Face Encryption

SDS3386 Data Science Lab

Fall 2022

Team TensorOverflow

Xiang Li 300056427
Jiaxun Gao 300063462
Bowen Zeng 300115382

Course Coordinator: Tanya Schmah

November 30, 2022

University of Ottawa

uOttawa

Contents

1 Introduction

Recent reports (?, ?) confirm the privacy risks associated with big data in public social media. The amount of user-generated content uploaded to the internet is increasing rapidly, but large corporations such as Google and Facebook have been misusing it without their knowledge (?, ?). When consumers upload sensitive facial photographs, it is difficult to preserve their privacy. The company may use these pictures to develop their machine learning algorithms, which could result in privacy breaches. **How can we protect privacy when sharing facial images?**
In this research, we offer a viable privacy-protecting solution based on adversarial attacks and facial recognition technology. After detecting a face in a picture using a facial recognition model, we encrypt the face with noise generated by adversarial attacking model. The composed image may appear as clear as the original, but the facial recognition software will recognise a different individual in it.

2 Dataset

2.1 CelebA

CelebA is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including **5,000 celebrity identities**, **202,599 face images**, and **40 binary attributes** annotations per image. The dataset can be employed as the training and test sets for the following computer vision tasks: face attribute recognition, face detection, landmark (or facial part) localization, and face editing/synthesis.

Figure 1: CelebA dataset

CelebA is utilised as the face recognition model's dataset set in this project. The training set has 4429 photos while the test set has 1267 images.

2.2 Private Dataset

The private dataset consists of photos collected by team members. The photos were taken in SITE and feature various facial expressions.
This dataset is utilized to perform adversarial attacking on the face recognition model. The training set has 121 photos while the test set has 15 images.

3 Methodology

3.1 Adversarial Attacks

Adversarial attacking is a technique that can be used to fool machine learning models. It is a type of attack that aims to change the input data in a way that the model will misclassify it. The adversarial attacking technique is based on the fact that machine learning models are vulnerable to small perturbations in the input data. The perturbations are usually imperceptible to the human eye, but they can cause the model to misclassify the input data.
Adversarial examples are hard to defend against because it is difficult to construct a theoretical model of the adversarial example crafting process. Adversarial examples are solutions to an optimization problem that is non-linear and non-convex for many ML models, including neural networks. Because we don't have good theoretical tools for describing the solutions to these complicated optimization problems, it is very hard to make any kind of theoretical argument that a defense will rule out a set of adversarial examples.

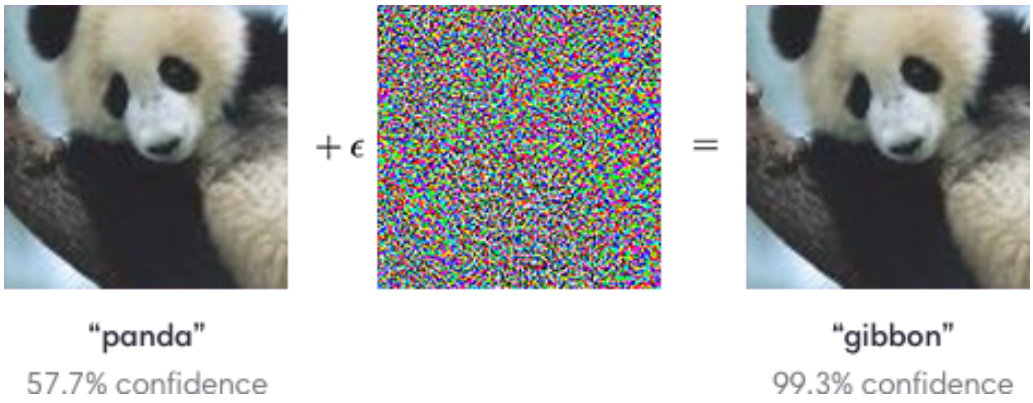


Figure 2: An adversarial input, overlaid on a typical image, can cause a classifier to miscategorize a panda as a gibbon.

Another reason is they require machine learning models to produce good outputs for every possible input. Most of the time, machine learning models work very well but only work on a very small amount of all the many possible inputs they might encounter. (?, ?)

3.2 Backbone Model: ResNet-18

ResNet-18 is a convolutional neural network (CNN) that is used as a backbone model in this project. The architecture can be illustrated as figure 3 (?, ?). It is a 18-layer deep neural network that is trained on the ImageNet dataset. The ImageNet dataset is a large dataset that contains 1.2 million images with 1000 classes. The ResNet-18 model is trained on the ImageNet dataset to classify the images into 1000 classes.

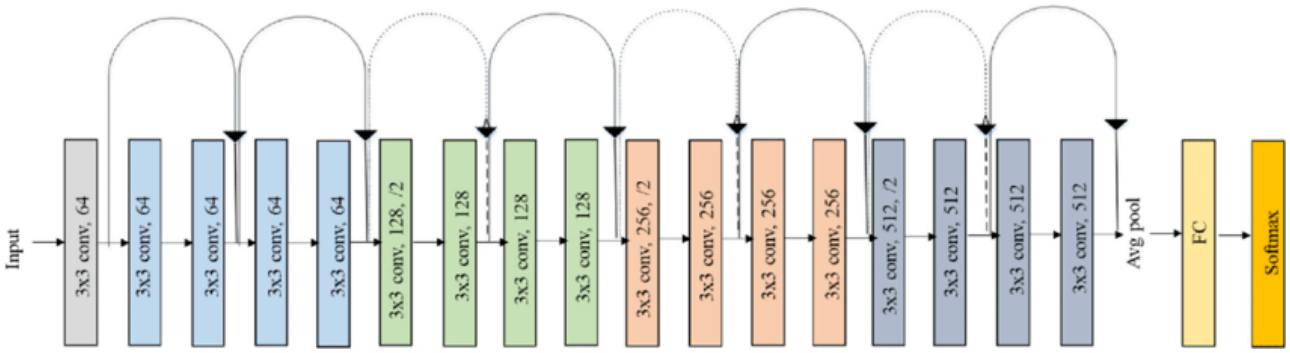


Figure 3: ResNet-18 Architecture

4 Data Wrangling

4.1 Faces extraction from recorded video

After capturing three videos, OpenCV2 is used to sample 40 frames from each video. The faces are then rotated 180 degrees to accommodate package mediapipe's face detection model. Then, we crop the faces from the frames using the centre of the bounding box's coordinates. The faces are then saved in a folder named `private_dataset` and added to `CelebA_HQ_facial_identity_dataset`.
This process is done in `Data_generation.ipynb.ipynb`.

4.2 Associate id with name

After downloading the CelebA dataset from the official website, we utilise CelebA-HQ-to-CelebA-mapping.txt to generate a map `hq_A_mapping` from an id to the image file name, such as `5: 000615.jpg`. Then, we use list_identity_celeba.txt to generate the second map `id_name_mapping` from a file name to its corresponding identity's name, for instance: `000615.jpg: Martha Hunt`.
The benefit of this process is that we can now use the id to determine an identity's name. By example, we may use the following code: `id_name_mapping[hq_A_mapping['5']]` to determine the name of the individual with id = 5.
This process is done in `Preprocessing.ipynb`.

4.3 Transforming the dataset

We resized the images to 224x224 because their original size was too large for our model, which could cause performance issues. After that, we augment the tensor by giving it a random horizontal flip as part of the transformation.
This process is a part of `simple_model.ipynb`.

5 Modelling

5.1 Face Recognition

Before we can achieve our objective, we need a face recognition system to use as an attacking target. To recognise faces, we employ a pre-trained ResNet-18 model in pytorch. In the neuron network, we use cross entropy loss as the loss function and stochastic gradient descent (SGD), a simple yet highly effective method for fitting linear classifiers and regressors under convex loss functions, as the optimizer. The model is trained for 10 iterations on the CelebA-HQ dataset.

5.2 PGD Attack

To generate adversarial examples, we use the PGD attack (?, ?). In this project, we use the gradient of the loss function to generate adversarial examples. The attack is iterative and uses a step size to determine the size of the perturbation. The attack is also constrained by a maximum perturbation size.

5.2.1 Non-targeted Attack

In a non-targeted attack, the goal is to generate an adversarial example that is misclassified by the target model A. Which means we are **maximizing** the loss function with respect to the target class A. The attack is as follows:

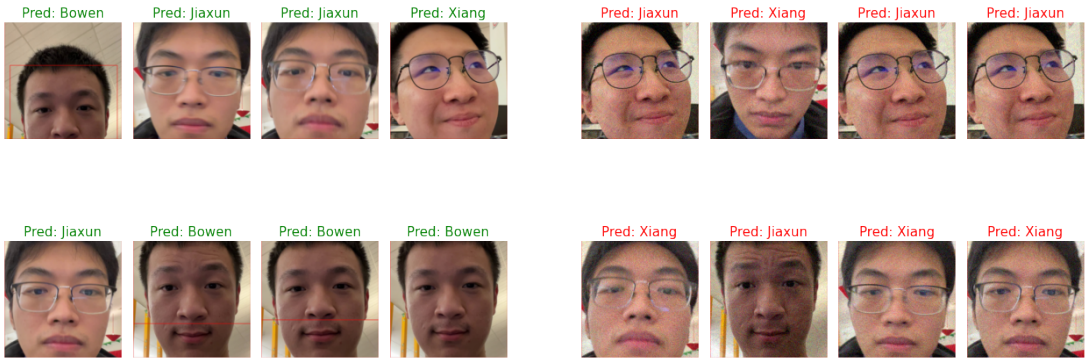
1. Generate a random noise tensor δ with the same shape as the input image.

2. Calculate the gradient of the loss function with respect to the noise tensor δ.
3. Add the gradient to the noise tensor δ.
4. Clip the noise tensor δ to the range $[-\epsilon, \epsilon]$.
5. Add the noise tensor δ to the input image.
6. Repeat steps 2 to 5 until the model misclassifies the image.

The loss function (maximizing the difference with the true label) can be represented as:

```
Difference(Model(Face Image + Delta), True Label)
```

where `Model(Face Image + Delta)` is the prediction of perturbed data.



(a) Prediction without attack *(b) Prediction with non-targeted attack*

Figure 4: Example of a non-targeted attack

With above steps, we can generate an adversarial example that is misclassified by the target model. In figure 4, the image on the left demonstrates that Resnet-18 correctly predicted the original photos. After adding the PGD model's generated noise. The image on the right depicts Resnet-18 classifying the adversarial example as a different category.

5.2.2 Targeted Attack

In a targeted attack, the goal is to generate an adversarial example that is misclassified A by the target model and classified as a specific class B. All the steps are the same as in a non-targeted attack, except for the last step should be: **Repeat steps 2 to 5 until the model classifies it as the target class.** Which means we are **minimizing** the loss function with respect to the target class B.
The loss function (minimizing the difference with the target label) can be represented as:

```
Difference(Model(Face Image + Delta), Target Label)
```

6 Visualisation and Real-Time Interface

6.1 Resnet-18 Performance

Figure 5 demonstrates that with 25 epochs, the face recognition model Resnet-18 achieves 99.98% accuracy on the training set and 81.93% accuracy on the test set. We halted training at this point because any additional epoch would result in an overtraining problem.

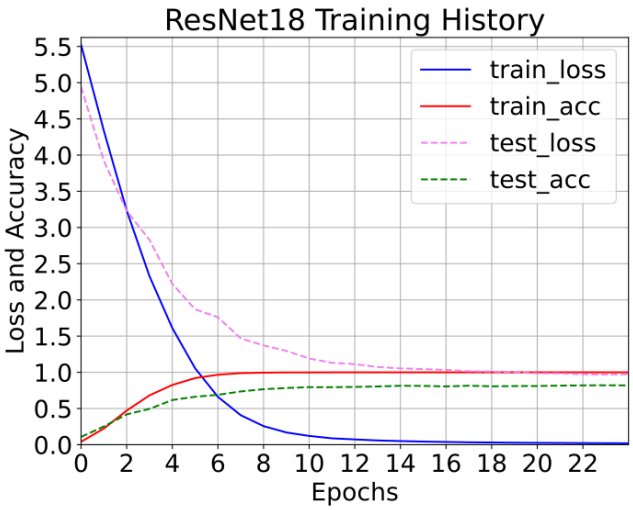


Figure 5: Resnet18 performance

6.2 Noise Performance

6.3 Real-Time Interface