CS - 2490 : Advanced Machine Learning
Spring 2022
Professor Lipika Dey and Professor Tirthankar Dasgupta
Final Project

Blurring and Anonymizing Faces for Security

Anusha Pant

11th May, 2022

1. About the Project

1.1 Problem Statement

With CCTV surveillance increasing in the world at a fairly fast pace, decrease in privacy is a major concern. The goal of this project is to ensure that the privacy of all recorded people is maintained, while also not compromising on the desired level of security – by developing a model which blurs and anonymizes faces of people, with the option of the disclosure of the identity of the blurred faces when legally required.

1.2 Example Scenario and Project Goal

Let us consider a place, such as a university, with CCTV cameras that have been put up at various places on campus.

Now, in general, recording the daily actions of people may be considered an invasion of privacy in such a scenario. It is here that a blurring and anonymizing model comes in handy.

However, let us now think of a specific kind of situation where this blurring/anonymization may not be ideal. Let us assume that Person A steals something from Person B's room. While they've been caught on camera, their face is blurred for privacy reasons, and they cannot be recognized. However, in this situation, it is very important for the given university to be able to catch Person A.

It is in such a scenario that, while keeping the importance of privacy in mind, a certain level of security also needs to be maintained.

Hence, the goal of this project is two-fold. Firstly, it aims to address the issue of the decrease in privacy as a result of the increase in CCTV surveillance around the world. Secondly, it aims to also ensure that the introduction of these privacy-protecting methods does not happen at the cost of security.

1.3 Proposed Solution

In order to attain the above mentioned goal, I have divided my solution into two broad steps.

1. The first step was developing a system which takes an image as its input, detects faces within it, and blurs them for the purpose of preserving privacy and anonymizing them. With this, the privacy preserving aspect of the project becomes easy. Given an image (which can also be a video feed), each face within it can be altered to be blurred.
2. While step 1 preserves privacy, we also do not want to compromise on the desired level of security. For this, a possible solution is as follows:
   Given a set of *n* people who are "suspects" or "known" with regard to a particular situation, a model is developed, which is trained on blurred face images of these people. Now the CCTV image, which contains a blurred face, is brought in. This blurred face is then fed into this model, and is classified as one of the *n* people.
   In order to train this blurred face recognition model, I implemented two different approaches:

(1) K Nearest Neighbours
(2) VGG-19 based transfer learning

I have described both of these in detail in the following sections.

1.4 Submission Details

My project submission contains the following:

- This report, listing out the details of the project, the underlying idea, and its implementation.
- AML-Final-Part1.ipynb - a Google Colab notebook which contains the code to blur faces within an image via face detection and Gaussian blur.
  Link :
  https://colab.research.google.com/drive/1t0qpRmv0LZwHo_1JneSaXDe1P3UeMHjy?usp=sharing

- AML-Final-Part2.ipynb - a Google Colab notebook containing the implementation of the two approaches to solving the security related (blurred face classification) issue, via KNN and VGG-19.
  Link:
  https://colab.research.google.com/drive/1K2C7-dgaun_2P-PGewwleBU8UbYyNNP4?usp=sharing

2. Implementation

As mentioned in the previous section, the implementation of my project is divided into two broad sections.

2.1 Blurring Faces

The first step in the implementation of the project is creating a model that takes an image as an input, detects any faces within it, and blurs each of these faces.

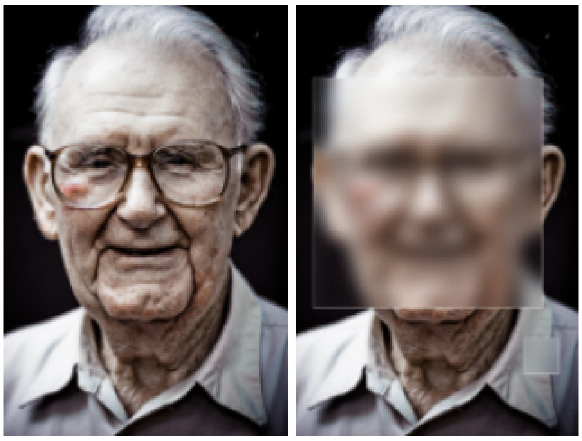
For this portion of the project, I chose a dataset of faces from Kaggle, called **Human Faces**. Each image here contains one or more faces within it.

In order to detect faces, I used OpenCV's support for face detection using Haar Cascades.

I defined a function, called blurFaces, which takes an image as input. This function performs the following steps:

- Using Haar Cascades, the faces in the image are detected. For each face, the bottom left coordinates (x,y) are returned, along with the width and the height, all stored in a variable called *faces*.
- For each x,y,w,h returned in *faces*, the corresponding part of the image is isolated in the form of a rectangle, and blurred using OpenCV's Gaussian Blur function.

- The original image has now been altered in such a manner that each face is no longer clearly visible.


A sample input image, and the final corresponding output image

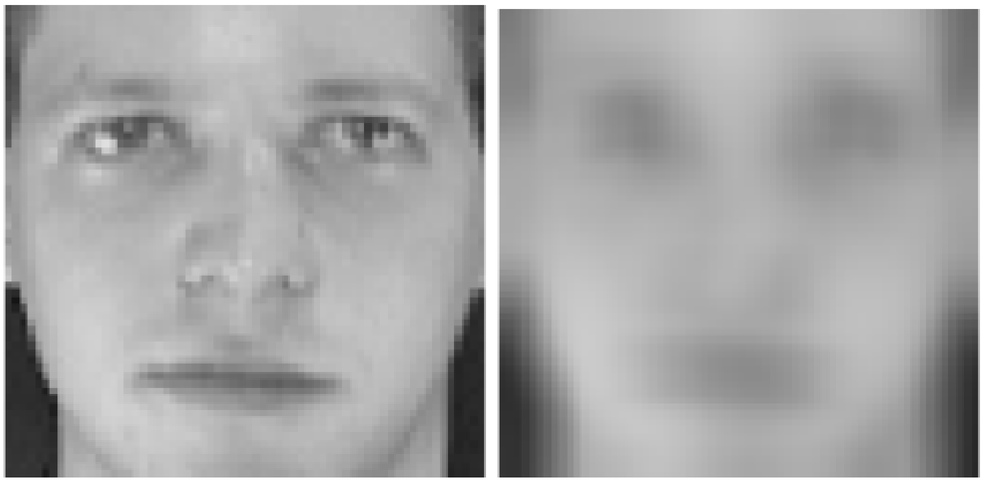
As can be seen above, given an image, it is blurred and hence altered in such a way that the identity of the individual can no longer be determined by looking at the image, fulfilling the purpose of maintaining one's privacy.

While we've successfully been able to blur faces, the goal is to also ensure that the desired level of security is not compromised. As explained in Section 1.2, we also need a way for at least authorities to still be able to identify an individual, based on their blurred image, if the need arises. I have discussed two potential approaches to this below.

2.2 The Dataset

In order to implement this part of the project and give a demo of the desired functionality, the dataset that I used is the Olivetti faces dataset. This dataset contains 400 images, belonging to 40 unique classes. Each image contains a single face (face detection is therefore not required here).

The first step is to take each image and blur it, as can be seen below.



Once this is done, a new dataset is formed, consisting only of these blurred images. The label corresponding to each image remains the same. This new dataset is then split into test and training subsets, where 10% is assigned to the former, and 90% to the latter. The majority of the dataset is used for training since we only have 400 samples in total. The splitting is done in such a way that for each class, 90% of the images are added to the training set, and the remaining 10% to the test set. This is done to ensure that each class is represented equally in the training set, and the model can be tested on images from each class.

2.3 Identifying Blurred Faces via K-Nearest Neighbours

The first approach is to identify a blurred face via K-Nearest Neighbours.
This requires the following:

- A blurred image of someone who needs to be identified.
- A set of images belonging to suspects/people from whom the above person needs to be identified
- Each image in the dataset should be labelled, i.e., there should be a corresponding name, ID number, etc, to be able to identify them uniquely.

The procedure can then be defined as follows:

- The entire dataset is blurred following the procedure mentioned in the previous section, to form a new blurred dataset. The corresponding labels remain the same.
- The dataset is transformed to a lower dimensional space as described below.
- A KNN model is trained on this dataset
- The value of K = 1, i.e., for a given image, the class of the closest image will be the output
- The original blurred image is fed into the model, to obtain the identity of the closest person from the given dataset.

Now, each of the images in our dataset has 4096 features, since the dimension of each is 64x64. Since not each and every of these 4096 features is of significant relevance, we first perform Principal Component Analysis (PCA) to bring this number down. These principal components are new features which are linear combinations of the old ones, such that we can retain as much of the initial information with a lower number of features. The first newly obtained feature will contain the most amount of information, followed by the second, and so on. In my case, I brought the number of features down to 120. These 120 vectors are our PCA Eigenvectors, or Eigenfaces as we call them here. Each of the images in the test and training dataset can now be represented as a linear combination of these 120 vectors.

The model is then trained on this new training set, which contains the blurred images as linear combinations of 120 features, and the labels corresponding to the original image.

I then tested the model on the test set, and obtained predictions for each image. These predictions were then compared to the actual label for each.

As can be seen in AML-Final_part2.ipynb, the resultant accuracy is 90% . This means that, given a blurred image of a face, roughly 90 out of 100 times, the person in the image will be identified correctly.

This approach seems to work reasonably well. While the individual's privacy is being maintained by cleaning their image, authorities can still identify people when needed – with an accuracy of 90%. The corresponding confusion matrix can be seen below:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

An example of the model making a prediction is given below:
True label and image → 10

```
10
<matplotlib.image.AxesImage at 0x7f18ae6aa590>
```

Predicted label →

```
print(prediction)
```

```
[10]
```

As can be seen, the model has been able to predict the correct label.

2.4 Identifying Blurred Faces via a VGG-19 based Model

The second approach to identifying someone based on a blurred image is using a VGG-19 based model.

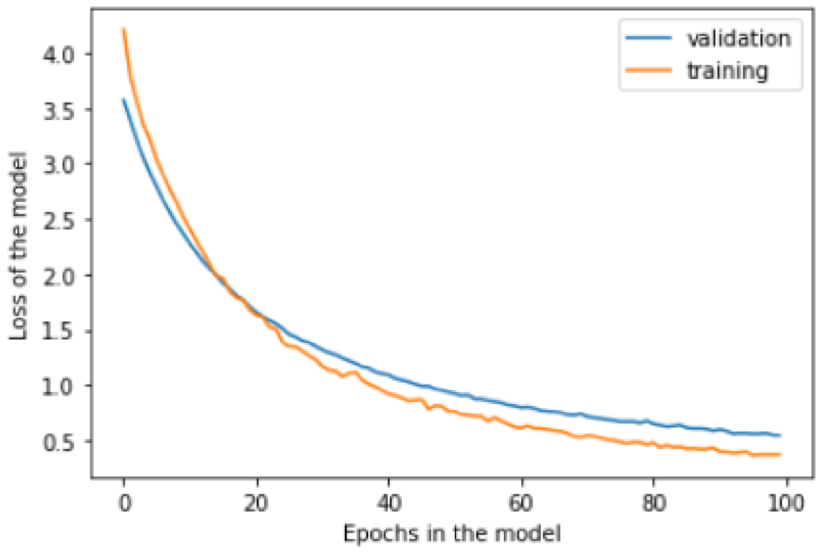
Since each image in the Olivetti dataset is a grayscale one, the first step is converting each image to a BGR one, so that the shape changes from 64x64 to 64x64x3, which aligns with the format that we require for the VGG-19 based model. Additionally, each image's label, which is simply a number between 0 to 39, is changed into its one-hot encoding form as well.

We then compile our model, which consists of the following:

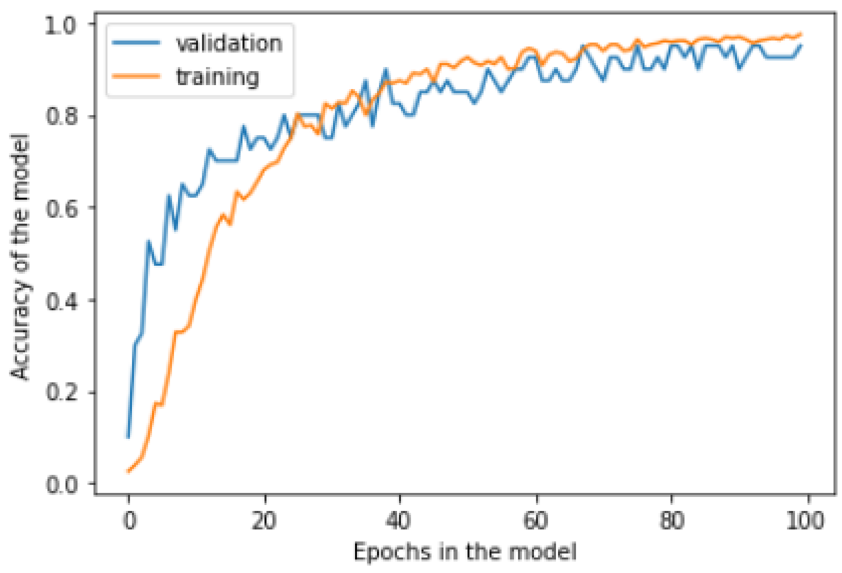
- The base, which is a VGG-19 model without the top, where each layer is not trainable
- A dropout layer, where the dropout rate is set to 0.5
- A flattened layer
- A dense layer, where the activation function is softmax, with 40 resultant class options

Since the VGG-19 model is already trained, what we are essentially doing here is transfer learning, where we take the knowledge that the model already has and then simply fine tune the top dense layer for our purpose.

The model is trained on the train set, with the test set imputed as the validation dataset. I trained the model for 100 epochs. The trends in loss and accuracy can be seen in the plots below:



Loss vs Number of Epochs



Accuracy vs Number of Epochs

As can be seen, the model seems to be doing well, not overfitting or underfitting the data. Based on this data, I came to the conclusion that the 88th epoch seems ideal, with a training accuracy of 95.83% and a validation accuracy of **95%**.

An example of this model making a prediction is:

True label and image vs the predicted label (below the print code block) →

```
0
```

```
prediction = model.predict(testimg)
print(np.argmax(prediction))
```

```
0
```

As can be seen, the model has been able to identify the person successfully, despite the image being blurred.

3. Observations and Conclusion

As can be seen, the primary goal of this project, which was to come up with a way to blur and anonymize individual faces, was successful. However, along with this, as discussed in section 1.2, an important factor to explore was how this anonymization does not happen at the cost of security. We hence experimented with the idea of recovering identities from blurred images to address this issue, and explored two possible ways to do this - K-Nearest Neighbours and VGG-19. The accuracy of the former is 90% and that of the latter is 95%. Clearly, both models have been able to perform reasonably well on the desired task, with the VGG-19 based model performing better.

However, in a legal/security situation, 95% accuracy might not be good enough. Further exploration in this space can be done with more complex deep learning models and larger datasets.

4. References

Lectures and lecture slides
https://keras.io/api/models/model_training_apis/
https://keras.io/api/applications/vgg/
https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/
https://pyimagesearch.com/2020/04/06/blur-and-anonymize-faces-with-opencv-and-python/
https://machinelearningmastery.com/transfer-learning-for-deep-learning/