CCT College Dublin

ARC (Academic Research Collection)

2023

Facial Recognition using Neural Network

Georges Diego Hawile Pinheiro
CCT College Dublin

Allison Alves de Moura
CCT College Dublin

CCT College Dublin

Assessment Cover Page

Module Title:	Problem Solving for Industry
Assessment Title:	Capstone Group Project
Lecturer Name:	Muhammad Iqbal
Student Full Name:	Georges Diego Hawile Pinheiro Alisson Alves de Moura
Student Number:	2019466 2019142
Assessment Due Date:	19th May 2023 (23:59 Irish Time)
Date of Submission:	

Table of Contents

Word Count: **4960 Words.**

Abstract:

Face recognition technology using machine learning and neural networks has become increasingly prevalent in recent years, revolutionizing various fields such as security, law enforcement, and marketing. The development of an AI-based system that can perform face recognition using these technologies has become a significant focus for researchers and developers worldwide.

This project aims to create such a system that can recognize and classify faces accurately using machine learning algorithms and neural networks. By leveraging these advanced technologies, the system can learn and improve over time, leading to higher accuracy rates and enhanced performance.

Introduction:

Face recognition technology using machine learning and neural networks has become increasingly popular over the years, particularly in security and surveillance systems. This project aims to develop an AI-based system that can perform face recognition using machine learning and neural network algorithms.

Business Understanding

Face recognition technology using machine learning and neural networks has many potential applications in various industries, including security, law enforcement, marketing, and healthcare.

"Creating and sustaining superior performance is the key to achieving competitive advantage. In the security industry" (Porter, 1985), this technology can be used to enhance the accuracy and efficiency of facial recognition systems used for access control, surveillance and identity verification. With the help of machine learning algorithms and neural networks, the system can learn and improve over time, leading to higher accuracy rates and faster processing times. This can improve security measures in places like airports, government facilities, and private organizations. In the marketing industry, this technology can be used to personalize the customer experience and improve customer engagement. For example, a retail store can use facial recognition technology to identify a customer's age, gender, and mood, and use that information to provide customized recommendations and advertisements. "This can lead to increased sales and customer satisfaction" (Grant, 1991).

In the healthcare industry, this technology can be used to improve patient care and outcomes. For example, a hospital can use facial recognition technology as a unique identifier ID; that grants access to their medical records quickly and accurately, improving the security and ensuring quality of care and by consequence reducing the risk of medical errors. Overall, organizations would be interested in such product to improve efficiency, accuracy, and convenience in various industries. The potential applications of face recognition technology using machine learning and neural networks are vast, making it an attractive investment for organizations looking to stay ahead of the competition.

"Machine Learning algorithms enable the system to learn and improve over time by using a large dataset to recognize and classify faces accurately" (K, et al., 2016). Machine learning algorithms and neural networks are the key technologies used in this project. Neural networks, on the other hand, are used to create complex models that can perform more sophisticated tasks such as emotion recognition.

Open-source technologies are preferred for this use case because they provide flexibility, customization, and community support. "They also provide access to a vast array of resources, tools, and libraries that can help developers to create efficient and reliable systems even for large-scale machine learning" (Abadi, 2016). With constant public use of open libraries, this grants the development of the tool in a organic way; also motivates most users to contribute not only financially but also with code improvement, constant feedback and the testing of the application.

Project Plan

Business Understanding
- Define the problem and identify the stakeholders
- Determine the business objectives and constraints
- Determine how the software will be used and what outcomes are desired
- Deadline 17/03/2023

Data Understanding
- Identify the data sources and collect the relevant data
- Perform exploratory data analysis (EDA) to understand the data and identify any issues
- Determine what data pre-processing and feature engineering will be necessary
- Deadline 24/03/2023

Data Preparation
- Clean and pre-process the data
- Perform feature engineering and feature selection
- Split the data into training, validation, and testing sets
- Deadline 31/03/2023

Modelling
- Choose appropriate machine learning algorithms and neural networks to use
- Train the models on the training set
- Evaluate the models using the validation set and adjust hyperparameters as necessary
- Deadline 21/04/2023

Evaluation
- Evaluate the best-performing models on the testing set
- Determine how well the software is meeting the business objectives
- Assess the model's performance against any regulatory or ethical requirements
- Deadline 28/04/2023

Deployment
- Integrate the model into the software and test it thoroughly
- Deploy the software to production and monitor its performance
- Provide documentation and training to end-users and stakeholders
- Deadline 16/05/2023

Risks and contingences

Limited availability of data

The project may be delayed or fail if there is insufficient or low-quality data to build a predictive model. Also a challenge that is constant in this Dataset type is the variations between each class can be of a big challenge to fit a model within a limited amount of examples of pictures per individual.

Contingency plan: Explore additional data sources, augment existing data with synthetic data, fine tuning of training models.

Model overfitting

The predictive model may be over fitted to the training data, resulting in poor or bias performance when training new data.

Contingency plan: Use regularization techniques, ensemble methods, or cross-validation to reduce overfitting and improve the generalization performance of the model.

Regulatory compliance issues

The project may face legal or ethical challenges related to data privacy, discrimination, or other issues, which could result in significant delays or legal barriers.

Contingency plan: Conduct regular compliance reviews, ensure all data usage is transparent and justified, and address any compliance issues immediately, contingency plan for data destruction and model retrain.

Data understanding

Dataset: DigiFace 1M
Source: https://github.com/microsoft/DigiFace1M
Format: .png files
Dimension: 112x112 pixels
Width: 112 pixels
Height: 112 pixels
Colour channels: 3 (RGB)
Bit Depth: 32

The DigiFace-1M dataset is a collection of over one million diverse synthetic face images created also by AI, it was introduced by Microsoft and can be used to train deep learning models for facial recognition.

The dataset contains 720 000 images with 10 000 identities (72 images per identity). For each identity, 4 different sets of accessories are sampled and 18 images are rendered for each set.
The dataset is represented by a root folder and ten thousand subfolders which are named from "0" to "10000" each folder represents an identity, each folder contains seventy two synthetic face images, in a real world scenario the subfolders would be renamed with a person name and that information would be used within the software when the model identify the subject.

The dataset used for our model is the part 1 that contains 2000 individuals with 72 images per identity in a total of 144000 images

Dataset EDA

Performing an exploratory data analysis it was possible to conclude that there was no "missing values" in the dataset, in this scenario missing values would be represented by missing images in the subfolders, another consideration was the format of the files in the dataset, if the files were in different formats that could disrupt the functionality of the software , the dataset had no issues in regards file format or corrupted data, the EDA was also performed looking for any inconsistencies in the data format or labelling. In the case of image datasets, it is also important to check for any duplicates or near-duplicates, as well as any images that may be corrupted or of low quality.

After further analyses on the dataset there are a few techniques being considered at this point, our research on how convolutional neural networks performs mathematical functions and read data suggests that the following techniques might be necessary in order to achieve the expected results.

Image pre-processing:

Depending on the nature of the dataset, it may be necessary to pre-process the images to improve their quality or to extract relevant features. Techniques like image resizing, normalization, and filtering can be applied to interact with image pixels information for data processing.

Feature extraction:

In some cases, it may be useful to extract features from the images, such as colour histograms, texture features, or shape features. These features can then be used as inputs to machine learning models for classification or clustering as in an example of different backgrounds, bright or dark images of the same individual and etc.

Data augmentation:

Depending on the size and diversity of the dataset, it may be useful to perform data augmentation techniques to increase the size of the dataset or to introduce additional diversity. Techniques like random cropping, flipping, and rotation can be applied to create new images from the existing dataset, which could be an interesting step to create models to recognize specific features of an individual such as smile, eyes or nose shape.

The data quality of the dataset was satisfactory enough to move to the next stage, the present data is complete and might be enough to cover most cases required for this project, in case any problems comes up during the development the contingency plan would be to apply the EDA techniques listed above.

Data Preparation

Now that we performed some EDA techniques to understand our data, the next step is to prepare the data to be used with our models, in this step it was necessary to research on how deep learning algorithms reads and process data from images, so our data could be adjusted/prepared for optimal results, our first findings was that the algorithms analyse the data from the imaged based on the values of each pixel.This research was important as we have learned that it was necessary to pre-process the images from the dataset in order to achieve better results.

The pre-processing of our images consists in two steps, in the first step we scale down our images, resizing the images to a smaller size will result in less data to be processed/analysed by the model and that would improve the speed for training the model and also how fast the model would predict and classify our images. Our dataset is composed of images in the following format: 112 pixels by 112 pixels and 3 colour channels (RGB), the visualization of our current data before pre-processing can be observed in figure 1 below:

Figure 1. Vizualization of image data

The result is a three-dimensional array where it is possible to see the values of pixels ranging from 0 to 255 representing each colour channel. The visualization of the datathat each picture holds is important so not only we can understand it but we can see ways of improving it by applying some techniques, in this scenario we believe that two techniques can result in a considerable improvement for this project, image resizing and normalization, our approach was to resize the images from 112 to 100pixels that would reduce the amount of data to be processed in about 10.71% without sacrificing too much quality of the data and that would result in a faster model and less computation power, by applying normalization technique we normalized the values stored for every pixel to be between 0 and 1 so the data spread would be contained for processing and even visualization without jeopardizing the shape, to achieve that we know that the values for every pixel range from 0 to 255 and we can normalize this values to range from 0 to 1 by dividing every value by 255, the reason we are normalizing our data is because the mathematical operations that are performed by the deep learning algorithms would not function properly if we have numbers that range from a very small number to a very large value, by normalizing the data we have this problem addressed and we could move to the modelling stage with more confidence in regards to the quality of our data.

There is one more technique that is being considered at this stage , data augmentation using synthetic data, this project aims to build a model that would be able to classify images and identify individuals, to achieve that we will test different methods with different approaches training different models, the results of our research points that we might need to apply data augmentation for a model that performs something called "one shot classification", on this specific model our dataset will be used to train the model and the data will be used as negative samples , this approach requires another 2 sets of data, anchor images samples and positive images samples, both will be introduced using data augmentation with synthetic data. To apply this technique we are planning to create a connection to a camera device and take pictures from individuals that will be classified by the model.

After those first steps in regards to the data preparation it was possible to create a labelled dataset, to achieve that, we have the images from the dataset in a folder named "negative", a portion of the images from the augmented synthetic data were stored in a folder named "positive" and the other portion was stored in a folder named "anchor", each observation of our dataset is now composed by three objects, an "anchor" image, a positive or negative image and a label, the label has a value of 1 if the second object is positive to the anchor and has a value of 0 if the object is negative to the anchor.

To reinforce the reasoning behind those choices and applied techniques there are other visualizations on the jupyter notebook that are not mentioned on this report, in case the understanding of these techniques are still unclear it is highly suggested looking into the visualizations and comments on the project files as it will make it a lot easier to understand each step.

After we structured this dataset we had to shuffle the samples so when we start training the model we have a fairly mixed set of samples, our training/test partition were split in a ratio of 70/30 in this approach.

Modelling

"The field of machine learning has taken a dramatic twist in recent times, with the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the performance of previous forms of artificial intelligence in common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offers a simplified method of getting started with ANNs."(K O'Shea, 2015)

These computational models inspired on known biological architecture are able to best performance of previous forms of artificial intelligence in other common machine learning tasks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNNs are mainly the tool used to solve difficult image-driven pattern recognition tasks and with their precise yet simple mathematical architecture, it offers a simplified method of getting started with ANNs.
Convolutional Neural Networks (CNNs) are analogous to traditional ANNs in that they are comprised of neurons that self-optimise through learning. Each neuron will still receive an input and perform an operation (such as a scalar product followed by a non-linear function) - the basis of countless ANNs. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight).

The last layer will contain loss functions associated with the classes, and all of the regular tips and tricks developed for traditional ANNs still apply. The only notable difference between CNNs and traditional ANNs is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model.

One of the largest limitations of traditional forms of ANN is that they tend to struggle with the computational complexity required to compute image data. Common machine learning benchmarking datasets such as the MNIST database of handwritten digits are suitable for most forms of ANN, due to its relatively small image dimensionality of just 28 × 28. With this dataset a single neuron in the first hidden layer will contain 784 weights (28×28×1 where 1 bear in mind that MNIST is normalised to just black and white values), which is manageable for most forms of ANN.

If you consider a more substantial coloured image input of 64 × 64, the number of weights on just a single neuron of the first layer increases substantially to 12, 288. Also take into account that to deal with this scale of input, the network will also need to be a lot larger than one used to classify colour-normalised MNIST digits, then you will understand the drawbacks of using such models.

The starting point for one of our models was based on a convolutional neural network architecture published by the Department of Computer Science, University of Toronto (Koch Gregory, 2020). The architecture describes a convolutional neural network designed for a one shot image recognition.

One-shot learning can be directly addressed by developing domain-specific features or inference procedures which possess highly discriminative properties for the target task. As a result, systems which incorporate these methods tend to excel at similar instances but fail to offer robust solutions that may be applied to other types of problems.

To develop a model for one-shot image classification, we aim to first learn a neural network that can discriminate between the class-identity of image pairs, which is the standard verification task for image recognition. We hypothesize that networks which do well at verificationshould generalize to one-shot classification. The verification model learns to identify input pairs according to the probability that they belong to the same class or different classes. This model can then be used to evaluate new images, in a pairwise manner against the test image. The pairing with the highest score according to the verification network is then awarded the highest probability for the one-shot task.

Figure 1. Model overview

```
Found 116000 images belonging to 2000 classes.
Found 28000 images belonging to 2000 classes.
```

```python
[*]: model1 = Sequential()

     # add convolutional layer with 32 filters and 3x3 kernel size, basic activation and image size and RGB
     model1.add(Conv2D(32, (7, 7), activation='relu', input_shape=(112, 112, 3)))

     # add max pooling layer with 2x2 pool size - complexity of the neural network
     model1.add(MaxPooling2D((2, 2)))

     # add convolutional layer with 64 filters and 3x3 kernel size
     model1.add(Conv2D(64, (3, 3), activation='relu'))

     # add max pooling layer with 2x2 pool size
     model1.add(MaxPooling2D((2, 2)))

     # add convolutional layer with 128 filters and 3x3 kernel size
     model1.add(Conv2D(128, (3, 3), activation='relu'))

     # add max pooling layer with 2x2 pool size
     model1.add(MaxPooling2D((2, 2)))

     # add flatten layer to convert output to 1D
     model1.add(Flatten())

     # add dense layer with 256 units
     model1.add(Dense(256, activation='relu'))

     # add output layer with 2000 units for multi-class classification
     model1.add(Dense(2000, activation='softmax'))

     # compile the model
     model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

     model1.fit(train_generator, epochs=100, validation_data=validation_generator)
```

In Figure 7 the test model creates a CNN layer with 32 Filters that will have a 7x7 kernel size that has the interest of reducing the image sharpness with the activation in "relu" (Rectified Linear Unit). This has an intrinsic weakness: blocking inputs that are smaller than zero. Once this was mitigate in the data preparation this function will be able to do its strengths that is resisting against the vanishing gradient problem that squeezes information until there is no learning that other functions would have.

The Max Pooling layer will do two important tasks: reducing the resolution of the given output from last layer will reduce computational power needed and help to reduce over fitting by eliminating values that are not important and could cause noise in the analysis,; the bigger the parameter are, more reducing happens; this is the smallest value possible before this wouldn't do anything.

This process is done three times, with different parameters being reduced and enhanced by the filters from CNNs increasing and the Max pooling enhanced to 3x3 on the second two layers. After these filters we would expect to have layers with multiple filters sizes detecting patterns and the most important pixel values highlighted by the max pooling.

The Flattening function will prepare the learnt features for the dense layers that work with vectors. The two dense layer will perform the classification firstly use the Relu activation again with a size of 256 and at last the Softmax with size of the number of classes we have that will use the values for all classes and returns the best result for all of them as a classification.

Evaluation

To evaluate the Siamese CNN two evaluation metrics were considered:

Precision: Precision is the fraction of true positives (TP) among all instances that are predicted as positive (TP + FP). In other words, precision measures how many of the predicted positive instances are actually positive. A high precision means that the model is accurate in identifying positive instances.

Precision = TP / (TP + FP)
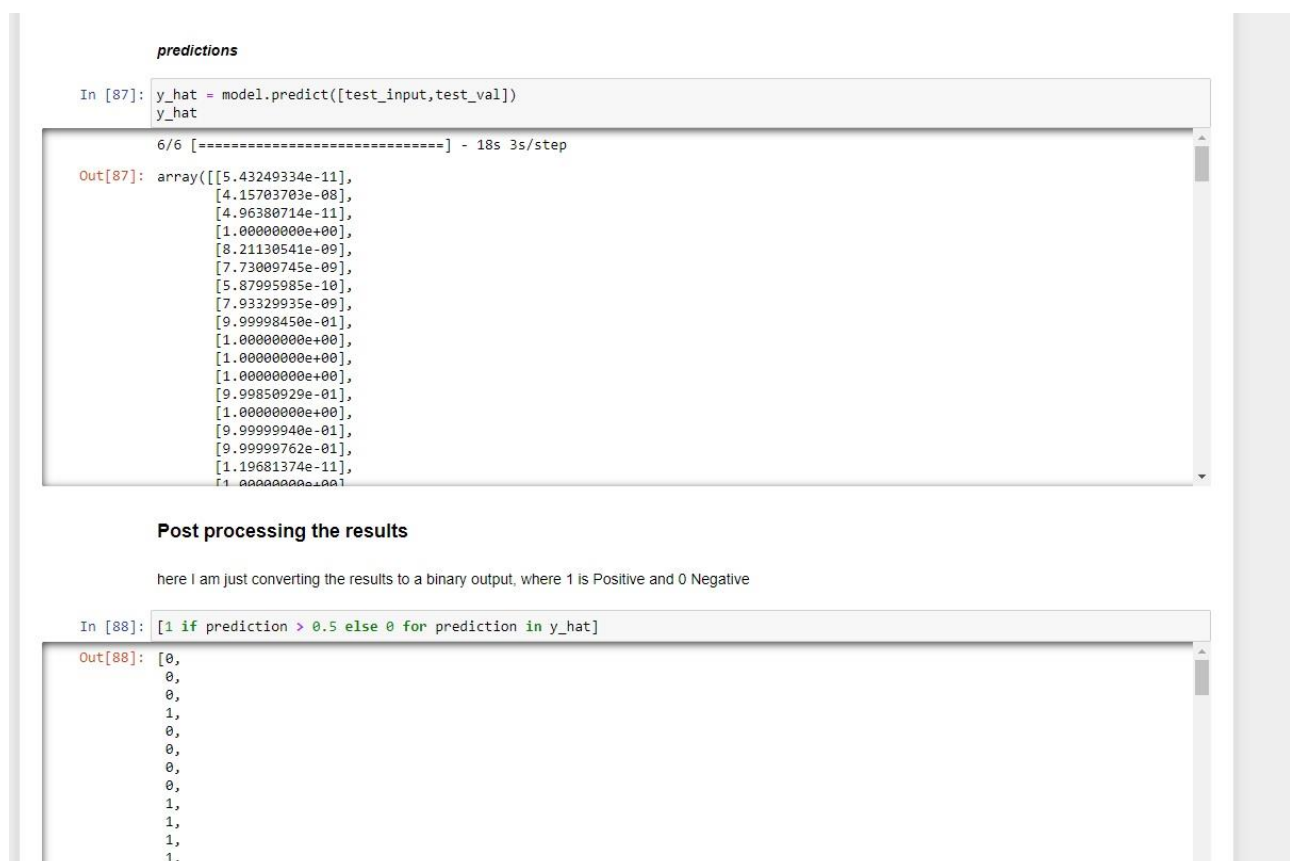
Recall: Recall is the fraction of true positives (TP) among all instances that are actually positive (TP + FN). In other words, recall measures how many of the actual positive instances are identified by the model. A high recall means that the model is able to identify a high proportion of positive instances.

Recall = TP / (TP + FN)

In summary, precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions. A good model should have a balance between high precision and high recall, as maximizing one metric may negatively impact the other.

The first step evaluating the model was preparing the validation set, then we used the model to make predictions on the validation set, finally we converted the values from our prediction to a binary output as demonstrate in figure 2.

Figure 2. Post-process of prediction values to binary output

With the results of our predictions it was possible to calculate the Recall and Precision as it can be observed in Figure3:

Figure 3. Visualization of metrics and confirming a negative result


```
In [89]:  #Creating a metric object
          m = Recall()
          # Calculating the recall value
          m.update_state(y_true, y_hat)

          # Return result
          m.result().numpy()
Out[89]:  1.0

In [90]:  #Creating a metric object
          m = Precision()
          # Calculating the recall value
          m.update_state(y_true, y_hat)

          # Return result
          m.result().numpy()
Out[90]:  0.9885057
```

Vizualize Results

```
In [92]:  plt.figure(figsize=(6,3))
          plt.subplot(1,2,1)
          plt.imshow(test_input[0])
          plt.subplot(1,2,2)
          plt.imshow(test_val[0])
          plt.show()
```
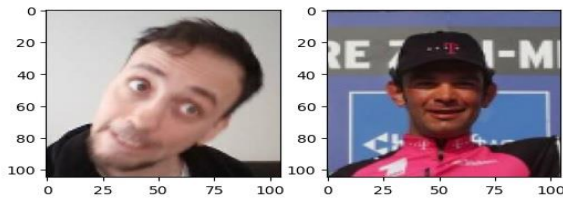

The visualization of the results help us to confirm if the model is working as expected and if the model is capable of predict correctly, remember that 0 represents a negative result where the identity in the first and second images are not the same person. Getting the index 3 from our result array (figure 2) should return a positive result where the two person in both images is the same identity, this can be observed in figure 4:

Figure 4: Visualization of a positive result.

Vizualize Results

```
In [93]:  plt.figure(figsize=(6,3))
          plt.subplot(1,2,1)
          plt.imshow(test_input[3])
          plt.subplot(1,2,2)
          plt.imshow(test_val[3])
          plt.show()
```
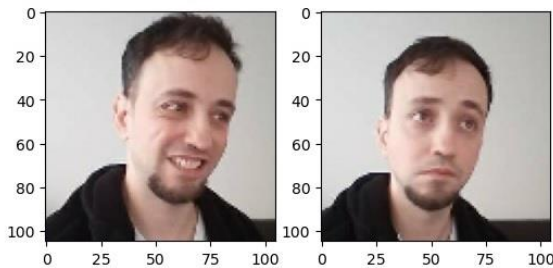
High class Volume

The dataset from Digiface creates a challenge for the data classification once each new class added to the model will decrease the accuracy and increase the processing of the images.
Some Pre-processing was used to reduce this computational power and increase the amount of tests with the dataset.

When testing a small share of the dataset for only two folders, the accuracy trained with only 30 epochs resulted in a satisfactory result of 96% and some even of 100%

Figure 5. Training results

```
Epoch 28/30
4/4 [==============================] - 7s 2s/step - loss: 9.5293e-05 - accuracy: 1.0000 - val_loss: 0.0339 - val_accuracy: 1.00
00
Epoch 29/30
4/4 [==============================] - 8s 2s/step - loss: 7.4293e-05 - accuracy: 1.0000 - val_loss: 0.0380 - val_accuracy: 0.96
43
Epoch 30/30
4/4 [==============================] - 8s 2s/step - loss: 6.9674e-05 - accuracy: 1.0000 - val_loss: 0.0426 - val_accuracy: 0.96
43
```

This shows the validation accuracy can be considered valid due to the Loss being extremely low. Meaning the Dataset images even with big variations can be used to train a model.

The challenge for multiple individuals is doing the choice between time for training the models and the parameters to change to achieve a fast processing of the batches and a good accuracy at the same time.

Once more individuals or classes are added to the model, this increases exponentially the model processing and decreases in the same order the accuracy of the model trained. Also Its possible to see the training of the model suffers with this and cannot be accepted as valid due to the loss value that also increases accordingly.

Figure 6. Training evaluation

```
Epoch 26/100
23200/23200 [==============================] - 3839s 165ms/step - loss: 7.6113 - accuracy: 3.2759e-04 - val_loss: 7.6021 - val_
accuracy: 5.0000e-04
Epoch 27/100
23200/23200 [==============================] - 3905s 168ms/step - loss: 7.6114 - accuracy: 2.7586e-04 - val_loss: 7.6019 - val_
accuracy: 5.0000e-04
Epoch 28/100
23200/23200 [==============================] - 3860s 166ms/step - loss: 7.6114 - accuracy: 3.2759e-04 - val_loss: 7.6019 - val_
accuracy: 5.0000e-04
Epoch 29/100
23200/23200 [==============================] - 4976s 214ms/step - loss: 7.6112 - accuracy: 2.3276e-04 - val_loss: 7.6019 - val_
accuracy: 5.0000e-04
Epoch 30/100
23200/23200 [==============================] - 6469s 279ms/step - loss: 7.6113 - accuracy: 3.1034e-04 - val_loss: 7.6019 - val_
accuracy: 5.0000e-04
Epoch 31/100
23200/23200 [==============================] - 4163s 179ms/step - loss: 7.6113 - accuracy: 3.3621e-04 - val_loss: 7.6019 - val_
accuracy: 5.0000e-04
Epoch 32/100
23200/23200 [==============================] - 4484s 193ms/step - loss: 7.6113 - accuracy: 2.5000e-04 - val_loss: 7.6020 - val_
accuracy: 5.0000e-04
Epoch 33/100
23200/23200 [==============================] - 3866s 167ms/step - loss: 7.6113 - accuracy: 3.1034e-04 - val_loss: 7.6020 - val_
accuracy: 5.0000e-04
Epoch 34/100
10484/23200 [===========>..................] - ETA: 29:33 - loss: 7.6077 - accuracy: 4.0061e-04
```

As it is possible to see in figure 6, after more than 30 epochs it is possible to see the model is acquiring some improvement in the dataset processing as per the loss value average showing a shy descent and the accuracy showing a slow average climbing. As all these four important numbers are interdependent and also due to the fact the model algorithm is trained to tune some internal parameters to acquire a better result in average for all these features, if the observer focus only on one of them it is harder to see the improvement of the result parameters.

The fine tuning of this model is crucial to find the best result for this dataset, it is possible to see somedifferences in results depending on the main parameters chosen for the test above.

One way to improve accuracy is by training multiple models and combining their predictions. This technique is known as ensemble learning and can lead to better results than a single model.

So I created a Test dataset with 21 individuals to test the accuracy achieved with the same model.

Also divided in 80 for training and 20% for Validation.

Figure 8. Defining data split, size and batch size.

```
: train_generator = datagen.flow_from_directory(
      dataset_path,target_size=(112, 112),batch_size=5,class_mode='categorical',subset='training')

  validation_generator = datagen.flow_from_directory(
      dataset_path, target_size=(112, 112), batch_size=5, class_mode='categorical',subset='validation')

  Found 1218 images belonging to 21 classes.
  Found 294 images belonging to 21 classes.
```

Figure 9. Training of model before Fine Tuning

```
In [8]: model.fit(train_generator, epochs=10, validation_data=validation_generator)

        Epoch 1/10
        244/244 [==============================] - 59s 233ms/step - loss: 3.0504 - accuracy: 0.0320 - val_loss: 3.0445 - val_accuracy:
        0.0476
        Epoch 2/10
        244/244 [==============================] - 52s 212ms/step - loss: 3.0537 - accuracy: 0.0443 - val_loss: 3.0453 - val_accuracy:
        0.0476
        Epoch 3/10
        244/244 [==============================] - 49s 200ms/step - loss: 3.0477 - accuracy: 0.0427 - val_loss: 3.0445 - val_accuracy:
        0.0476
        Epoch 4/10
        244/244 [==============================] - 56s 229ms/step - loss: 3.0469 - accuracy: 0.0378 - val_loss: 3.0448 - val_accuracy:
        0.0476
        Epoch 5/10
        244/244 [==============================] - 56s 231ms/step - loss: 3.0447 - accuracy: 0.0287 - val_loss: 3.0751 - val_accuracy:
        0.0612
        Epoch 6/10
        244/244 [==============================] - 51s 207ms/step - loss: 2.9660 - accuracy: 0.1182 - val_loss: 2.8737 - val_accuracy:
        0.1463
        Epoch 7/10
        244/244 [==============================] - 54s 220ms/step - loss: 2.0053 - accuracy: 0.4245 - val_loss: 2.1517 - val_accuracy:
        0.3946
        Epoch 8/10
        244/244 [==============================] - 56s 230ms/step - loss: 0.9128 - accuracy: 0.7266 - val_loss: 2.0408 - val_accuracy:
        0.5510
        Epoch 9/10
        244/244 [==============================] - 56s 228ms/step - loss: 0.3304 - accuracy: 0.8941 - val_loss: 2.0522 - val_accuracy:
        0.5884
        Epoch 10/10
        244/244 [==============================] - 56s 229ms/step - loss: 0.0907 - accuracy: 0.9680 - val_loss: 2.9690 - val_accuracy:
        0.5476

Out[8]: <keras.callbacks.History at 0x21628f5f640>
```

There was improvement in both training and validation accuracies, Training accuracy is already acceptable due to its loss value being so small and the accuracy being of over 90%.

The validation on the other hand was of 59% at best, which is no ideal for individual recognition and to apply the method of multiple models.
The problem could be perhaps solved with data augmentation, so an idea was to divide the dataset in a different ratio between training and testing.

This approach created an accuracy of 100% from epoch 11 and the accuracy improved to over 60%.
The loss value was extremely small for training, but the loss for the validation was over 2.0. Meaning it was not only inaccurate but not very trustworthy.

Figure 10. Prediction visualization

```
In [46]: from tensorflow.keras.preprocessing import image
         import numpy as np

         # Load the model
         model = load_model(r'C:\Users\aliss\Documents\last Semester\Industry Project\dataset\Digiface\Models\batch_1.h5')

         # Load the image
         img = image.load_img(r'C:\Users\aliss\Documents\last Semester\Industry Project\dataset\Digiface\framework validation\0.png', targ

         # Convert the image to a numpy array
         x = image.img_to_array(img)

         # Expand the dimensions of the image array to match the input shape of the model
         x = np.expand_dims(x, axis=0)

         # Normalize the image data to values between 0 and 1
         x = x/255.0

         # Make a prediction on the image using the model
         preds = model.predict(x)

         # Print the prediction
         print(preds)
```

```
1/1 [==============================] - 0s 82ms/step
[[9.87707913e-01 3.92032098e-06 3.60199266e-08 9.21862586e-09
  9.88033789e-05 1.01501293e-07 5.42127382e-05 3.23755273e-10
  1.12770041e-02 1.15930035e-11 3.79857171e-04 1.17611991e-04
  3.13443110e-07 1.61204539e-11 1.98927773e-06 5.72270874e-05
  2.79186643e-04 4.99610908e-10 2.10470480e-05 6.40810924e-07]]
```

With this result without labels it was possible to see the difference in the magnitude of the results in the index of values for each class in the model. We know this model has the individual number 0 trained into it for position 0 in the result array.

As we can observe in Figure 10, the results not only shows a high accuracy value in the position 0, but we can also observe that the position 0 was the first model tested which corresponds to this identity.

Observing the other values, it is possible to verify that not only we can determine that an identity is contained in the dataset, but also that the higher value is also significantly higher than the other values, even if this value does not represents a very high accuracy.

We can then conclude that, knowing that an identity is contained in the dataset, it will be possible to identify that individual to the detriment of options in the dataset.

Another point to consider was the idea of splitting the models, by doing so facilitate the "filtering" of identities, as the model that does not corresponds to that particular identity will return extremely low values up to seven orders of magnitude (10 to the power of minus seven) or 10^{-7} . These results compared to a filtering equivalent to minimum of 5% are sufficient enough to identify individuals with similar characteristics for further analysis.

Figure 11. Results after Fine Tuning

```
model.fit(train_generator, epochs=10, validation_data=validation_generator)

Epoch 1/10
215/215 [==============================] - 52s 239ms/step - loss: 3.0506 - accuracy: 0.0850 - val_loss: 2.8603 - val_accuracy:
0.1451
Epoch 2/10
215/215 [==============================] - 52s 243ms/step - loss: 2.0591 - accuracy: 0.4108 - val_loss: 1.7899 - val_accuracy:
0.4853
Epoch 3/10
215/215 [==============================] - 52s 243ms/step - loss: 0.5539 - accuracy: 0.8375 - val_loss: 1.4999 - val_accuracy:
0.6077
Epoch 4/10
215/215 [==============================] - 52s 241ms/step - loss: 0.1508 - accuracy: 0.9570 - val_loss: 1.8782 - val_accuracy:
0.6077
Epoch 5/10
215/215 [==============================] - 53s 248ms/step - loss: 0.0598 - accuracy: 0.9860 - val_loss: 1.8043 - val_accuracy:
0.5986
Epoch 6/10
215/215 [==============================] - 52s 244ms/step - loss: 0.0137 - accuracy: 0.9972 - val_loss: 1.8454 - val_accuracy:
0.6417
Epoch 7/10
215/215 [==============================] - 69s 319ms/step - loss: 0.0032 - accuracy: 0.9991 - val_loss: 1.8878 - val_accuracy:
0.6689
Epoch 8/10
215/215 [==============================] - 56s 262ms/step - loss: 0.0015 - accuracy: 0.9991 - val_loss: 1.9435 - val_accuracy:
0.6553
Epoch 9/10
215/215 [==============================] - 59s 273ms/step - loss: 0.0037 - accuracy: 0.9991 - val_loss: 1.9130 - val_accuracy:
0.6667
Epoch 10/10
215/215 [==============================] - 59s 276ms/step - loss: 2.6033e-04 - accuracy: 1.0000 - val_loss: 1.9818 - val_accura
cy: 0.6689

<keras.callbacks.History at 0x2162a4ac880>
```

The model was retrained several times to achieve the best combination between loss values and accuracy, at the same moment as trying to avoid overtraining.

After only 10 epochs, the model was trained with accuracy of 100% and the validation number improved to 67% with a loss number under 2.0.

After testing parameters that presents good progress within 10 epochs, I've trained the best model for 100 epochs to see the ideal number of epochs before the model gets over trained, it can be observed in figure 11.

The validation accuracy will at some point reach the value of 70% with a much smaller loss function value between epochs 8 and 12. Therefore I will work with a number of epochs of 10 for this model type for Ensemble learning.

Conclusion

We have presented a strategy for performing one-shot classification by convolutional neural networks for verification.

We demonstrated that the strong performance of these networks on this project indicate that high level of accuracy is possible with our approach, and that this approach should extend to one-shot learning tasks in other domains, especially for image classification.

Considering the comprehensive analysis conducted in this report, numerous noteworthy findings and insights have emerged concerning the application of Convolutional Neural Networks (CNNs) in image processing. The gathered data and information offer valuable perspectives on the effective processing and comprehension of image data.

It is evident that the utilization of Ensemble Learning as an approach for data filtering showcases promising outcomes even with limited training examples. This particular discovery underscores the significance of employing multiple models to accomplish a given task and highlights the necessity for further exploration and investigation.

Furthermore, the analysis indicates the existence of an optimal threshold for training data in each individual model. This finding underscores the challenge of striking a balance between high-definition or large-scale datasets and the associated training time, thereby suggesting potential avenues for enhanced understanding and the application of multiple model techniques such as Ensemble Learning.

Additionally, the report underscores the observed tendency of trained classes to consistently perform well within their respective datasets. This insight holds significant relevance as it serves as the primary reason why the framework effectively operates as a filter, thereby presenting potential opportunities for further advancements in techniques that incorporate multiple models for data filtering as an additional step in the data preparation phase.

Overall, based on the profound insights and findings derived from this extensive analysis, it becomes apparent that achieving high accuracy across a substantial number of classes without adequate data preparation remains a formidable challenge. Ensemble Learning serves as a crucial step in class filtering, albeit necessitating careful tuning for each dataset, and merits continued research and exploration. Furthermore, the complexity and potential for false positives can be mitigated by monitoring the loss value during dataset validation throughout the training process. These conclusions firmly emphasize the continuous improvement of data preparation techniques in the context of AI-driven image analysis.

To capitalize on the identified opportunities and address the challenges at hand, it is recommended to further train models based on the top-performing results, while ensuring accuracy post-filtering. Such an approach will facilitate the development of robust libraries for commercial applications in security and crowd analysis. These recommendations are firmly supported by the compelling results presented in this report.

Bibliography

Porter, M., 1985. Competitive Advantage: Creating and Sustaining Superior Performance. New York: Free Press.

Grant, R. M., 1991. The resource-based theory of competitive advantage: implications for strategy. s.l.:California Management Review.

K, H., X, Z., S, R. & J, S., 2016. Deep Residual Learning for Image Recognition. Proceedings of the IEEE

Conference on Computer Vision and Pattern, pp. 770-778.

K O'Shea, R. N., 2015. *An Introduction to Convolutional Neural Networks,* Aberystwyth: Department of Computer Science.

Koch Gregory, Z. R. S. R., 2020. *cs.cmu.edu.* [Online]
Available at: https://www.cs.cmu.edu/
[Accessed 11 04 2023].