, it is important to emphasize the difference between high classification accuracy and highly confident predictions. While classification accuracy may be relatively high in some cases, our model's confidence in each prediction (as reported in Table IV) is generally significantly lower when testing on adversarial inputs. Our model outputs a final classification for each image based on the highest probability class, where each class indicates a specific individual. Thus, a prediction that an image x contains individual y with confidence 25% may still lead the model to classify x as y if no other classes have a higher confidence. Table IV illustrates that our model's confidence decreased for both attack mechanisms, thereby suggesting that they were effective in reducing prediction confidence.

IX. Future work

When initially planning our project, we hoped to examine whether clustering algorithms, such as K-means, could provide effective defenses against random perturbations. However, as K-means has a relatively high and inefficient runtime, we chose to focus on developing additional attack mechanisms and testing adversarial training. However, this could provide an additional method of defending against the attacks tested in this paper.

Although we sought to study black-box attacks, testing our model on inputs generated with FGSM may have permitted a closer examination of whether adversarial training can defend facial recognition DNNs. However, while this is a promising direction for future research, we chose not to focus on FGSM attacks given our desire to understand how an individual could undermine facial recognition without knowledge of a model's internal parameters.

Thus, given the political relevance of this topic, we also hoped to explore the possibility of creating a physical "adversarial patch" that could be worn to confuse facial recognition DNNs. As our results suggest that facial landmarks are relevant to classification - and particularly so for models with less training data, perhaps an individual could wear a patch near their nose or mouth to evade recognition [2].

X. Link to GitHub repository

Our GitHub repository is available at https://github.com/amilich/face.

References

[1] Amos, Brandon, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. "OpenFace: A general-purpose face recognition library with mobile applications." CMU, 2016. https://github.com/cmusatyalab/openface

[2] Brown, Tom B. et al. âĂIJAdversarial Patch.âĂİ ArXiV, December 2017. https://arxiv.org/abs/1712.09665.

[3] "Cleverhans: An adversarial example library for constructing attacks, building defenses, and benchmarking both." GitHub, https://github.com/tensorflow/cleverhans.

[4] Ioffe, Sergey, Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv, February 2015. https://arxiv.org/pdf/1502.03167v3.pdf.

[5] Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." ArXiv, December 2014. https://arxiv.org/abs/1412.6572.

[6] "Inception in TensorFlow." GitHub, https://github.com/tensorflow/models/tree/master/research/inception.

[7] Kaggle. "Basic Fully Connected NN." https://www.kaggle.com/madhawav/basic-fully-connected-nn/data.

[8] âĂIJLabeled Faces in the Wild." http://vis-www.cs.umass.edu/lfw/.

[9] Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks." ArXiv, June 2017. https://arxiv.org/pdf/1706.06083.pdf.

[10] Makelov, Aleksandar et al. "Towards Deep Learning Models Resistant to Adversarial Attacks." ArXiv, June 2017. https://arxiv.org/pdf/1706.06083.pdf.

[11] "Medium-Facenet-Tutorial." GitHub. https://github.com/ColeMurray/medium-facenet-tutorial.

[12] Murray, Cole. "Building a Facial Recognition Pipeline with Deep Learning in Tensorflow." *Hacker Noon*, https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning\-in-tensorflow-66e7645015b8.

[13] Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard. "DeepFool: a simple and accurate method to fool deep neural networks." ArXiv, July 2016. https://arxiv.org/abs/1511.04599.

[14] Narodytska, Nina and Shiva Prasad Kasiviswanathan. "Simple Black-Box Adversarial Perturbations for Deep Networks." ArXiv, December 2016. https://arxiv.org/pdf/1612.06299.pdf.

[15] Song, Yang, Taesup Kim, Sebastian Nowozin, Stefano Ermon, Nate Kushman. "PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples." ArXiv, https://arxiv.org/abs/1710.10766.

[16] Selyukh, Alina. "Microsoft Urges Congress To Regulate Facial Recognition Technology." NPR, December 6, 2018. https://www.npr.org/2018/12/06/674310978/microsoft-\
urges-congress-to-regulate-facial-recognition-technology.

[17] Su, Jiawei, Danilo Vasconcellos Vargas, Sakurai Kouichi. "One pixel attack for fooling deep neural networks." ArXiv, October 2017. https://arxiv.org/abs/1710.08864.