Detection and Classification of Human Gender into Male and Female Using Convolutional Neural Network (CNN) Model

Abstract

This paper focuses on detecting the human gender using Convolutional Neural Network (CNN). The aim is to design and develop a real-time gender detection model using CNN, a deep learning algorithm used as an extractor of features that takes input images and assigns value to different aspects of the image and differentiate between them. The model focuses on classifying human gender only into two different categories; male and female. The major reason why this work was carried out is to solve the problem of imposture. A CNN model was developed to extract facial features such as eyebrows, cheek bone, lip, nose shape and expressions to classify them into male and female gender, and also use demographic classification analysis to study and detect the facial expression. We used both image processing technique and machine learning algorithm for implementation and achieved a promising result for the Kaggle dataset.

Keywords: Gender Detection, Convolutional Neural Network, Deep Learning, Artificial Intelligence, Gender

1.0 Introduction

Gender is one's identity in our society, a significant element used to identify a person in our society with reference to their biological difference. Artificial intelligence gender recognition can be used in many fields, such as smart human machine interface growth, health, cosmetics electronic commerce etc. For this work, Convolutional Neural Network (CNN), a Deep Learning Algorithm, was used to classify the human gender. CNN takes input images and assigns value to different aspects of the image and differentiate between them. Human gender detection which is part of facial recognition has received extensive attention. This can be achieved based on different static body features for example face, eyebrow, hand-shape, body shape, fingernail etc.

One of the most active areas in facial technology is facial recognition. A lot of works has been done using Deep Learning Methods such as Artificial Neural Network (ANN), CNN to determine age, gender estimation and emotion detection. [1] proposed a detection system using CNN model which can achieve 95% accuracy rate in age. In this work we have presented human gender classification using CNN.

The major reason why this work was carried out is to solve the problem of imposture. Imposture is an instance of pretending to be someone else in order to deceive others.

The aim of this work is to model a real-time gender detection system using CNN with the following objectives:

1. Design a convolutional neural network system to extract facial features of human.
2. Design a machine learning system that classifies human facial features into male and female gender.
3. To design a CNN using demographic classification analysis to study and detect facial expression and gender.

2.0 Review of Related Literature

In the field of Image Processing and Machine Learning, a lot of research work has been done on human gender estimation. Lian et al, [2] obtained an accuracy of 94.81% applying local binary pattern (LBP) and SVM within polynomial kernel on the CAS-PEAL face database.According to this method, a good accuracy can be achieved if the block size for the LSP operator is correctly selected, which is really a difficult task.

[3] performed the classification of gender utilizing only facial features (eyes, nose, mouth, brows, forehead). One drawback of this research is that the feature extraction method they have used is affected by complex backgrounds.

[4] used geometric based future for male and female classification where they have AR and ethnic dataset containing 126 frontal images in each dataset. Here they achieved 80.3% and 86.6% accuracy respectively.

[5] created a real-time Public Facial Recognition System (FRS) for Nigerian criminal investigation using HAAR cascades classifier technique, a prototype system to detect criminals in real time. The system achieved an 80% accuracy rate of collected photos stored in the database. This shows HAAR Cascade Classifier Technique may not be the best option for Gender Classification.

In 2010 a texture based local binary pattern has been for feature extraction and as classification algorithm naïve s Byes, ANN and linear SVM has been applied. They achieved 63% accuracy

with only 100 face images, that has been collected from Nottingham scan database which is quite low.

[6] have worked on gender classification from face image using LSP, SVM and black propagation. In this research they have used ORL dataset which contains 400 images and Nottingham scan data which contains 100 images. After implementation they gained 100% accuracy for ORL dataset and 71% accuracy for Nottingham scan database respectively.

So considering the whole literature review, it is clear that an improvement in gender classification is needed. The main disadvantage of the above gender classification research works is that the feature extraction and the classification are performed separately.

To obtain an optimum pre-processing and feature extraction design, prior knowledge is needed here. In the case of CNN which is a multi-layer neural network model, it can optimize fitters through automated learning where it is independent of prior knowledge which demonstrate a superior performance can be achieved using CNN.

A better proposition was done by [7]. They proposed real-time emotion detection used CNN with 9 layers for training and categorizing 7 different types of emotions which gives an accuracy around 90.

The emotion and gender detection system using CNN proposed by [8] which achieved for emotion detection 66% with FER emotion recognition dataset and with IMDB-WIKI dataset achieved 95% for gender detection.

[9] proposed an automatic emotion recognition system using LSP classifier which achieved 94.39% with CKT dataset and the system achieved a 92.22% with JAFFE dataset.

[10] proposed an age gender detection system achieved 72.53% for age detection and 98.90% for gender detection. Octavio, [11] proposed a real-time emotion gender classification system using CNN which achieved 66% accuracy on emotion classification on FER-20133 dataset and with IMDB dataset achieved 95% for gender classification.

2.1 Machine Learning in Facial Detection

Machine learning is a branch of artificial intelligence that deals with the design and development of algorithms that can learn form and make predictions on data [12]. Machine learning

algorithms learn from data to solve problems that are too complex to solve with conventional programming.

Face recognition is a biometric identification technique that uses unique characteristics of an individual's face to identify them. The most common type of Machine Learning Algorithm used in facial recognition is a deep learning Convolutional Neural Network (CNN) [12].

CNN is a type of artificial neural network that is well- suited for image classification tasks [13]. CNN learns to extract features from images and use those features to classify the images into different categories. The depth of a CNN is important for facial recognition because it allows the CNN to learn more complex facial features. Steps in facial recognition are;

1. **Face alignment and detection**: The first step is to detect faces in the input images. This can be done using a Haar Cascade Classifier [5], which is a type of a machine learning algorithm that is trained on positive and negative images.
2. **Features measurement extraction**: Once faces have been aligned and detected, the next step is to extract features from them.
3. **Face recognition**: The last step is to match the extracted features with faces in a database.

3.0 Materials and Methods

The methods used in this work are; Records view and Observation methods. Image samples used for data classification and feature extraction ware analyzed.The authors took time to observe the end methods of which images were captured and classified and how the database saved the records.

The primary objective of this work is to recognize the gender with emotions from the human face images utilizing the set of facial features in real-time.

Facial extraction from face images is an important part of this method (as shown in figure 1), and it involves four stages. The stages include;

1. Image Pre-Processing: The preprocessing stage can improve the quality of the input image by eliminating noise and smoothing the images. It eliminates images redundancy without image details. Preprocessing also includes filtering and normalizing the image to produce a uniform size and rotated image.

2. Face Detection: In object detection [14] explained feature extraction as extracting features from the background of input images with various lighting conditions and complex backgrounds. It involves segmentation and extraction of facial features from uncontrolled background.
3. Feature Extraction: It includes shapes, movement, color, the texture of the facial image. It extracts meaningful information of an image compared to the original image.
4. Feature Classification: The classification stage recognizes facial images and group them according to certain classes and helps them skilled recognition. Feature classification is a selection stage, which deals with exchanges the retain essential information and connect them in certain parameters.

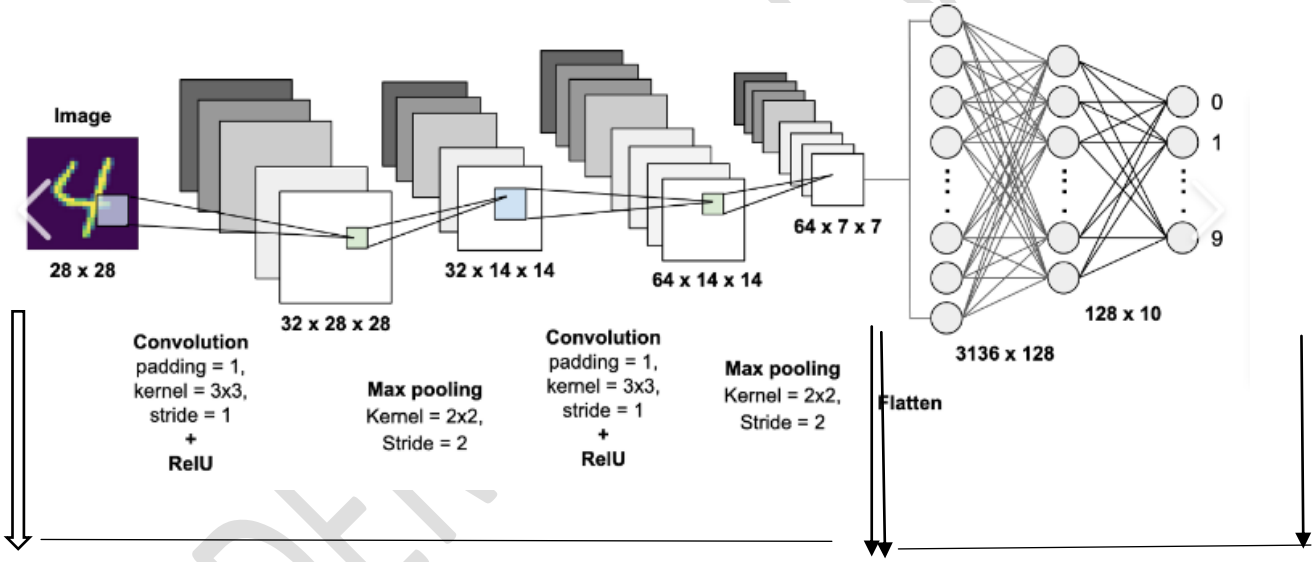


Figure 1: Feature Extraction and Classification stages of Facial Extraction

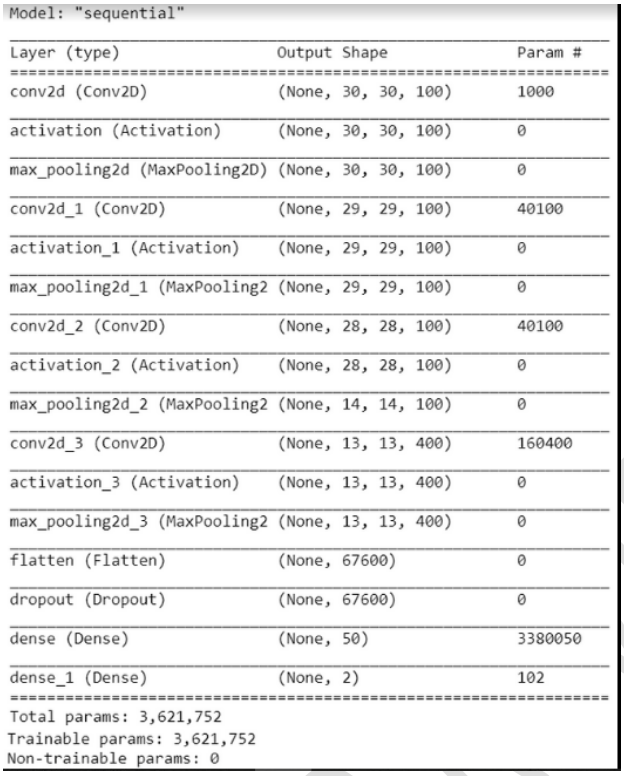
Data Training

Data was gotten from Kaggle, and trained. We summarize the model by using "Model.Summary()" method. The following are summary of the trained data set;

TOTAL PARAM's: 3,631,752

Trainable PARAM's: 3,621,752

Non-trainable params: 0. See more details in fig 2.



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 100)	1000
activation (Activation)	(None, 30, 30, 100)	0
max_pooling2d (MaxPooling2D)	(None, 30, 30, 100)	0
conv2d_1 (Conv2D)	(None, 29, 29, 100)	40100
activation_1 (Activation)	(None, 29, 29, 100)	0
max_pooling2d_1 (MaxPooling2	(None, 29, 29, 100)	0
conv2d_2 (Conv2D)	(None, 28, 28, 100)	40100
activation_2 (Activation)	(None, 28, 28, 100)	0
max_pooling2d_2 (MaxPooling2	(None, 14, 14, 100)	0
conv2d_3 (Conv2D)	(None, 13, 13, 400)	160400
activation_3 (Activation)	(None, 13, 13, 400)	0
max_pooling2d_3 (MaxPooling2	(None, 13, 13, 400)	0
flatten (Flatten)	(None, 67600)	0
dropout (Dropout)	(None, 67600)	0
dense (Dense)	(None, 50)	3380050
dense_1 (Dense)	(None, 2)	102

Total params: 3,621,752
Trainable params: 3,621,752
Non-trainable params: 0

Figure 2: Details of the Trainable and Untrainable Parameters

Now the model is trained and its ready to recognize the gender of any random image from a dataset. We plot a random sample of 15 test images, their predicted labels and ground truth.

In this work, Keras is used. Keras is an open-source Neural Network library. It enables the model to perform random transformations and normalization operations on batches of image data by working on different attributes such as height shift, width shift, rotation range, rescale, range of shear, horizontal rip and fill mode. Using these attributes, the system can automatically rotate, translate, rescale and zoom into or out of images as well as apply shearing transformation, rip images horizontally, fill in newly created pixels, etc. for the purpose of image classification.

Formulations

The number of layers of the proposed CNN model is reduced by fusing a convolutional and a subsampling layer into one layer.In this work, we replace consecutive convolutional and subsampling layers with one convolutional layer (*l*) using strides of 2.

A feature map, $Y_j(x, y)$ in layer l can be described by the following equation:

$$Y_j^{(l)}(x, y) = f\left(\sum_{i=0}^{N}\sum_{u=0}^{K_x^{(l)}}\sum_{v=0}^{K_y^{(l)}} Y_i^{(l-1)}\left(S_x^{(l)}x + u, S_y^{(l)}y + v\right) w_{ji}^{(l)}(u, v) + \theta_j^{(l)}\right), \quad (1)$$

where $Y_i^{(l-1)}$ and $Y_j^{(l)}$ are the input and output feature maps respectively, *f*() denotes the activation function, $w_{ji}^{(l)}$ is the convolutional kernel weight, $\theta_j^{(l)}$ is the bias, *N* represents the total number of input feature maps,

$S_x^{(l)}$ is the horizontal convolution step size, $S_y^{(l)}$ is the vertical convolution step size, and $K_x^{(l)}$ and $K_y^{(l)}$ are the width and height of convolutional kernels, respectively. The width $W^{(l)}$ and height $H^{(l)}$ of the output feature map with convolution step sizes of $S_x^{(l)}$ and $S_y^{(l)}$ can be computed as:

$$W^{(l)} = \frac{W^{(l-1)} - K_x^{(l)}}{S_x^{(l)}} + 1 \tag{2}$$

and

$$H^{(l)} = \frac{H^{(l-1)} - K_y^{(l)}}{S_y^{(l)}} + 1, \tag{3}$$

where $W^{(l-1)}$ and $H^{(l-1)}$ correspond to the width and height of input feature map.

Figure 3 gives a pictorial representation of the formulation above. The diagram shows that a 5 × 5 convolution followed by a 2 × 2 subsampling operation can be replaced by a single 6 × 6 convolution operation with step size (strides) of 2, because they generate exactly the same output feature map.

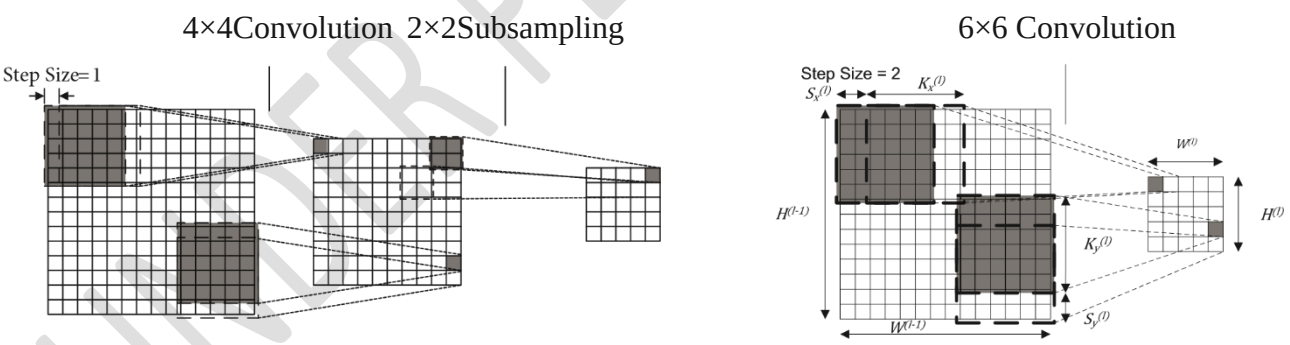


Figure 3: Operations on feature maps in the convolutional layer: (a) conventional approach applying convolution (with step size of 1) followed by subsampling, (b) fused convolution/subsampling approach applying convolutions with step size of 2.

Note that, in the formulation given above, we perform cross-correlation instead of convolution. In image processing, convolution and cross-correlation perform similar functions, except that in the convolution process, the kernel weights are flipped vertically and horizontally. The general equation for a 2D discrete convolution is given by:

$$K_x \;\; Ky$$

$$Y(x, y) = \sum \sum X(x - u, y - v)w(u, v) \qquad (4)$$

$$u=0 \; v=0$$

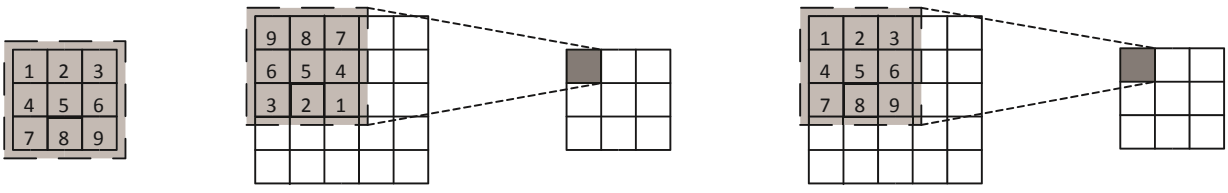
where X is an input image, Y is the output image, w is the kernel, and K_x and K_y represent the width and height of convolutional kernel, respectively. This operation involves flipping of the kernel weights before the dot product is performed. In contrast, a 2D discrete cross-correlation (for image processing) is described by the following equation:

$$K_x \;\; Ky$$

$$Y(x, y) = \sum \sum X(x + u, y + v)w(u, v). \qquad (5)$$

$$u=0 \; v=0$$

Essentially, Eqs. (4) and (5) are similar, except in Eq. (5), the kernel weights are not flipped. We illustrate these operations using the diagrams in Figure 4, where Figure 4a gives an example of convolution kernel. In the conventional approach, as depicted in Figure 4b, a 2D discrete convolution is performed by convolving an overlapped input plane with the convolution kernel with kernel weights being flipped in both horizontal and vertical directions. In Figure 4c, the same operation is shown, but now without flipping the kernel. Since values of the convolution kernel (weights) in a convolutional layer are randomly initialized, flipping has little effect on the convolution output.



(a) (b) (c)

Figure 4: The 2D discrete convolution operation: (a) example convolution kernel, (b) convolution with kernel weights flipped, (c) convolution with kernel weight not flipped.

Flipping operations add more computational time in both feedforward and backward propagations. Hence, the modification of replacing convolution with cross-correlation is advantageous, especially when dealing with numerous convolutions to be performed on large kernel sizes, and on large datasets within a large number of training iterations (epochs).

We apply, in the convolutional layers, a scaled hyperbolic tangent function as described by the following equation:

$$f(x) = A \tanh(Bx), \tag{6}$$

Where A denotes the amplitude of the function and B determines its slopes at the origin. The values of A and B are set at 1.7159 and 2/3 as suggested by [15].

For the output layer, a single perceptron is used to determine the class of a particular input pattern. The value at output neuron is given by the following equation:

$$Y_j^{(l)} = f\left(\sum_{i=1}^{N} Y_i^{(l-1)} w_{ji}^{(l)} + \theta_j^{(l)}\right), \tag{7}$$

where $f()$ represents the activation function used for each neuron, N is the total number of input neuron(s), $Y_i(l^{-1})$ is the value of the input neuron, $w_{ji}^{(l)}$ is the synaptic weight connecting the input neuron(s) to the output neuron, and $\theta_j^{(l)}$ denotes the bias. Male gender is represented by the output value of +1, whereas −1 indicates that the pattern belongs to the female class.

4.0 Implementation

Python programming language and MxNet frameworks were used to implement this project. MxNet is an open-source deep learning framework that allows you to define, train, and deploy deep neural networks on a wide array of platforms, from cloud infrastructure to mobile devices.

Getting Started

Kaggle's 2018 dataset named "Facial Age" was used. It's a 9,778 RGB images of faces in PNG format of size 200X200 pixels each. It covers a long age span, across 1 to 116 years. Image labels are embedded in file names, as per this nomenclature. So, we have age, gender and race, separated by underscores.

We extracted gender labels embedded into these image file names and made our model fitting less computationally expensive by converting our images to grayscale, as we used CNN based deep learning approach for model building. We split our dataset into 75% –25% training and test sets respectively. Then on this training set, we trained our CNN model, and validated performance on the test set.

Model Building

We set up our environment by importing needed libraries. These libraries include; pandas, numpy, cv2, matplotlib, tensorflow, Google Colab, keras, sklearn, etc. We then loaded our dataset, using a a "for loop" to read all images, convert from RGB to grayscale, and resize to 100x100. And then, we appended all image pixel data to this pixels numpy array and labels to gender array as shown in figure 5.

```
# Mount to Drive
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/My Drive/Project5_AgeGenderEmotion_Detection/1.2_gender_input_output

# Load dataset
path = "./input/UTKFace"
pixels = []
age = []
gender = []
i=0
for img in os.listdir(path):
  i=i+1
  genders = img.split("_")[1]
  img = cv2.imread(str(path)+"/"+str(img))
  img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  img=cv2.resize(img,(100,100))
  pixels.append(np.array(img))
  gender.append(np.array(genders))
pixels = np.array(pixels)
gender = np.array(gender,np.uint64)

# Split Dataset into 75:25 training & test
x_train,x_test,y_train,y_test = train_test_split(pixels,gender,random_state=100)
```

Figure 5: Code showing loading of dataset

We then define our CNN architecture. To summarize, we have:

- An input 2D Convolutional layer (with 32 filters) paired with an 2D MaxPooling layer.

- 3 pairs of 2D Convolutional layers (with 64, 128 & 256 filters respectively) paired again with 2D MaxPooling layers.
- 1 Dense layer with 128 nodes.
- An output Dense layer with 2 nodes, which are essentially our labels male or female

Next up, we are compiled the above defined CNN architecture. We used model checkpoint for saving our model as it continues improving in performance across 30 epochs for model fitting. Using code block in figure 6, the model's performance was checked by plotting the lineplots for loss and accuracy, as shown in figure 7.

```
# Checking the train and test loss and accuracy values from the neural network above.
train_loss = save.history['loss']
test_loss = save.history['val_loss']
train_accuracy = save.history['accuracy']
test_accuracy = save.history['val_accuracy']

# Plotting a line chart to visualize the loss and accuracy values by epochs.
fig, ax = plt.subplots(ncols=2, figsize=(15,7))
ax = ax.ravel()
ax[0].plot(train_loss, label='Train Loss', color='royalblue', marker='o', markersize=5)
ax[0].plot(test_loss, label='Test Loss', color = 'orangered', marker='o', markersize=5)
ax[0].set_xlabel('Epochs', fontsize=14)
ax[0].set_ylabel('Categorical Crossentropy', fontsize=14)
ax[0].legend(fontsize=14)
ax[0].tick_params(axis='both', labelsize=12)
ax[1].plot(train_accuracy, label='Train Accuracy', color='royalblue', marker='o', markersize=5)
ax[1].plot(test_accuracy, label='Test Accuracy', color='orangered', marker='o', markersize=5)
ax[1].set_xlabel('Epochs', fontsize=14)
ax[1].set_ylabel('Accuracy', fontsize=14)
ax[1].legend(fontsize=14)
ax[1].tick_params(axis='both', labelsize=12)
fig.suptitle(x=0.5, y=0.92, t="Lineplots showing loss and accuracy of CNN model by epochs", font
```

Figure 6: Codeblock for line plots

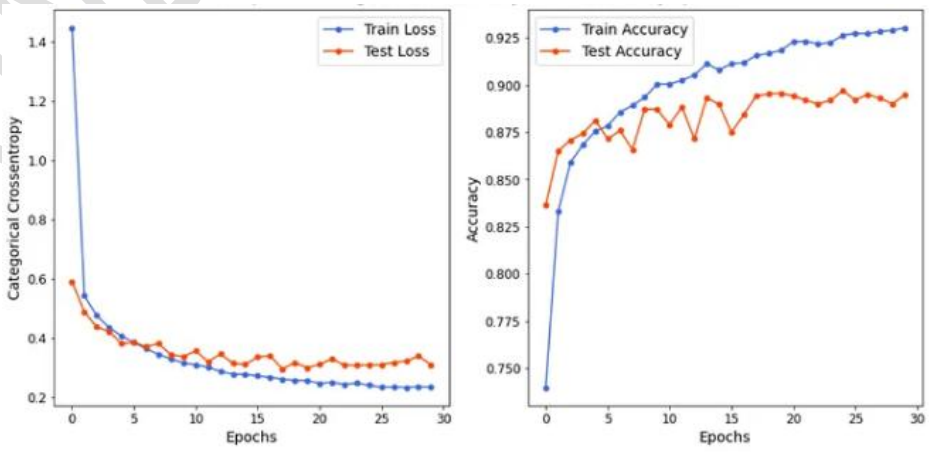


Figure 7: Lineplots showing loss and accuracy by epochs

Result

A real-time gender detection system using CNN was developed. The system detects gender of the human being's image that is captured via the web camera of the system. Figure 8 shows a screenshot of the system.

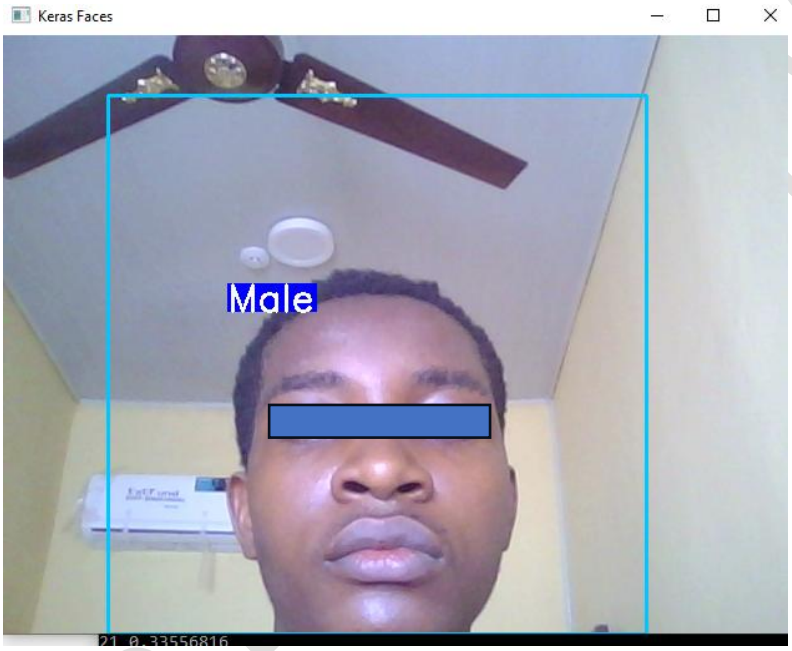


Figure 8:Real-time Gender detection system

5.0 Conclusion

The classification of human gender is a valuable tool for gathering data about and from individuals. The human face contains enough information to be useful in a variety of contexts. Gender classification is crucial for targeting the appropriate audience. We implemented a machine learning algorithm along with image processing techniques in this work, and the results look good for Kaggle data. To find out which optimizer produces the best output, several optimizers have been used. Following analysis of the data, a comparison between the efficacy of our model and our article has been provided, demonstrating where our system outperforms them.

References

[1] Uddin Md. Jashim, Dr. Paresh Chandra Barman, Khandaker TakdirAhmed S.M. Abdur Rahim , Abu Rumman Refat , Md Abdullah-AlImran6 "A Convolutional Neural Network for Real-time FaceDetection and Emotion & Gender Classification" IOSR Jo (2009)

[2] Lian, H.-C., Lu, B.-L.: Multi-view gender classification using local binary patternsand support vector machines. In: Wang, J., Yi, Z., Zurada, J.M., Lu, B.-L., Yin,H. (eds.) ISNN 2006. LNCS, vol. 3972, pp. 202–209. Springer, Heidelberg (2006).https://doi.org/10.1007/11760023 30

[3] Li, P & Ma, X (2009) "Learning gender with support faces. IEEE Trans, vol,24 no 5

[4] Seaced , D (2009) " A fast accurate unconstrained face detection. IEEE TRANS, vol 38,n02

[5] Onyemachi J N, Gift Adene, Belonwo T. S., Chinedu E. M., Adannaya U. G-A. (2024, March 11). *Digital Criminal Biometric Archives (DICA) and Public Facial Recognition System (FRS) for Nigerian criminal investigation using HAAR cascades classifier technique*. World Journal of Advanced Engineering Technology and Sciences. https://doi.org/10.30574/wjaets.2024.11.2.0077

[6] Sajja, T. (2009) " Deep face recognition ". In proc-bmvc vol,1.2009.

[7] Rajesh, K & Gavi, K (2017) "Facial emotion analysis using deep CNN: IEEE No:339:443 vol 4 .2017

[8] Abdullah-Al-Imran MD "A Convolutional Neural Network for Realtime Face Detection and Emotion & Gender Classification'' e-ISSN:2278-2834, p- ISSN: 2278-8735. Volume15, Issue 3, Ser. I (May -June2020), PP 37-46

[9] Happy SL and Aurobinda Routray"Automatic Facial ExpressionRecognition Using Features of Salient Facial Patches'' DOI: 10.1109/TAFFC. 2014. 2386334 https://rb.gy/9m5dt2.

[10] Jang-Hee Yoo, So-Hee Park, and Yongjin Lee "Real-Time AgeandGender Estimation from Face Image" ISBN: 978-0-6480147-3-7.

[11] Octavio Arriaga1 and Matias Valdenegro-Toro and Paul G. Pl¨oger (2017) Real-time Convolutional Neural Networks for emotion and gender classification."

[12] Yavanoglu, O., Aydos, M.: A review on cyber security datasets for machine learningalgorithms. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 2186–2193(2017)(PDF) Machine Learning and Deep Learning Techniques for Cybersecurity: A Review.

[13] Khalil, T. (2020) "On convolutional neural network CNN, for real-time face detection & gender classification.

[14] Sajja, T & Voila, P (2009)" on comparison of feature extraction algorithms for gender classification from face images. Int, research technol, vol2,no5, 2009

[15] LeCun, Y., Bengio, Y., & Hinton, G. E. (2015, May 27). *Deep learning*. Nature. https://doi.org/10.1038/nature14539