Finger vein biometric identification using CNN

Author: Chandrakanth[UID: 118259949]

ABSTRACT

Using biometrics for identification is the most popular and reliable way to authenticate and identify individuals. The peculiarity with biometric data is that it must be recognizable, verifiable, and unique to each person and therefore, fingerprints are the most used biometric data for verification and authentication. A novel approach of this project is to make a slight departure from the kind of data that is used to make the identification which is the thumb impressions. Using thumb impressions as biometric data has a disadvantage where, the impressions can be tampered with, which might compromise the authenticity of the identification process. It is proposed to use the infrared images of the fingers. Given the current norm due to the pandemic, people are forced to maintain social distancing, the thought of taking the thumb impressions would mean that the hygiene is a bit compromised. Since biometrics are used almost everywhere from the phone, we use to fingerprint authentication methods used by investigative agencies. To overcome this problem, a novel solution is proposed where, instead of using fingerprints as biometric data, infrared images of the veins running inside the fingers are captured and these images are unique to everyone. Taking leverage of this, a CNN(Convolutional Neural Network) model is proposed to be build and trained to correctly classify individuals. The main advantage of this method is that this is a non-contact extraction of data and more authentic as this data cannot be tampered as compared to fingerprint data.

INTRODUCTION

Using biometrics is the most used approach to authenticate and identify individuals. The biometric data can be of various forms such as a photo of the face, record of an individual's voice, or an image of an individual's fingerprint. Since biometric data is unique to everyone, which is a key requirement for authentication and identification. The main goal of this project is to come up with a better way to identify people, using infrared images of the finger veins. Since this is non-contact way of authentication and identification, this seems prospective in the future. A CNN model is built to correctly identify 123 volunteers. The model is first built with a normal architecture and the accuracies of the train, and the validation data are plotted. After that, different models were designed each one with a different value of hyperparameters, like the learning rate, number of hidden layers and their effect on the model was examined and the accuracies are plotted. The final model gave a test accuracy of 80.4% which is a good sign but it can be better with effective hyperparameter tuning. Using keras tuner which is kind of an automatic hyperparameter tuner is something that is left for future work.

RELATED WORK

Since this is a novel concept, research still needs to be done in this regard. However, in [1], In this paper, a new approach of multimodal finger biometrics based on the fusion of finger vein

and finger geometry recognition is presented. In the proposed method, Band Limited Phase Only Correlation (BLPOC) is utilized to measure the similarity of finger vein images. As for finger geometry recognition, a new type of geometrical features called Width-Centroid Contour Distance (WCCD) is proposed. This WCCD combines the finger width with Centroid Contour Distance (CCD). As compared with the single type of feature, the fusion of W and CCD can improve the accuracy of finger geometry recognition. Finally, the finger vein and finger geometry recognitions were integrated by a score-level fusion method based on the weighted SUM rule.
In [2], an improved deep network was employed, named Merge Convolutional Neural Network (Merge CNN), which uses several CNNs with short paths. The scheme is based on the use of multiple identical CNNs with different input images qualities, and the unification of their outputs into a single layer.

I drew inspiration by reading these two papers and the novelty of the concept made me feel excited to work on this data and to examine and build a good model and also to understand the effect of hyperparameters on the model.

DATA

- There is no readily available database for finger vein recognition. I have found after extensive search that the dataset was available with Dr. Bakhtiar Affendi Rosdi who is teaching at the Universiti Sains Malaysia and upon request to him, he gladly accepted my request and shared the data with me.
- The data was collected at their university using the staff of the university who volunteered. The database consists of infrared images of finger vein and vein geometry. It can be used to verify either unimodal biometrics (finger vein and finger geometry) or bimodal biometrics (fusion of vein and geometry) systems.
- The images in the database were collected from 123 volunteers comprising of 83 males and 40 females, who were staff and students of Universiti Sains Malaysia.
- Every subject provided four fingers: left index, left middle, right index and right middle fingers resulting in a total of 492 finger classes obtained. The captured finger images provided two important features: the geometry and the vein pattern. Each finger was captured six times in one session, and everyone participated in two sessions, separated by more than two weeks' time.
-
- The spatial and depth resolution of the captured finger images were 640 x 480 and 256 grey levels, respectively

The data was prepossessed as follows:

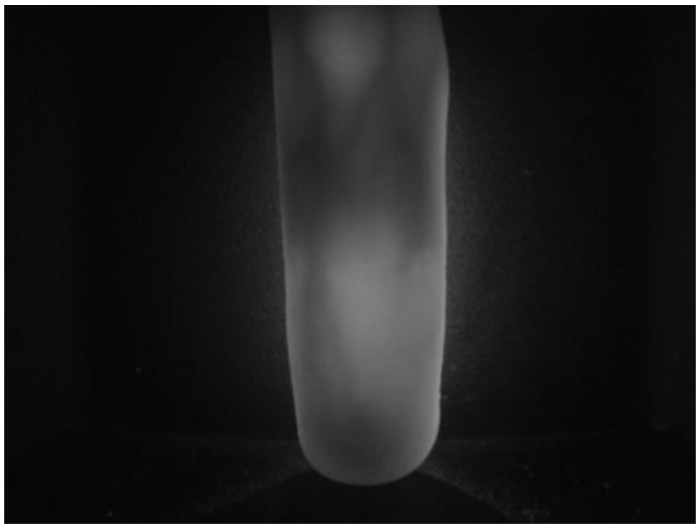
- For each image stored in the 1st session, the features for the underlying image are stored in the class corresponding to the individual's id for instance "vein001_1" for fingerprints corresponding to user ID 1. A similar approach was used for fingers from the second session after which the data was merged before splitting into train and test sets using a 80:20 ratio.

- Validation split was done using the keras framework and ImageDataGenerator. The data generator approach that was used to generate features of the images is generally useful in generating features and the corresponding labels (target). This augmentation was done using ImageDataGenerator, through which I also split the training data into validation set by using 30% of the train data.

- The original images were taken with size 640 by 480. During the generation of the features from the images, the sizes were set to 100 by 100 for the extracted veins and filled with bilinear extrapolation incase the image features had smaller dimensions than the input dimensions.

Sample of the data:

(Extracted vein)



Infrared image of the finger vein (raw data)

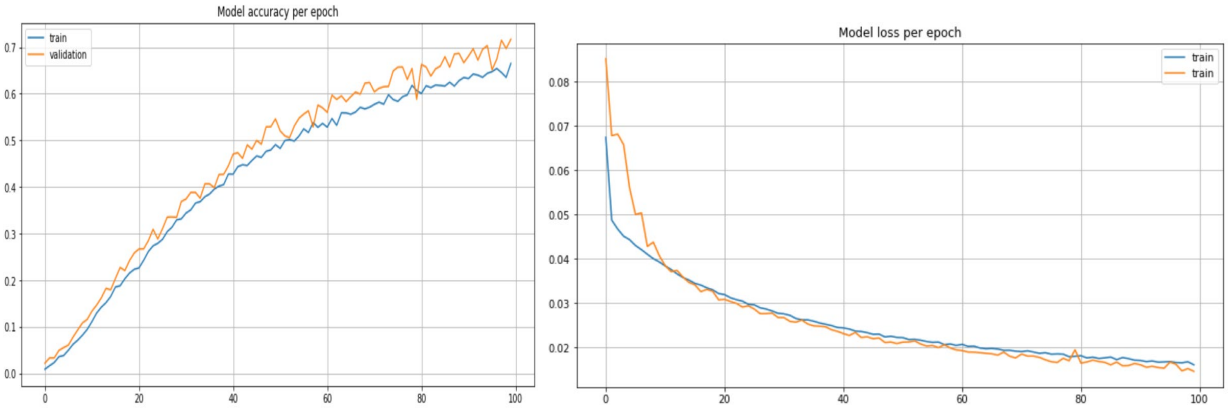
METHODS AND EXPERIMENTS

Since this is a novel concept, there isn't much research done in this field and only few papers were published. This made me use my skills to experiment and try to come up with my own methodology and skills that I had learnt in the class.
The main goal of the project is to correctly identify individuals using biometric data (which are the infrared images of the finger veins). While the paper that I read and got inspiration to do this project used a different approach in which the model was trained to correctly identify the fingers of everyone, I decided to train a model which can correctly identify only the individuals based on the infrared images of the vein.
First, I have preprocessed the image dataset by combining the extracted vein dataset and raw data (which is the finger vein images) and then assign the correct label of the individual.
I believed that the right CNN model with the right architecture is one experiment away from getting the right hyperparameters to achieve the highest classification accuracy. Therefore, my goal is to build the best CNN model with the right architecture. I tried to come up with different models with different architectures while varying different hyperparameters to minimize loss and maximize accuracy.
I have done quite a few experiments and, in the process, learned lot of things about the models that were built. Finally, I tried to play around with keras tuner which is kind of like an automatic hyperparameter tuner to give me the best model under the given conditions.

Build a sequential CNN model with the following architecture:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 100, 100, 32)      160

 max_pooling2d (MaxPooling2D  (None, 50, 50, 32)       0
 )

 dropout (Dropout)           (None, 50, 50, 32)        0

 conv2d_1 (Conv2D)           (None, 50, 50, 64)        51264

 max_pooling2d_1 (MaxPooling  (None, 25, 25, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 25, 25, 64)        0

 flatten (Flatten)           (None, 40000)             0

 dense (Dense)               (None, 256)               10240256

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 124)               31868

=================================================================
Total params: 10,323,548
Trainable params: 10,323,548
Non-trainable params: 0
```

The size of the filters and the effect of max pooling and drop out were carefully considered and chosen and the model trained and generalized well.
The following plot is the training and the validation accuracy against the number of epochs:



The training accuracy for this model is 66.51% and the validation accuracy came out to be 71.69%, which meant that there is no overfitting.
The model is tested on the testing set and is saved to avoid training later. The testing accuracy came out to be 71.7%.

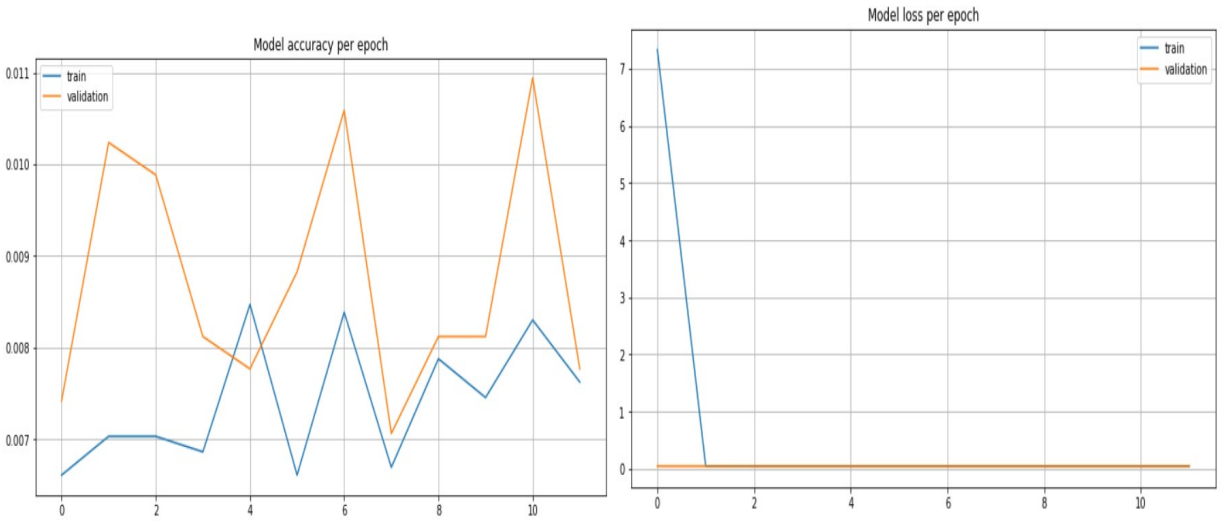
To see how different hyperparameters affect the accuracy of the model, I decided the following experiments to come up with different models:

- **Experiment Model 1 – Same architecture but increase the Learning rate from 0.01 to 0.1.**

Architecture:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 100, 100, 32)      320

max_pooling2d_2 (MaxPooling  (None, 50, 50, 32)       0
2D)

dropout_3 (Dropout)          (None, 50, 50, 32)        0

conv2d_3 (Conv2D)            (None, 50, 50, 64)        51264

max_pooling2d_3 (MaxPooling  (None, 25, 25, 64)       0
2D)

dropout_4 (Dropout)          (None, 25, 25, 64)        0

flatten_1 (Flatten)          (None, 40000)             0

dense_2 (Dense)              (None, 256)               10240256

dropout_5 (Dropout)          (None, 256)               0

dense_3 (Dense)              (None, 124)               31868

=================================================================
Total params: 10,323,708
Trainable params: 10,323,708
Non-trainable params: 0
```

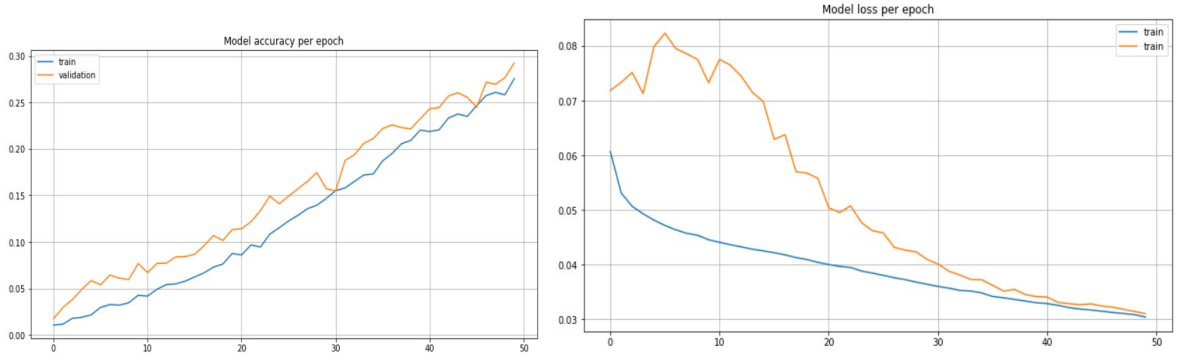
The following plot is the training and the validation accuracy against the number of epochs for the new learning rate:



As I have used early stopping to avoid the model from overfitting, I have set a patience of 5 such that the training stops if the performance does not improve after 5 epochs. From the plots, the model experienced early stopping at the 12th epoch as the validation loss was failing to improve. Learning rate of 0.1 is not such a good idea as there are lot of fluctuations and the curves are not smooth.

- **Experiment Model 2 – Same architecture but decrease the Learning rate from 0.01 to 0.001**

The architecture is the same as model1.



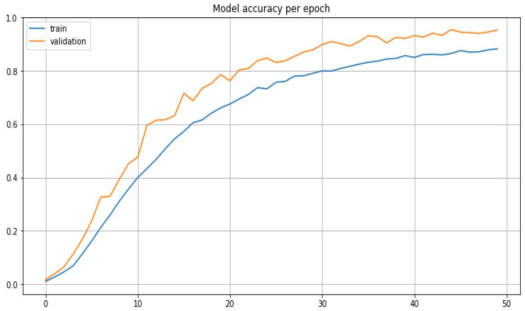
Since the learning rate is very less, it takes a lot of time to train the model and the number of epochs in this case was reduced to 50 instead of 100. Though the learning rate was low, I found that the 50 epochs were not enough for the validation accuracy to reach the maximum. It is important to note that if the learning rate is being reduced, the number of epochs must be increased too.

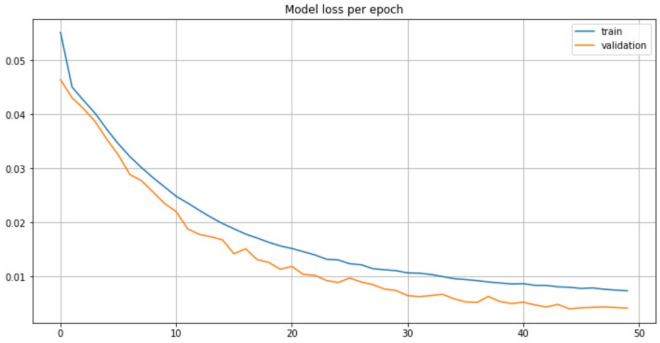
- **Experiment Model 3 – Increasing the number of hidden layers**

More emphasis was laid on increasing the number of hidden layers, as I felt that decreasing the hidden layers more would handicap the model and would not be a good model. So, I tried to increase the number of hidden layers with the following architecture.

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 100, 100, 32)      320

 max_pooling2d_8 (MaxPooling  (None, 50, 50, 32)       0
 2D)

 dropout_12 (Dropout)        (None, 50, 50, 32)        0

 conv2d_9 (Conv2D)           (None, 50, 50, 64)        51264

 max_pooling2d_9 (MaxPooling  (None, 25, 25, 64)       0
 2D)

 dropout_13 (Dropout)        (None, 25, 25, 64)        0

 conv2d_10 (Conv2D)          (None, 25, 25, 128)       204928

 max_pooling2d_10 (MaxPoolin  (None, 12, 12, 128)      0
 g2D)

 dropout_14 (Dropout)        (None, 12, 12, 128)       0

 dense_8 (Dense)             (None, 12, 12, 256)       33024

 dropout_15 (Dropout)        (None, 12, 12, 256)       0

 flatten_4 (Flatten)         (None, 36864)             0

 dense_9 (Dense)             (None, 512)               18874880

 dropout_16 (Dropout)        (None, 512)               0

 dense_10 (Dense)            (None, 124)               63612

_________________________________________________________________
Total params: 19,228,028
Trainable params: 19,228,028
Non-trainable params: 0
```

Since the number of parameters are substantially increased as the number of hidden layers is increased, the time to train the model is also increased substantially. However, the model performed well on the validation data. Increasing the number of hidden layers improves the model from the plots.





Evaluation metrics and general reflection:

The CNN model returns a classification accuracy of approximately 80.44% on the test data which implies that the model on average makes approximately 80 correct predictions for every 100 individuals which is generally a good performance. We also note that the model tends to learn well i.e., it does not overfit on the train data.

We also note that the model when predicting the individuals corresponding to a given fingerprint, the model attains a classification accuracy of approximately 81.16% while classifying the fingerprints of individuals 22 and 43 with a f1-score of 0.93. The model attained a classification accuracy of approximately 81.16% on the complete dataset which overall is a good score given the amount of data the model was trained with.

When examining the effect of adjusting the learning rate on the performance of the model. It is observed that lowering or increasing the learning rate leads to a drop in the performance of the model per epoch. The higher learning rate leads to a model call back due to the failure of the model's loss to improve after 5 epochs.

Glimpse of the Confusion matrix for the model for each individual.

	precision	recall	f1-score	support
1	0.67	0.86	0.75	74
2	0.73	0.82	0.77	85
3	0.86	0.89	0.88	93
4	0.72	0.68	0.70	101
5	0.65	0.53	0.58	118
6	0.57	0.79	0.66	70
7	0.66	0.69	0.67	91
8	0.66	0.69	0.67	91
9	0.66	0.84	0.74	75
10	0.64	0.79	0.71	77
11	0.71	0.67	0.69	101
12	0.74	0.68	0.71	104
13	0.46	0.59	0.51	75
14	0.76	0.85	0.80	86
15	0.80	0.88	0.84	88
16	0.71	0.74	0.72	92
17	0.69	0.79	0.73	84

MY OBSERVATIONS AND LIMITATIONS OF THE MODEL.

- There are lots of individuals in the target variable. Given that there are 124 nodes in the final layer goes to show that the model will get more complicated to compile and build as the number of subjects to classify increase.
- Running the model once (100 epochs) which is necessary for the best model to attain a reasonable classification accuracy takes about 6 hours running three as I had done will take at least 18 hours of runtime.
- Reducing the hidden layers will lead to an insufficient model while increasing the hidden layers will increase the runtime exponentially to 6 minutes and 20 seconds per epoch which translates to 633 minutes (10 hours and 55 minutes). We instead chose to use the same number of layers but changed the learning rate (to check the effect of a learning rate hyperparameter). We have however included sections where we vary the number of hidden.
- The model didn't perform well if the learning rate is substantially increased, where as it performed well if the number of hidden layers is increased but at the expense of compilation complexity.

CONCLUSION

Since this is a novel method of using infrared images of the fingers, much research needs to be done on this subject. The data used in one of the papers that I studied was different to the data that I used, and the accuracies are, therefore, different. I experimented a lot using different values of hyperparameters and came up with different models. I had learnt in the class how different hyperparameters affect a CNN model, but through this project, I was experiencing what I had been taught in class to see the role of different hyperparameters.

Though a lot of work must be done, and I intend to continue and better my model in the future by using keras tuner which is an automatically tunes the hyperparameters and gives the best model with the best hyperparameters.

REFERENCES

[1] Mohd Shahrimie Mohd Asaari, Shahrel A. Suandi, Bakhtiar Affendi Rosdi,
of Band Limited Phase Only Correlation and Width Centroid Contour Distance for finger based biometrics,
Expert Systems with Applications,
Volume 41, Issue 7,
2014,
Pages 3367-3382,
ISSN 0957-4174

[2] @article{Boucherit2020FingerVI,
title={Finger vein identification using deeply-fused Convolutional Neural Network},
author={Ismail Boucherit and Mohamed Ould Zmirli and Hamza Hentabli and Bakhtiar Affendi Rosdi},
journal={Journal of King Saud University - Computer and Information Sciences},
year={2020}