DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

SPECIAL TOPIC -1 REPORT

SIGNATURE CLASSIFICATION USING CNN

By

ABDUL HAFEEZ [ENG21CS0004]

ALISHA ARIC FERNANDES[ENG21CS0029]

BHAMINI A.V[ENG21CS0072]

BHAT DHANVI [ENG21CS0074]

Under the supervision of

DR. PRAMOD KUMAR NAIK

ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY

DAYANANDA SAGAR UNIVERSITY

School of Engineering

Department of Computer Science and Engineering

Kudlu Gate, Bangalore –560068

Karnataka, India

CERTIFICATE

This is to certify that the Special Topic 1 titled **"SIGNATURE CLASSIFICTAION USING CNN"** is carried out by **Abdul Hafeez (ENG21CS0004), Alisha Aric Fernandes (ENG21CS0029), Bhamini A.V(ENG21CS0072), Bhat Dhanvi (ENG21CS0074)** bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2022-2023**.

Dr.Pramod Kumar Naik	**Dr. Girisha G S**	**Dr. Uday Kumar Reddy K R**
Associate Professor Dept. of CSE, School of Engineering Dayananda Sagar University Date:	Chairman, CSE School of Engineering Dayananda Sagar University Date:	Dean School of Engineering Dayananda Sagar University Date:

Name of the Examiner **Signature of Examiner**

1.

2.

DECLARATION

We, **Abdul Hafeez(ENG21CS0004), Alisha Aric Fernandes(ENG21CS0029),Bhamini A.V(ENG21CS0072),Bhat Dhanvi(ENG21CS0074)** are students of the fourth semester B. Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Special-topic Project titled **"**Signature Classification using CNN**"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2022-2023**.

Student	**Signature**
Abdul Hafeez **ENG21CS0004**	
Alisha Aric Fernandes **ENG21CS0029**	
Bhamini A.V **ENG21CS0072**	
Bhat Dhanvi **ENG21CS0074**	
Place: Bangalore **Date: 9-06-2023**	

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this Special Topic 1.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. Uday Kumar Reddy K R**, **Dean, School of Engineering & Technology, Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Chairman, Department of Computer Science, and Engineering**, **Dayananda Sagar University**, for providing the right academic guidance that made our task possible.

We would like to thank our guide **Dr.Pramod Kumar Naik**, **Associate Professor, Dept. of Computer Science and Engineering, Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our Special Topic 1, which paved the way for smooth progress and the fruitful culmination of the project.

We would like to thank our Special Topic 1 Coordinators, Dr. Savitha Hiremath, Dr. Bondu Venkateswarlu, and all the staff members of Computer Science and Engineering for their support.

We are also grateful to our family and friends who provided us with every requirement throughout the course.We would like to thank one and all who directly or indirectly helped us in the Special Topic 1.

CONTENTS

NOMENCLATURE USED

DL	Deep Learning
ML	Machine Learning
PY	Python

ABSTRACT

Signature classification has traditionally relied on manual comparison of signatures with genuine samples, a process prone to human error and time-consuming. This study presents a novel approach to automate the classification process using machine learning techniques and neural networks. A signature dataset is utilized to develop a predictive model, enabling faster and more accurate classification.

Transfer learning is applied by leveraging pre-trained convolutional neural networks (CNNs) such as Xception and VGG16. These models were compared to assess their performance in signature classification. Xception emerged as the top-performing architecture, achieving an outstanding accuracy of 97%.

The automation of signature classification through machine learning offers several advantages over manual methods. It reduces the time required for analysis, eliminates human error, and enhances overall accuracy.

CHAPTER 1

INTRODUCTION

Machine learning is a groundbreaking approach invented by humans to simplify complex problems by training computers to mimic human brain functioning. Deep learning, a subset of machine learning, is specifically designed to imitate how humans learn and process specific types of information. It incorporates statistical techniques and predictive modeling, making it highly advantageous for data scientists. Deep learning, with its capability to handle vast amounts of data, has emerged as a powerful tool in various domains.

In the field of deep learning, Convolutional Neural Networks (CNNs or ConvNets) have gained significant popularity, especially in Computer Vision applications. These networks are particularly effective in tasks like image classification, object detection, and recognition. Recognizing the potential of CNNs, we developed a CNN model for signature classification.

To enhance the performance of the model, transfer learning was employed. Transfer learning involves leveraging the pre-trained features of existing CNN architectures such as GoogLeNet, Xception, VGG16, ResNet and etc. we have compared the architectures of Xception and VGG16 to access their performance in signature classification.

The evaluation of our models involved computing accuracy, confusion matrix, and classification report. The results obtained were outstanding, showcasing the efficacy of the developed model. Among the two CNN architectures, Xception achieved the highest accuracy of 97%, demonstrating its superior performance in accurately classifying signatures.

The use of transfer learning and the comparative analysis of various CNN architectures highlight the robustness and flexibility of the designed model. By harnessing the power of deep learning and leveraging pre-trained networks, we were able to achieve exceptional Accuracy in signature classification.

The findings of this study provide valuable insights into the potential applications of CNNs and deep learning in signature classification tasks. The high accuracy achieved by Xception signifies its effectiveness in accurately differentiating signatures. These results contribute to the advancement of automated signature classification and have significant implications in fields such as document verification, fraud detection, and forensic analysis

CHAPTER 2

PROBLEM STATEMENT

Manual comparison of signatures with copies of genuine signatures for classification is a time-consuming process prone to human error. This traditional method not only consumes significant resources but also results in inconsistent and inaccurate predictions. It is crucial to develop an automated solution that can efficiently and reliably classify signatures, overcoming the limitations of manual comparison

SOLUTION

We have developed an effective solution to address the challenges of manual signature classification by creating a machine learning model using neural networks. This model significantly reduces the time required for classification while achieving high accuracy results.

CHAPTER 3

LITERATURE REVIEW

Multi-scripted Writer Independent Off-line Signature Verification using Convolutional Neural Network" by Teressa Longjam, Dakshina Ranjan Kisku & Phalguni Gupta(01 August 2022)

This paper reports a writer-independent offline signature verification system that uses Convolutional Neural Network (CNN) architecture for feature extraction and classification. The objective of the proposed work is to model a CNN-based adaptable system to verify multi-scripted offline signatures. The model has been trained and tested on two publicly available databases, viz. CEDAR and BH-Sig260, which consist of Hindi, Bengali, and English signatures. Individual signature classes of unique scripts and a combination of these scripts have been considered for testing the proposed model that determines verification accuracies of 90%, 95%, 98.33%, and 93.33%, respectively. Experimental results are compelling, and the proposed model outperforms the verification accuracies of some well-known models.

"Offline Signature Verification with Convolutional Neural Networks"by Gabe Alvarez galvare, Blue Sheffer bsheffer and Morgan Bryant(Stanford …, 2016)

This paper aims to automate the process of signature verification by using convolutional neural networks (CNNs). This paper is based on the googlenet architecture, and we use the ICDAR 2011 SigComp dataset to train our model with transfer learning. When classifying whether a given signature was a forgery or genuine, we achieve accuracies of 97% for Dutch signatures and 95% for Chinese Signatures. We also performed several experiments altering the types of training data and prediction task to make it more relevant to real-life applications, for which our methods seem promising but for which we were not able to achieve results much higher than a naive baseline.

"Writer-independent feature learning for Offline Signature Verification using Deep Convolutional Neural Networks"by Luiz G. Hafemann, Robert Sabourin and Luiz S. Oliveira(2016 International Joint Conference on Neural Networks (IJCNN))

Automatic Offline Handwritten Signature Verification has been researched over the last few decades from several perspectives, using insights from graphology, computer vision, signal processing, among others. In spite of the advancements on the field, building classifiers that can separate between genuine signatures and skilled forgeries (forgeries made targeting a particular signature) is still hard. We propose approaching the problem from a feature learning perspective. Our hypothesis is that, in the absence of a good model of the data generation process, it is better to learn the features from data, instead of using hand-crafted features that have no resemblance to the signature generation process. To this end, we use Deep Convolutional Neural Networks to learn features in a writer-independent format, and use this model to obtain a feature representation on another set of users, where we train writer-dependent classifiers. We tested our method in two datasets: GPDS-960 and Brazilian PUC-PR. Our experimental results show that the features learned in a subset of the users are discriminative for the other users,

including across different datasets, reaching close to the state-of-the-art in the GPDS dataset, and improving the state-of-the-art in the Brazilian PUC-PR dataset.

"Learning features for offline handwritten signature verification using deep convolutional neural networks"by Luiz G. Hafemann a, Robert Sabourin and Luiz S. Oliveira(October 2017)

This paper propose learning the representations from signature images, in a Writer-Independent format, using Convolutional Neural Networks. In particular, This paper propose a novel formulation of the problem that includes knowledge of skilled forgeries from a subset of users in the feature learning process, that aims to capture visual cues that distinguish genuine signatures and forgeries regardless of the user. Extensive experiments were conducted on four datasets: GPDS, MCYT, CEDAR and Brazilian PUC-PR datasets. On GPDS-160, we obtained a large improvement in state-of-the-art performance, achieving 1.72% Equal Error Rate, compared to 6.97% in the literature. We also verified that the features generalize beyond the GPDS dataset, surpassing the state-of-the-art performance in the other datasets, without requiring the representation to be fine-tuned to each particular dataset.

CHAPTER 4
PROJECT DESCRIPTION

We aim to develop a model which can classify the signatures to their classes using the Convolutional Neural Network.

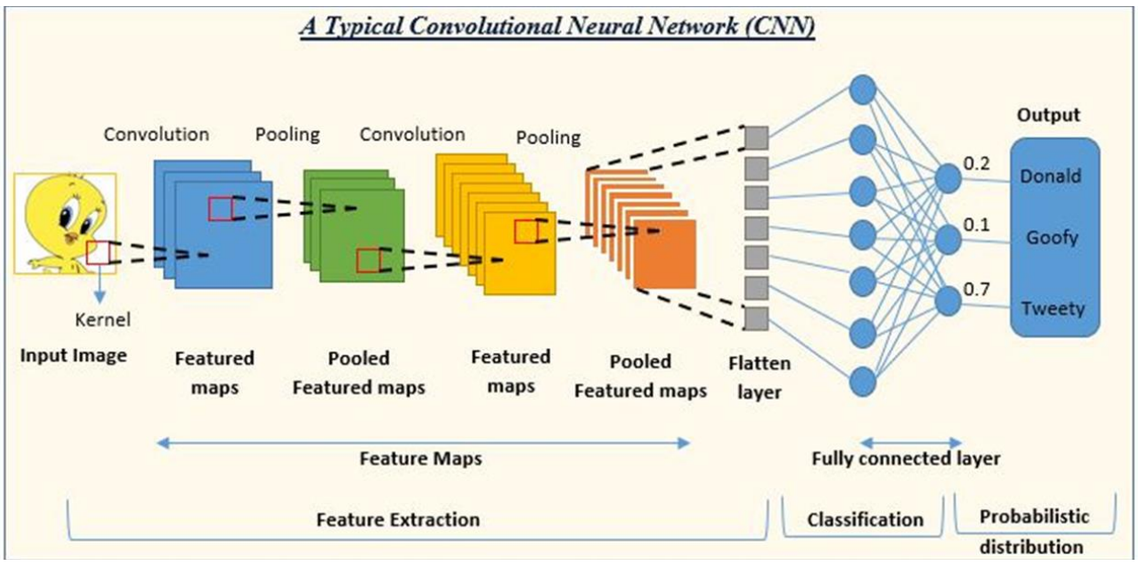


Fig 4.1: CNN Architecture

There are 14 layers which are added sequentially to the model.

- Rescaling layer
- Random zoom layer
- Random rotation layer
- Two Conv2D layer
- Two MaxPooling2D layer
- Batch Normalization layer
- Three Dropout layers
- Flatten layer
- Two Dense layer

Brief Description of each layer

1. **Rescaling:** This layer rescales the input image pixel values to a specified range (usually between 0 and 1) to help improve training.

2. **RandomZoom:** This layer randomly zooms into the input image to add more variability to the training data.

3. **RandomRotation:** This layer randomly rotates the input image to add more variability to the training data.

4. **Conv2D:** This layer applies a 2D convolution operation on the input image to extract features. The layer has 16 filters (or channels) and a kernel size of (3,3).

5. **MaxPooling2D:** This layer performs max pooling operation on the output of the previous layer to downsample the feature maps and reduce the dimensionality. The layer uses a pool size of (2,2).

6. **BatchNormalization:** This layer normalizes the output of the previous layer to help improve training and reduce overfitting.

7. **Dropout:** This layer randomly drops out (disables) a fraction of the neurons to help reduce overfitting.

8. **Conv2D_1:** This layer applies another 2D convolution operation on the output of the previous layer to extract more complex features. The layer has 16 filters and a kernel size of (3,3).

9. **MaxPooling2D_1:** This layer performs another max pooling operation on the output of the previous layer to further downsample the feature maps. The layer uses a pool size of (2,2).

10**. Dropout_1:** This layer randomly drops out a fraction of the neurons again to help reduce overfitting.

11. **Flatten:** This layer flattens the output of the previous layer into a 1D vector to feed into the dense layers.

12. **Dense:** This layer is a fully connected (dense) layer with 256 neurons that performs a linear operation on the input vector followed by a non-linear activation function (usually ReLU).

13. **Dropout_2:** This layer randomly drops out a fraction of the neurons to help reduce overfitting.

14. **Dense_1:** This layer is the output layer with 4 neurons (assuming this is a multi-class classification task) that performs a linear operation on the input vector followed by a softmax activation function to output probabilities for each class.

4.1 PROJECT DESIGN

1) Initially the signature is given as input to the CNN model.

2) Pre-processing of the image to remove errors and improve the quality of the input image.

3) Extracting the features from the input image by applying kernals and generating the feature maps with revelent and important features.

4) After extracting the features , the image will be classified to a particular class after passing through a series of convutional layers.

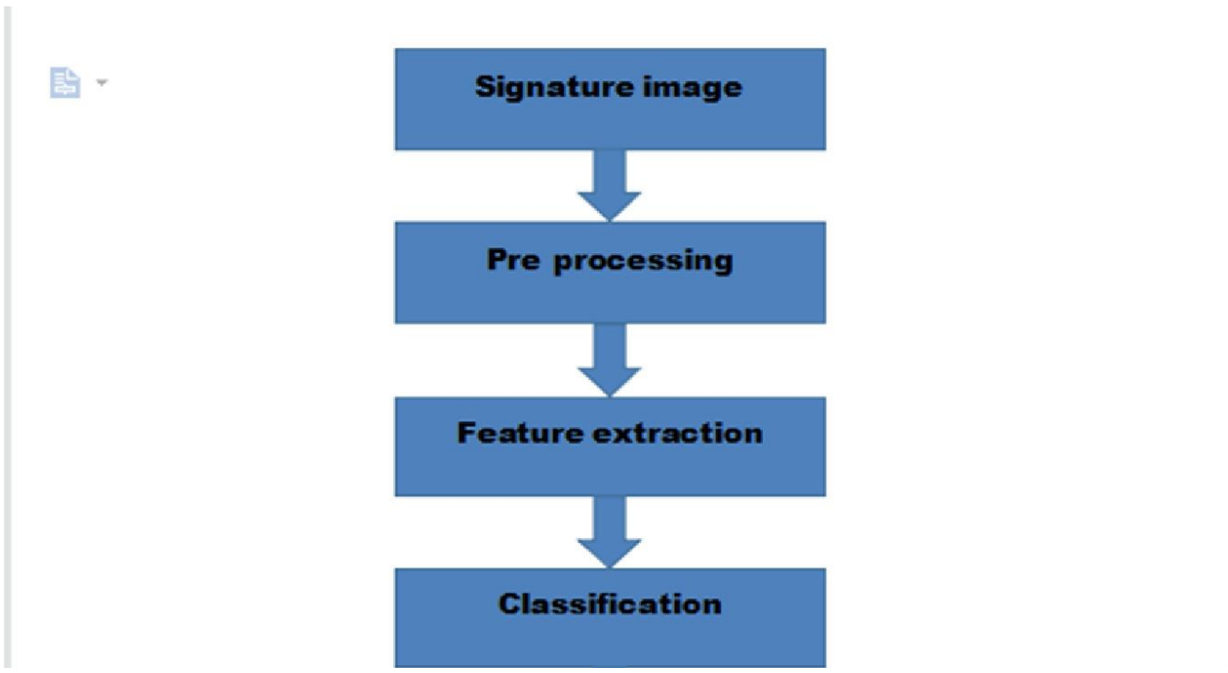


Fig 4.1.1: Design workflow diagram

CHAPTER 5

HARDWARE AND SOFTWARE REQUIREMENTS

5.1 HARDWARE REQUIREMENTS

- System(PC)
- Stable Internet Connection

5.2 SOFTWARE REQUIREMENTS

- Python 3.9
- Google colab

CHAPTER 6

METHODOLOGY

Approach:

1. Collect the dataset of signatures for classification purpose.
2. We did this by manually signing on the paper and created 4 folders(class) in Google drive and mounted it using google colab.
3. Divide the dataset into training and testing set in 70:30 ratio.
4. Pre-process the dataset and prepare it for classification.
5. Train a CNN(convulational neural network) model for signature classification.
6. Evaluate the performance by calculating accuracy.

CHAPTER 7

EXPERIMENTATION

7.1 BASE CODE

```
import tensorflow as tf
import numpy as np

from google.colab import drive
drive.mount('/content/gdrive')

train_path='/content/gdrive/MyDrive/dataset/train'
test_path='/content/gdrive/MyDrive/dataset/test'

val_ds=tf.keras.utils.image_dataset_from_directory(test_path,ima
ge_size=(100,100),batch_size=20)

train_ds=tf.keras.utils.image_dataset_from_directory(train_path,
image_size=(100,100),batch_size=20)

train_ds.prefetch(1)
class_names=train_ds.class_names
num_classes=len(class_names)
print(class_names)
print(num_classes)

import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
for img,label in val_ds.take(1):
  for i in range(4):
    ax=plt.subplot(2,2,i+1)
    plt.imshow(img[i].numpy().astype("uint8"))
    plt.title(class_names[label[i]])
    plt.axis("off")

norm_layer=tf.keras.layers.experimental.preprocessing.Rescaling(
1./255)
zoom_layer=tf.keras.layers.experimental.preprocessing.RandomZoom
(height_factor=(0.1,0.3))
rot_layer=tf.keras.layers.experimental.preprocessing.RandomRotat
ion(factor=0.2)

from keras.layers import
Conv2D,MaxPooling2D,Input,Flatten,Dense,BatchNormalization,Dropo
ut
from keras.models import Sequential
```

```
def get_model(num_classes):
  model = Sequential([norm_layer,
        zoom_layer,
        rot_layer,
        Conv2D(16, (5, 5), padding='same', activation='relu'),
        MaxPooling2D((2,2)),
        BatchNormalization(),
        Dropout(0.3),
        Conv2D(16, (3, 3), padding='same', activation='relu'),
        MaxPooling2D((2,2), strides=(2,2)),
        Dropout(0.3),
        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
    ])
  model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),

 loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=
False, reduction='auto'),
         metrics=['accuracy'])
  return model
model = get_model(num_classes)

history=model.fit(train_ds,validation_data=val_ds,epochs=50)

import matplotlib.pyplot as plt
fig=plt.figure(figsize=(12,5))
fig.add_subplot(121)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss vs.epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training','Validation'],loc='best')
fig.add_subplot(122)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy vs.epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training','Validation'],loc='best')
plt.show()

from keras.utils.np_utils import to_categorical
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
test_datagen = ImageDataGenerator()
```

```
test_ds = test_datagen.flow_from_directory(
test_path,
target_size=(100,100),
batch_size =1,class_mode ='categorical',
color_mode ="rgb",
shuffle=False,

)

test_ds.reset()
predictions=model.predict(test_ds,steps=len(test_ds.filenames),v
erbose=1)

import pandas as pd
predicted_class_indices=np.argmax(predictions,axis=1)
labels=(test_ds.class_indices)
labels=dict((v,k) for k,v in labels.items())
pred_labels=[labels[k] for k in predicted_class_indices]

filenames=test_ds.filenames
results=pd.DataFrame({"Filename":filenames,"Predictions":pred_la
bels})
results.tail(110)

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import seaborn as sns


Y_pred = model.predict_generator(test_ds,120)
y_pred = np.argmax(Y_pred, axis=1)

target_names = ['Abdul','Alisha','Bhamini','Dhanvi']
cm = confusion_matrix(test_ds.classes, y_pred )

sns.heatmap(cm, annot=True, cmap='Blues',
xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.show()

print('Classification Report')
print(classification_report(test_ds.classes, y_pred,
target_names=target_names))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 100, 100, 3)       0

 random_zoom (RandomZoom)    (None, 100, 100, 3)       0

 random_rotation (RandomRota  (None, 100, 100, 3)      0
 tion)

 conv2d (Conv2D)             (None, 100, 100, 16)      1216

 max_pooling2d (MaxPooling2D  (None, 50, 50, 16)       0
 )

 batch_normalization (BatchN  (None, 50, 50, 16)       64
 ormalization)

 dropout (Dropout)           (None, 50, 50, 16)        0

 conv2d_1 (Conv2D)           (None, 50, 50, 16)        2320

 max_pooling2d_1 (MaxPooling  (None, 25, 25, 16)       0
 2D)

 dropout_1 (Dropout)         (None, 25, 25, 16)        0

 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 256)               2560256

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 2,564,884
Trainable params: 2,564,852
Non-trainable params: 32
_________________________________________________________________
```

Fig 7.1: Model Summary

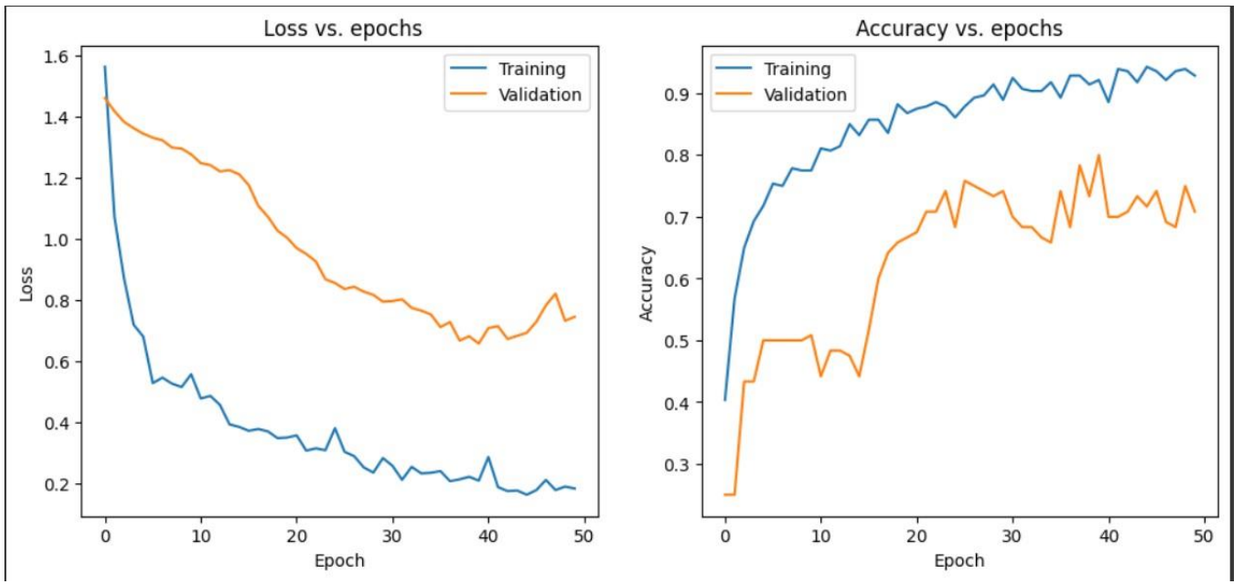


Fig 7.2: Loss and Accuracy graphs of test and validation data

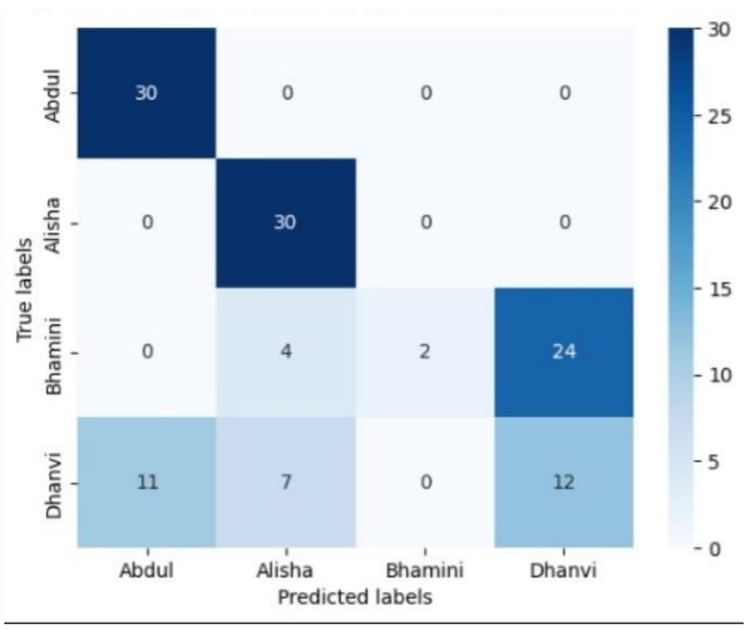


Fig 7.3 : Confusion matrix

CLASSIFICATION REPORT

	Precision	Recall	F1-score	Support
Abdul	0.73	1.00	0.85	30
Alisha	0.73	1.00	0.85	30
Bhamini	1.00	0.07	0.12	30
Dhanvi	0.33	0.40	0.36	30
Accuracy			0.62	120
macro avg	0.70	0.62	0.54	120
Weighted avg	0.70	0.62	0.54	120

7.2 IMPLEMENTING VGG16 AND XCEPTION USING TRANSFER LEARNING

VGG16 →

FUNCTION

```
conv_base=VGG16(
    include_top=False,
    weights="imagenet",
    classes=4,
    classifier_activation="softmax",
    input_shape = (224,224,3)
)
```

CODE

```
model = Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256,activation ='relu'))
model.add(Dense(4,activation ='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False, reduction='auto'),
                                                                     metrics=['accuracy'])
```

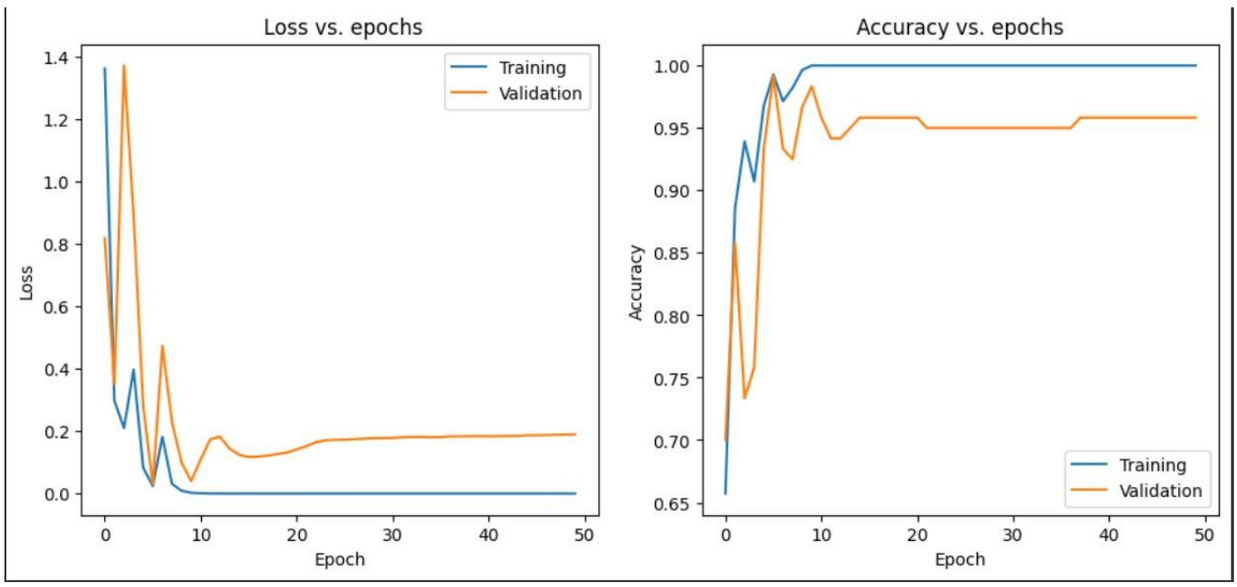


Fig 7.2.1: Loss and Accuracy graphs VGG16

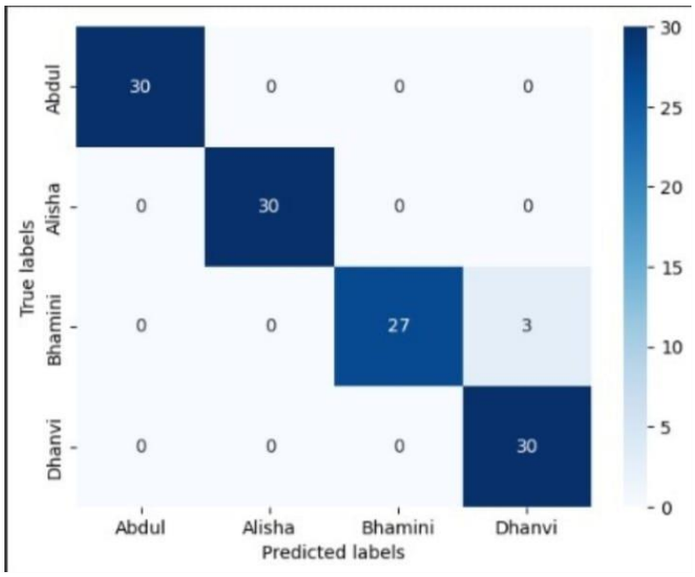


Fig 7.2.2 : Confusion matrix of VGG16

CLASSIFICATION REPORT

	Precision	Recall	F1-score	Support
Abdul	1.00	1.00	1.00	30
Alisha	1.00	1.00	1.00	30
Bhamini	1.00	0.90	0.95	30
Dhanvi	0.91	1.00	0.95	30
Accuracy			0.97	120
macro avg	0.98	0.97	0.97	120
Weighted avg	0.98	0.97	0.97	120

7.2.3: Architecture of VGG16

XCEPTION→

FUNCTION

```
conv_base=Xception(
    include_top=False,
    weights="imagenet",
    classes=4,
    classifier_activation="softmax",
    input_shape = (299,299,3)
)
```

CODE

```
model = Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256,activation ='relu'))
model.add(Dense(4,activation ='sigmoid'))
model.compile(optimizer=tf.keras.optimizers.Adam(1e-4),
                  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False, reduction='auto'),
                                                                     metrics=['accuracy'])
```

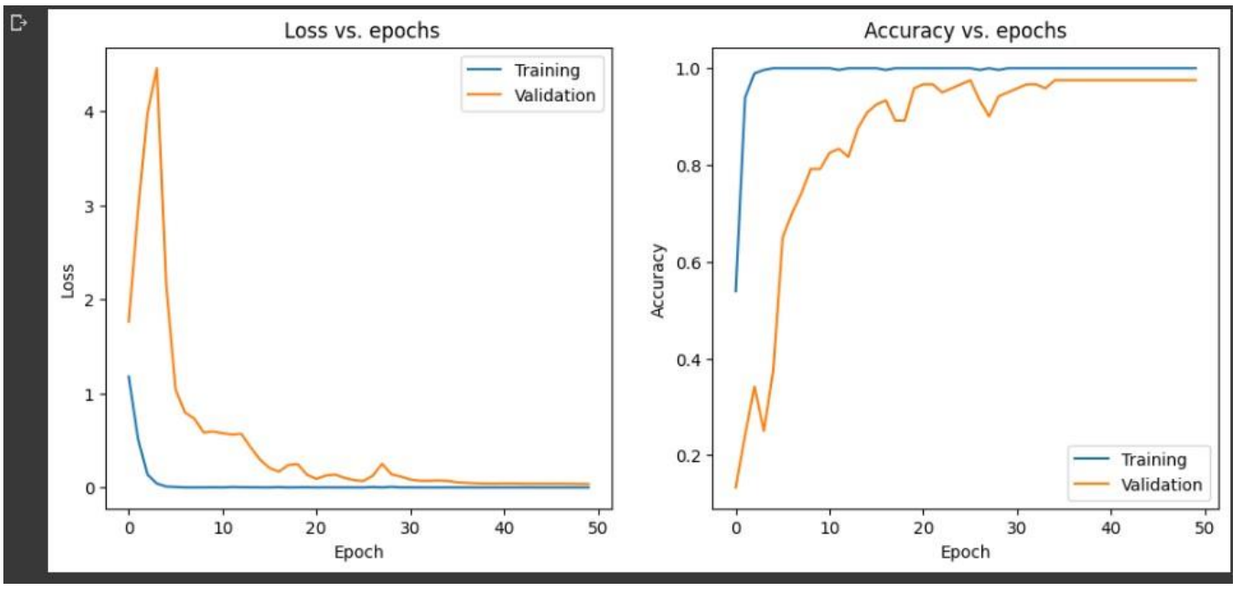


Fig 7.2.1: Loss and Accuracy graphs VGG16

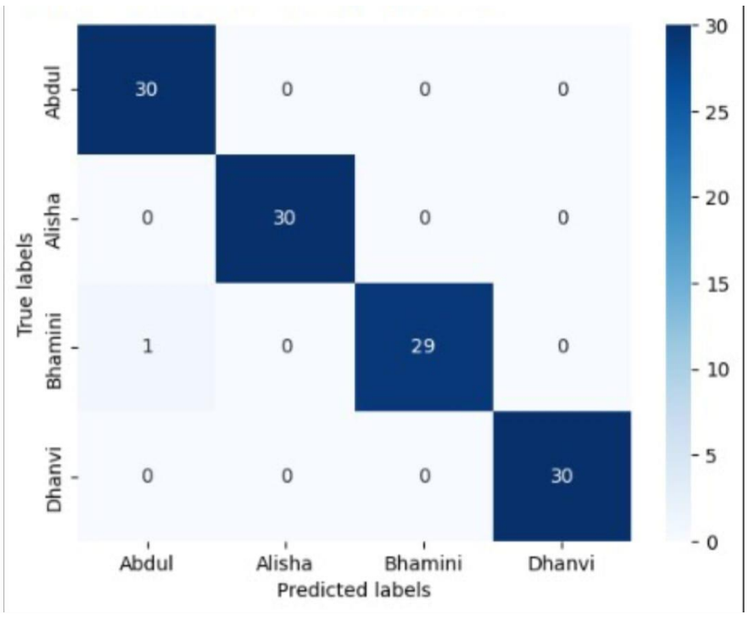


Fig 7.2.2 : Confusion matrix of VGG16

CLASSIFICATION REPORT

	PRECISION	RECALL	F1-SCORE	SUPPORT
Abdul	0.97	1.00	0.98	30
Alisha	1.00	1.00	1.00	30
Bhamini	1.00	0.97	0.98	30
Dhanvi	1.00	1.00	1.00	30
Accuracy			0.99	120
macro avg	0.99	0.99	0.99	120
Weighted avg	0.99	0.99	0.99	120

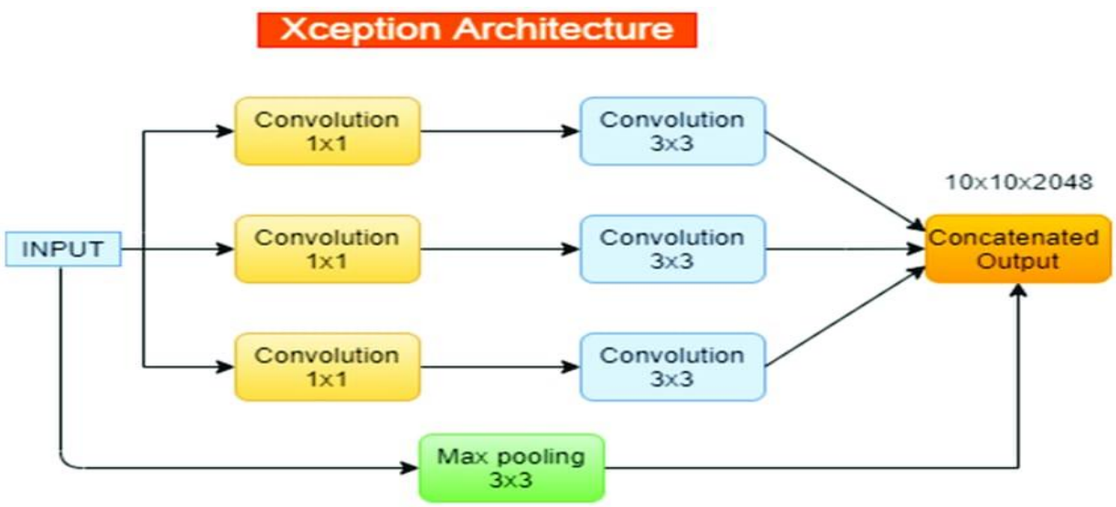


Fig 7.2.3: Architecture of Xception

7.3 COMPARISON

	VGG16	**XCEPTION**
Architecture	Deep Convolution neural network (CNN)	Deep Convolution neural network (CNN)
Depth	16 layers	71 layers
Key Feature	Use of small 3x3 filters with a stride of 1 and padding of 1	Depthwise Separable convolutions
Parameters	More than 138 million	Approximately 22 million
Performance	Achieved success in various computer vision tasks	Achieved the state of the art performance in image classifcation
Size	Relatively Large and computationally expensive	Smaller and more efficient Compared to VGG16
Architectural Difference	Emphasizes depth with multiple stacked convolutional layers	Utilizes depthwise separable Convolutions for improved efficiency
Pretrained models	Pretrained models are widely available and commonly used	Pretrained models are widely available but less commonly used

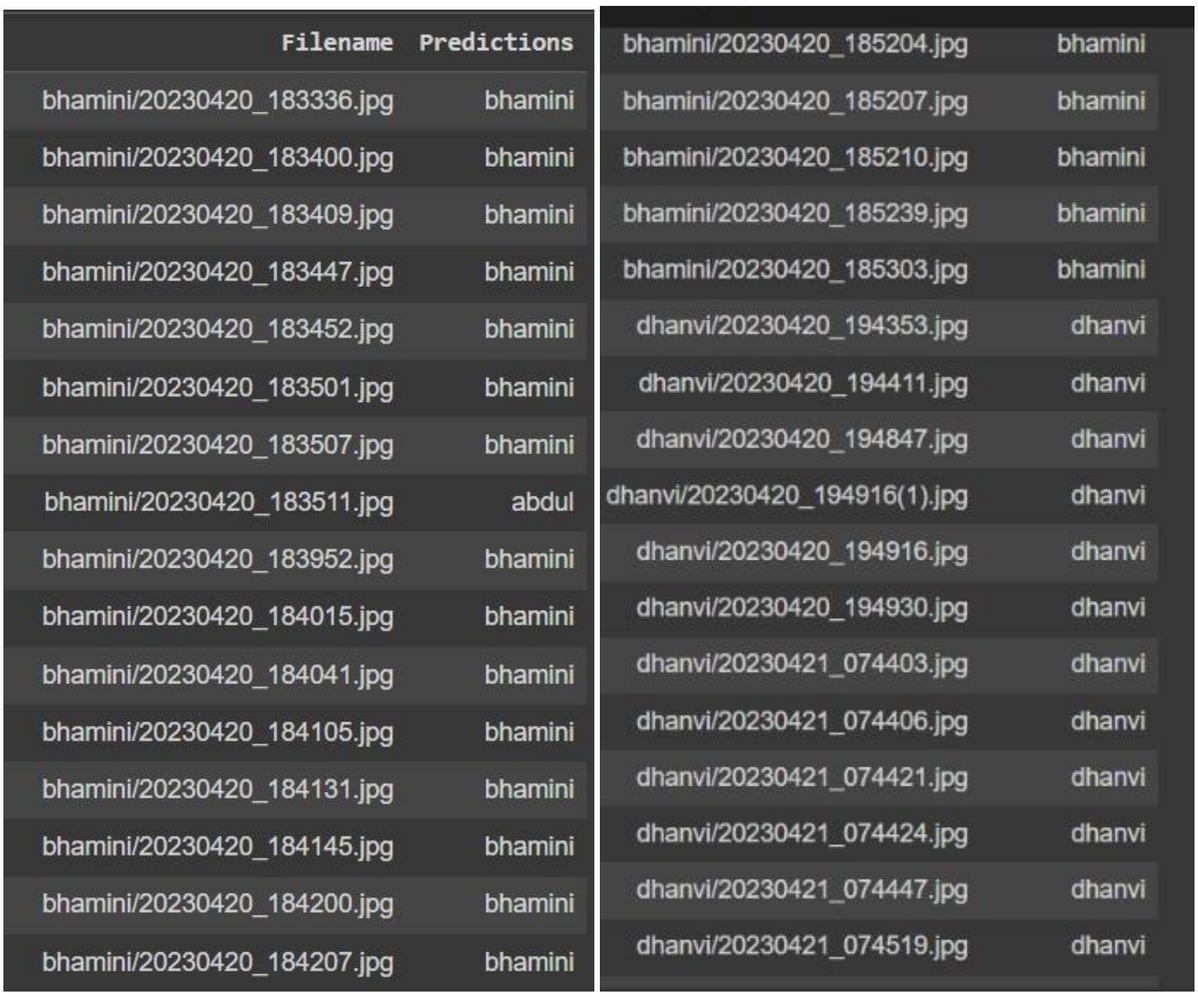
Hence, proceeding with the Xception model which gives more accurate result .

CHAPTER 8

TESTING AND RESULT

This table conveys about the percentages of all the machine learning models used and gives the best model to implement.

MODELS	ACCURACY
Basic CNN	62%
VGG16	97%
Xception	99%



Filename	Predictions
bhamini/20230420_183336.jpg	bhamini
bhamini/20230420_183400.jpg	bhamini
bhamini/20230420_183409.jpg	bhamini
bhamini/20230420_183447.jpg	bhamini
bhamini/20230420_183452.jpg	bhamini
bhamini/20230420_183501.jpg	bhamini
bhamini/20230420_183507.jpg	bhamini
bhamini/20230420_183511.jpg	abdul
bhamini/20230420_183952.jpg	bhamini
bhamini/20230420_184015.jpg	bhamini
bhamini/20230420_184041.jpg	bhamini
bhamini/20230420_184105.jpg	bhamini
bhamini/20230420_184131.jpg	bhamini
bhamini/20230420_184145.jpg	bhamini
bhamini/20230420_184200.jpg	bhamini
bhamini/20230420_184207.jpg	bhamini
bhamini/20230420_185204.jpg	bhamini
bhamini/20230420_185207.jpg	bhamini
bhamini/20230420_185210.jpg	bhamini
bhamini/20230420_185239.jpg	bhamini
bhamini/20230420_185303.jpg	bhamini
dhanvi/20230420_194353.jpg	dhanvi
dhanvi/20230420_194411.jpg	dhanvi
dhanvi/20230420_194847.jpg	dhanvi
dhanvi/20230420_194916(1).jpg	dhanvi
dhanvi/20230420_194916.jpg	dhanvi
dhanvi/20230420_194930.jpg	dhanvi
dhanvi/20230421_074403.jpg	dhanvi
dhanvi/20230421_074406.jpg	dhanvi
dhanvi/20230421_074421.jpg	dhanvi
dhanvi/20230421_074424.jpg	dhanvi
dhanvi/20230421_074447.jpg	dhanvi
dhanvi/20230421_074519.jpg	dhanvi

Fig 8.1: Results of Xception model

CHAPTER 9

CONCLUSION

Signature classification using Convolutional Neural Networks (CNN) is an effective approach that can significantly reduce processing time. By leveraging the power of CNNs, which excel in capturing spatial patterns in images, the classification process becomes more efficient and accurate.

Overall, signature classification using CNNs is a powerful solution that provides both accuracy and efficiency, making it an excellent choice for reducing processing time in signature-related tasks.

REFERENCES

1. A. Karouni, B. Daya, and S. Bahlak, “Offline signature recognition using neural networks approach,” Procedia Computer Science, vol. 3, pp. 155–161, Jan. 2011, doi: 10.1016/j.procs.2010.12.027.

2. P. R. Patil and B. Patil, “A Review - Signature Verification System Using Deep Learning: A Challenging Problem,” International Journal of Scientific Research in Science and Technology, pp. 295–298, Apr. 2021, doi: 10.32628/ijsrset2076

3. T. Raj, S. P. Thangaperumal, and U. Sundarraj, “Signature Verification Using Deep Learning,” T. Raj, S. P. Thangaperumal, and U. Sundarraj, “Signature Verification Using Deep Learning,” ResearchGate, May 2022, [Online].

4. J. a. P. Lopes, B. Baptista, N. Lavado, and M. Mendes, “Offline Handwritten Signature Verification Using Deep Neural Networks,” Energies, vol. 15, no. 20, p. 7611, Oct. 2022, doi: 10.3390/en15207611.