# METHODOLOGY

In this chapter, the methodology is described in detail for each of the objective. In addition, the constraints the experiment is provided. In this project we use python Programming to develop codes. The VGGFace Model is used for Deep Face Recognition. We will apply transfer learning method with VGGFace model, WIKI dataset is used for data training and testing. There are 4 main step shows in figure 3-1 below:

Diagram

Description automatically generated Figure 3‑1: the flowchart of objective

## 3.1 The constraints of the experiment:

### 3.1.1 Apply VGGFace Model

VGGFace model is announced in 2015 in “Deep Face Recognition” by Omkar Parkhi and his partner. VGGFace contains 2.6 mill.VGGFace is used as the basis for developing deep CNNs for face recognition tasks such as face verification and identification.

VGG Face model uses architecture with blocks of convolutional layers with small kernels and ReLU activations followed by using max pooling layers and the end of the network contains fully connected layers.

A picture containing text, piano

Description automatically generated

Figure 3.1.1.1: VGGFace Architecture

We will create and load\_weights the VGGFace model in Python:

model = Sequential()  
model.add(ZeroPadding2D((1, 1), input\_shape=(224, 224, 3)))  
model.add(Convolution2D(64, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D((2, 2), strides=(2, 2)))  
  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(128, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(128, (3, 3), activation='relu'))  
model.add(MaxPooling2D((2, 2), strides=(2, 2)))  
  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(256, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(256, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(256, (3, 3), activation='relu'))  
model.add(MaxPooling2D((2, 2), strides=(2, 2)))  
  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(MaxPooling2D((2, 2), strides=(2, 2)))  
  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(ZeroPadding2D((1, 1)))  
model.add(Convolution2D(512, (3, 3), activation='relu'))  
model.add(MaxPooling2D((2, 2), strides=(2, 2)))  
  
model.add(Convolution2D(4096, (7, 7), activation='relu'))  
model.add(Dropout(0.5))  
model.add(Convolution2D(4096, (1, 1), activation='relu'))  
model.add(Dropout(0.5))  
model.add(Convolution2D(2622, (1, 1)))  
model.add(Flatten())  
model.add(Activation('softmax'))  
  
model.load\_weights('models/vgg\_face\_weights.h5')

Figure 3.1.1.2: Create and load VGGFace model

### 3.2 The WIKI dataset

Our dataset is a collection of more than 6000 informations of people such as data of birth, photo\_take, full\_path, gender, name, face\_location, face\_score, second\_face\_score. In this project, we are only concern with face\_score, date of birth and photo\_take.

### 3.3 Apply datenum to date year by using datenum\_to\_datetime to calculate age

In WIKI dataset, the data we used to calculate age include: photo\_take (exactly the day he/she take this photo ), dob(date of birth), face\_score.

Firstly, we need to convert dob to a year of birth, then the age is calculated by photo\_take minus year of birth.

Graphical user interface, text

Description automatically generated

Figure 3.3.1: datenum\_to\_datetime algorithm.

Secondly, we need to preprocessing data by doing some step such as : remove some images not include face images, remove images which has more than two face images, remove some information not need in our project(gender, name, face\_location,……)

Text

Description automatically generated

Figure 3.3.2: Preprocessing data.

### 3.4 Apply Transfer Learning method with VGGFace model

During the model building process, we encountered a situation where our model did not predict accurately. Although it has adopted complex architectures and is considered state-of-art

Firstly, we check with basic errors like:

* **Small data is not representative: Our dataset is too small in size. Therefore, the trained model does not learn general features to apply to classification tasks.**
* **Data imbalance model: When the model is out of data, it is more difficult to predict minority samples.**
* **Model architecture is too complex: For large datasets up to several million images, a model with a complex architecture can bring high accuracy. But with small data sets, the complex model reduces accuracy. I think the main reason is because complex models often overfitting.**
* **The optimization process is difficult: Maybe you have not set the learning rate well, so the training model takes a long time to converge or has not reached the global optimal point. You can then consider changing the method of updating gradient descent and setting the schedule learning rate. On tensorflow.keras we can set up schedule learning through CheckPoint as follows:**

Text

Description automatically generated

Figure 3.4.1: checkpoint on tensorflow.keras

Here we load weights images of VGGFace and apply transfer learning for training model again.

Text

Description automatically generated

Figure 3.4.2: Transfer learning with VGGFace

Then the data will be trained on the training dataset and testing on the validation set. Some of the more rigorous model development processes even further divide the dev set to fine tune the parameters between the models and the test set to test the model on actual user-generated data sets. However, to simplify, we will only use the train/validation set.

Finally, we will retrain the model. The first thing to do is to initialize the base network for the model. The essence of transfer learning is to freeze the existing layers of the VGGFace model and then train again. Re-train of age recognition model to increase accuracy more than an available model.

### 

# PRELIMINARY RESULTS

A picture containing text, indoor, person, electronics

Description automatically generated

# CONCLUSION AND FUTURE WORK

# REFERENCES

|  |  |
| --- | --- |
| [1] | <https://www.pvcfc.com.vn/dong-hanh-cung-nha-nong/tu-van-ky-thuat/benh-kham-la-san-dang-hoanh-hanh-va-lan-rong> |
| [2] | [*https://www.mdpi.com/2227-7390/10/4/580*](https://www.mdpi.com/2227-7390/10/4/580) |
| [3] | <https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where?tabs=azcli&fbclid=IwAR0zAlAHS1EV54RyLYbtJ-DGciWIAS8gsDBTbGsr3HNtmaJaEmg9To7EMkw> |
| [4] | <https://github.com/ahmedfgad/KivyAndroidClassification> |
| [5] | <https://ieeexplore.ieee.org/document/9137896?fbclid=IwAR3oKGsNbSDRXuRRZDftvkvvy-6dH1k8BkCUDGZ1MFECUki0DuzUnXYAbhk> |
| [6] | [Flask API | Learn the Creation and Examples of Flask API (educba.com)](https://www.educba.com/flask-api/) |

APPENDICES