



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

February - May 2023

Report on

“SIGNATURE RECOGNITION BASED ON CNN”

**Master of Technology
in
Computer Science & Engineering**

UE22CS644A – STOCHASTIC MODELS AND MACHINE LEARNING

Submitted by:

MOOKAMBIKA S(PES1PG22CS027) SHALINI T(PES1PG22CS045)

Mookambika.S

Shalini.T

1st Semester M.tech

INTRODUCTION

ABSTRACT

The abstract of a signature verification model typically describes the main objectives, methods, and results of the model. In our model we are using a convolutional neural network (CNN). The proposed model utilizes an architecture to compare two signatures and classify them as genuine or forged. The CNN architecture consists of several convolutional layers with varying filter sizes and pooling layers to capture both local and global features of the signatures. The model is trained on a dataset of genuine and forged signatures, achieving a accuracy on the test set

OBJECTIVE

Our main objective is to develop a signature recognition system that utilizes Convolutional Neural Networks (CNNs) for improved accuracy and robustness. Signature verification and forgery detection is the process of verifying signatures automatically and instantly to determine whether the signature is real or not. The signature in question is then compared to previous samples of that person's signature, which set up the database. In the case of a handwritten signature on a document, the computer needs the samples to be scanned for investigation, whereas a digital signature which is already stored in a data format can be used for signature verification.

Our project consists of various tasks to be performed

First, we need to create a one-page summary using deep learning/ GPT 4 model to create a one-page headline summary. Along with our signatures and upload in G-drive folder.

Summary Page

SRN: PES1PG22CS027

The IPL, or Indian Premier League, is a highly popular annual cricket league created by the Board of Control for Cricket in India (BCCI) in 2007. The league consists of eight franchise teams that compete in a tournament format, with high profile cricketers from around the world participating. The IPL match is watched by millions of people across the world and is played over a period of six to seven weeks, with playoffs consisting of four teams. The final match is played with a white ball and is often held under floodlights in the evening to create a more exciting atmosphere. In addition to cricket, the IPL is also a platform for entertainment, with cheerleaders, music, and celebrity appearances adding to the excitement. The league has also provided an opportunity for young cricketers to showcase their talent at a global level and has helped raise the profile of Indian cricket. The IPL is not just a cricketing spectacle but also a celebration of cricket, unity, and diversity.

Signature - Name - Date

Mookambika S

MOOKAMBIKA S

05/04/2023

SRN: PES1PG22CS045

Vertical takeoff and landing (VTOL) refers to the ability of an aircraft to take off and land vertically without the need for a runway or any external support. This means that the aircraft can lift off directly from the ground or any other flat surface and can land in the same way, without requiring any forward speed or landing gear. The most common types of aircraft that use vertical takeoff and landing capabilities are helicopters, tiltrotor aircraft, and vertical takeoff and landing (VTOL) aircraft. These types of aircraft use different mechanisms to achieve vertical flight.

Helicopters use a rotor blade to generate lift and propulsion, allowing them to hover in place and maneuver in any direction. Tiltrotor aircraft, on the other hand, use rotors that can tilt, allowing them to take off and land like a helicopter, but then transition to a more efficient airplane mode for forward flight. VTOL aircraft typically use jet engines or other propulsion systems to generate lift and forward motion, allowing them to take off and land vertically. aircraft have a number of advantages over traditional fixed-wing aircraft, including the ability to operate in confined spaces, the ability to operate from remote locations or ships, and the ability to quickly respond to emergencies or other time-critical situations. However, they also tend to be more complex and expensive to operate and maintain than traditional aircraft.

They are several solutions that have been proposed to address these challenges in designing an autonomous air travel taxi

Tilt-rotor technology: This involves using rotors that can tilt from a Vertical to a horizontal position, allowing the aircraft to takeoff and land like a helicopter, but fly like a fixed wing aircraft. The V-22 osprey is an example of an aircraft that uses tilt-rotor technology.

Thrust vectoring: This involves using variable geometry nozzles on the engines to direct the thrust in different directions, allowing the aircraft to control its pitch, roll, and yaw. The F-35B Lightning II is an example of an aircraft that uses thrust vectoring.

Lift-fan technology: This involves using a fan or set of fans to provide lift, while the engine provides thrust for forward flight. The Harrier Jump Jet is an example of an aircraft that uses lift-fan technology.

Signature – Name – Date

Shalini T
05/04/2023

Document similarity model Using NLTK

WE are implementing a document similarity model using Natural Language Toolkit(nltk) to estimate the similarities between the documents based on their textual content. This can be useful for tasks such as clustering related documents, identifying content. Document similarity in NLTK is to represent each document as a vector of features such as word frequencies then compute the similarity between vectors using distance metrics such as cosine similarity.

Work Flow of document similarity using NLTK

- Load the documents into Python and preprocess them by removing stop words, punctuation, and other irrelevant information using NLTK's built-in tools.
- Represent each document as a vector of feature values using the bag-of-words model, where each feature corresponds to a unique word in the documents and the value is the frequency of that word in the document.
- Compute the similarity between document vectors using a distance metric such as cosine similarity.

Output of Document similarity

Document Similarity Model.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 11:47 AM

+ Code + Text

Connect

```

!pip install nltk

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: regex<=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2022.10.31)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.2.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.65.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.3)

!pip install PyPDF2

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl (232 kB)
    232.6/232.6 kB 4.0 MB/s eta 0:00:00
Installing collected packages: PyPDF2
Successfully installed PyPDF2-3.0.1

[ ] nltk.download('stopwords')

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
True

[ ] import nltk
nltk.download('punkt')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
True

[ ] import PyPDF2
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

```

Document Similarity Model.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 11:47 AM

+ Code + Text

Connect

```

stop_words = set(stopwords.words('english'))
filtered_tokens = [token for token in tokens if not token.lower() in stop_words]
# Stem the tokens
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
# Join the tokens into a string
preprocessed_text = ' '.join(stemmed_tokens)
return preprocessed_text

def compute_similarity(text1, text2):
    # Preprocess the text
    preprocessed_text1 = preprocess_text(text1)
    preprocessed_text2 = preprocess_text(text2)

    # Convert the preprocessed text into lists of words
    words1 = preprocessed_text1.split()
    words2 = preprocessed_text2.split()

    # Compute the similarity using the Jaccard similarity coefficient
    intersection = set(words1).intersection(set(words2))
    union = set(words1).union(set(words2))
    similarity = len(intersection) / len(union)
    return similarity

# Load the PDF documents
doc1 = load_pdf('/content/PES1PG22CS027(SML).pdf')
doc2 = load_pdf('/content/PES1PG22CS045.pdf')

# Compute the similarity
similarity = compute_similarity(doc1, doc2)

# Print the similarity
print('The similarity between the documents is: {:.2%}'.format(similarity))

```

The similarity between the documents is: 4.29%

SIGNATURE RECOGNITION USING CNN Model

IMPLEMENTATION

We are training our model using CNN

Steps involved in the process

Image acquisition: The signature image is acquired through a scanner or other digital capture device.

Pre-processing: The signature image is pre-processed to enhance its quality, remove noise, and normalize its size, orientation, and position. This step may also include the conversion of the image to grayscale or binarization.

Feature extraction: The signature image is analyzed to extract relevant features that can be used to identify the signature. This may involve techniques such as shape analysis, texture analysis, and directional analysis.

Dataset split: Split the dataset into training and testing sets

Model Training: Training the model using convolution neural networks, using the training set to learn to distinguish between matched or mismatched signatures.

Model Evaluation: Evaluate the performance of the trained model on the testing set using metrics such as accuracy, precision and to find the metrics of the model

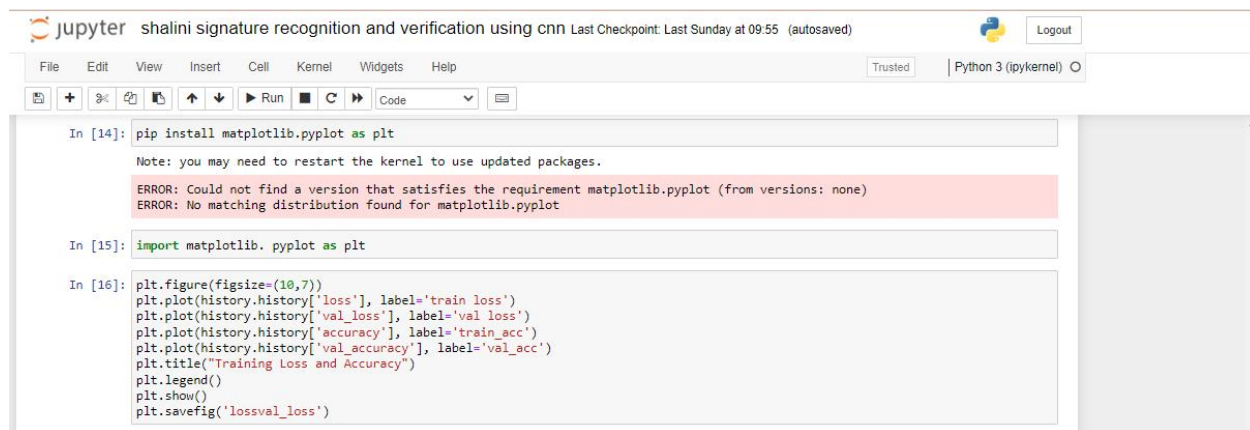
Signature Matching: The signature image is compared with a reference template or database of templates to determine if it matches or not. The matching process may use various methods such as Euclidean distance, correlation coefficients

Decision making: Based on the matching results, a decision is made whether the signature is matched or mismatched

Updating the reference signature: If the signature is authenticated, the reference signature may be updated with the new signature to improve the recognition accuracy in the future.

Metrics

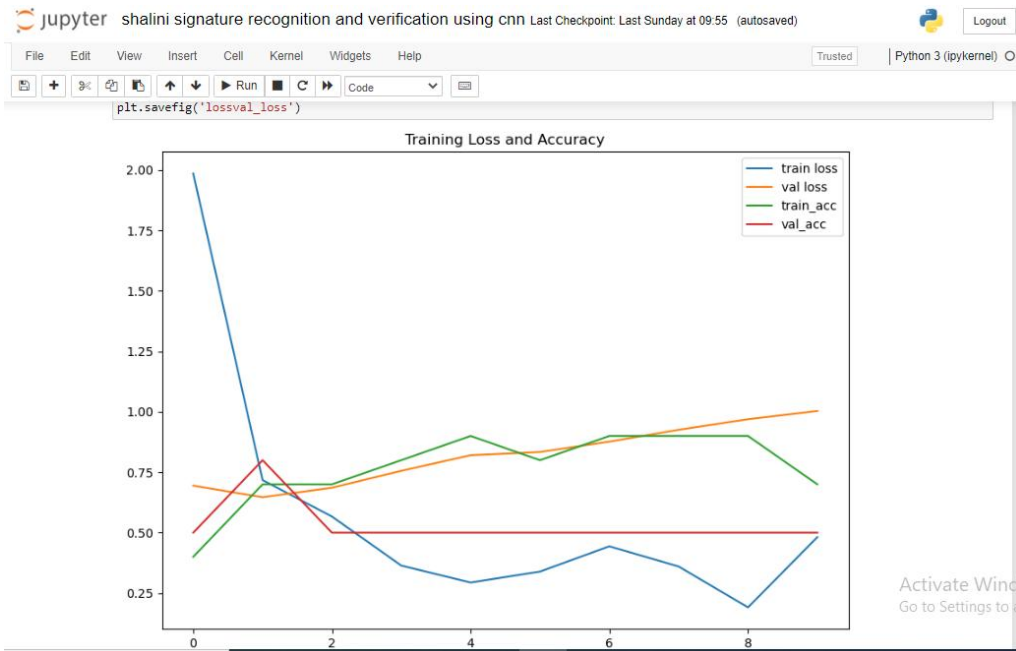
The below code is plotting the training and validation los and accuracy over the epochs of the CNN model training



```
jupyter shalini signature recognition and verification using cnn Last Checkpoint: Last Sunday at 09:55 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
In [14]: pip install matplotlib.pyplot as plt
Note: you may need to restart the kernel to use updated packages.
ERROR: Could not find a version that satisfies the requirement matplotlib.pyplot (from versions: none)
ERROR: No matching distribution found for matplotlib.pyplot

In [15]: import matplotlib.pyplot as plt

In [16]: plt.figure(figsize=(10,7))
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.title("Training Loss and Accuracy")
plt.legend()
plt.show()
plt.savefig('lossval_loss')
```

This code generates a plot of the training and validation loss and accuracy during the training of a machine learning model. It uses the matplotlib library to create a figure with a size of 10 by 7 inches. The history variable is assumed to contain the training history of the model, which includes the training and validation loss and accuracy values for each epoch. The plot function is used to plot the training loss and validation loss against the number of epochs on the x-axis, and the corresponding loss values on the y-axis. Similarly, the training accuracy and validation accuracy are plotted against the number of epochs on the x-axis, and the corresponding accuracy values on the y-axis. The title function is used to set the title of the plot, and the legend function is used to display a legend for the different curves in the plot. Finally, the show function is called to display the plot.

Output Snippets

```
jupyter shalini signature recognition and verification using cnn Last Checkpoint: Last Sunday at 09:55 (autosaved) Logout
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel) O
```

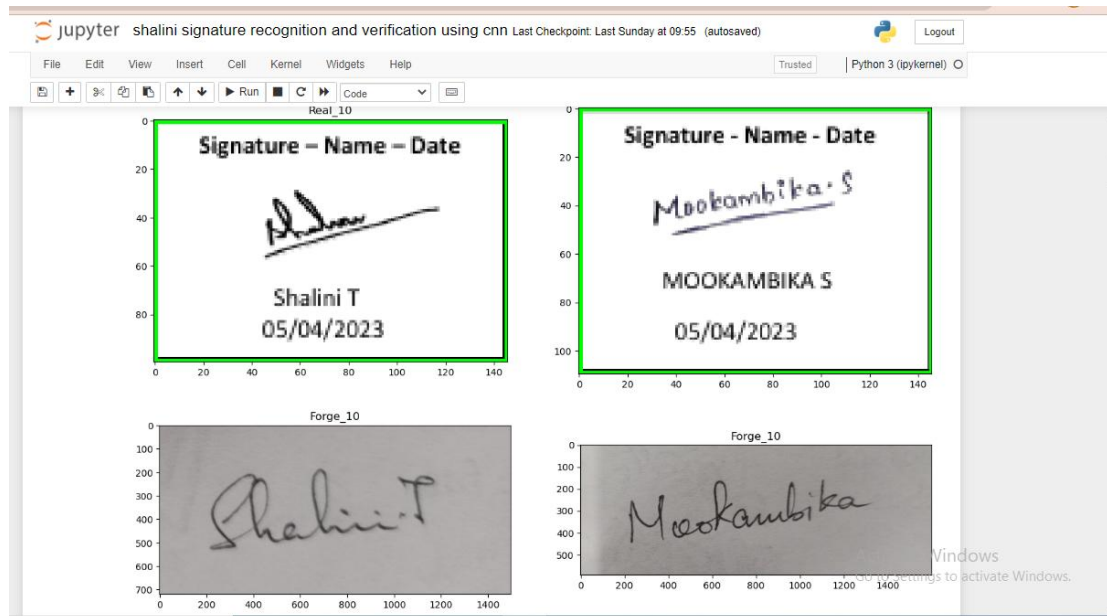
```
In [1]: import pandas as pd
import numpy as np
import skimage.io as sk
from skimage import img_as_ubyte
from skimage.io import imread
from scipy import spatial
from tensorflow.keras.layers import Dense, Flatten, Input, Lambda, MaxPooling2D, Conv2D, Dropout, BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from keras.models import Sequential
from glob import glob
from PIL import Image
import cv2
import matplotlib.pyplot as plt
```

```
In [3]: image1 = sk.imread("/Users/Lenovo/Desktop/SHLreports/SHL-SIGNATURE/genuine_images/shalini-match.png")
image2 = sk.imread("/Users/Lenovo/Desktop/SHLreports/SHL-SIGNATURE/genuine_images/mookambika-match.png")
fig, ax = plt.subplots(1,2, figsize = (15,10))
ax[0].imshow(image1)
ax[0].set_title("Real_10")
ax[1].imshow(image2)
ax[1].set_title("Real_10")
image3 = sk.imread("/Users/Lenovo/Desktop/SHLreports/SHL-SIGNATURE/forged_images/shalini-mismatch.jpeg")
image4 = sk.imread("/Users/Lenovo/Desktop/SHLreports/SHL-SIGNATURE/forged_images/mookambika-mismatch.jpeg")
fig, ax1 = plt.subplots(1,2, figsize = (15,10))
ax1[0].imshow(image3)
ax1[0].set_title("Forge_10")
ax1[1].imshow(image4)
ax1[1].set_title("Forge_10")
```

Activate Windows
Go to Settings to activate Windows.

to search

26°C 23:56 03-05-2023



Jupyter shalini signature recognition and verification using cnn Last Checkpoint: Last Sunday at 09:55 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [8]: from keras.callbacks import EarlyStopping, ReduceLROnPlateau
        early_stop = EarlyStopping(patience=10)
        learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc', patience=2, verbose=1, factor=0.5, min_lr=0.00001)
        callbacks = [early_stop, learning_rate_reduction]

In [9]: train_datagen = ImageDataGenerator(rotation_range=15,
        rescale=1./255,
        shear_range=0.1,
        zoom_range=0.2,
        horizontal_flip=True,
        width_shift_range=0.1,
        height_shift_range=0.1,)

In [10]: train_generator = train_datagen.flow_from_directory('/Users/Lenovo/Desktop/SMLreports/SML-SIGNATURE/training',
        target_size=Image_Size,
        batch_size=22,
        class_mode = 'categorical')

Found 10 images belonging to 2 classes.

In [11]: test_datagen = ImageDataGenerator(rescale=1./255)

In [12]: test_generator = test_datagen.flow_from_directory('/Users/Lenovo/Desktop/SMLreports/SML-SIGNATURE/testing',
        target_size=Image_Size,
        batch_size = 2,
        class_mode='categorical')

Found 10 images belonging to 2 classes.

In [13]: epochs = 10
```

Activate Windows
Go to Settings to activate Windows.

Jupyter shalini signature recognition and verification using cnn Last Checkpoint: Last Sunday at 09:55 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

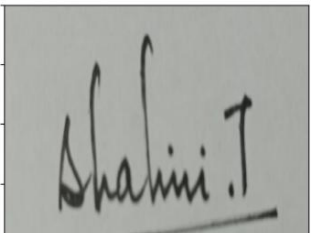
```
else:
    print("The signature is fraud")

The signature is not fraud

In [36]: print(a)
[1]

In [37]: print(img)
<PIL.Image.Image image mode=RGB size=512x512 at 0x22A03342230>

In [38]: plt.imshow(img)
Out[38]: <matplotlib.image.AxesImage at 0x22A06389ed0>
```



Activate Windows
Go to Settings to activate Windows.

LEARNING OUTCOMES

In conclusion, our signature recognition model uses a CNN network to effectively recognize signatures by utilizing convolution layers to extract and learn high-level features of the signature image. In the model we have created two classes matched or unmatched. Our model recognizes if the input signature is being matched or unmatched. We are implementing a document similarity model using Natural Language Toolkit (nltk) to estimate the similarities between the documents based on their textual content. By studying our project signature recognition using machine learning we have understood several ml techniques .