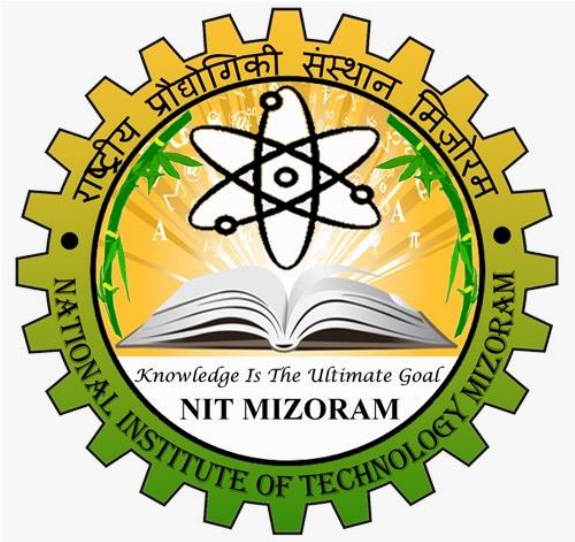
SIGNATURE VERIFICATION USING DEEP LEARNING



Ruzina Haque Laskar

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

MIZORAM: 796012, INDIA

November 2022

SIGNATURE VERIFICATION USING DEEP LEARNING

Reported Submitted to

National Institute of Technology, Mizoram

of

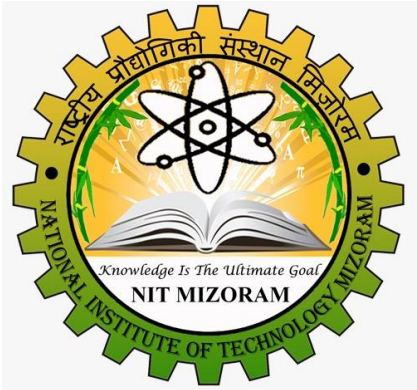
Bachelor of Technology

by

Ruzina Haque Laskar (Enrollment No: BT19CS021)

Supervisor/Supervisors

Ms. Bollapalli Sneha



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

MIZORAM: 796012, INDIA

November 2022

APPROVAL SHEET

This thesis/dissertation/report entitled "**Signature Verification using Deep Learning**" by **Ruzina Haque Laskar** is approved for the degree of Bachelor of technology in Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor (s)

_____

_____

_____

_____

Date: _____

Place: _____

DECLARATION

I declare that this written submission represents my ideas in my own words and where other ideas or words have been include, I have adequately cited and referenced the sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Ruzina Haque Laskar
BT19CS021

NATIONAL INSTITUTE OF TECHNOLOGY, MIZORAM

Department of Computer Engineering

MIZORAM: 796012, INDIA

CERTIFICATE

This is certify that the project entitled

"**Signature Verification by Deep Learning**"

Submitted by

Ruzina Haque Laskar BT19CS021

It is certified that the work contained in the project report titled "Signature verification using Deep Learning" which is submitted by "Ruzina Haque Laskar" in fulfilment of the requirement for the award of Bachelor of Technology Degree in Computer Science and Engineering at NIT Mizoram has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Date:

Dr. Sandeep Kumar Dash
HOD & Asst. Professor
Computer Science and Engg.

Ms. Bollapalli Sneha
Project supervisor
Computer Science and Engg.

ACKNOWLEDGEMENTS

We have taken efforts in this project. However, it might not have been possible without the support, help, and guidance of the many individuals. We extend our sincere thanks to everyone involved in this project directly or indirectly

I would like to express our deepest gratitude and sincere thanks to our supervisor Ms. Bollapalli Sneha, for allowing us to work on this project under her guidance. We would like to thank her for her help, valuable advices, guidanceand full-hand co-operation at every stage of our work.

I would like to thank Mr. Jigar kumar Jhaverbhai Patel for his help, suggestions and full-hand co-operation at every stage of our work.

I would also like to thank the Department Computer Science and Engineeringat the National Institute of Technology Mizoram for providing excellent education, a congenial environment and a research facility.

Last but not the least, I express my cordial honor to my parents, for their unconditional support and encouragement, and for bringing me to this stage of my life. I am indebted to them for their endless love, care, and affection.

Contents

Chapter 1
Introduction

1.1 Signature Verification

Biometric authentication is the process of verifying the identity of individuals based on their unique biological characteristics. It has become a ubiquitous standard for access to high security systems. Current methods in machine learning and statistics have allowed for the reliable automation of many of these tasks (face verification, fingerprinting, iris recognition). Among the numerous tasks used for biometric authentication is signature verification, which aims to detect whether a given signature is genuine or forged.

Signature verification is essential in preventing falsification of documents in numerous financial, legal, and other commercial settings. The task presents several unique difficulties: high intra-class variability (an individual's signature may vary greatly day-to-day), large temporal variation (signature may change completely over time), and high inter-class similarity (forgeries, by nature, attempt to be as indistinguishable from genuine signatures as possible).

There exist two types of signature verification: online and offline. Online verification requires an electronic signing system which provides data such as the pen's position, azimuth/altitude angle, and pressure at each time-step. Superimposed examples of multiple genuine signatures from the same ID, indicating high intraclass variability (from [10]) signing. By contrast, offline verification uses solely 2D visual (pixel) data acquired from scanning signed documents. While online systems provide more information to verify identity, they are less versatile and can only be used in certain contexts (e.g. transaction authorization) because they require specific input systems.

We aim to build an offline signature verification system using a Convolutional Neural Network (CNN). Our paper focuses on building systems trained on data with varying degrees of information, as well as experimenting with different objective functions to obtain optimal error rates.

1.2 Objectives

The aim of this thesis is to propose an approach for verification of online Handwritten signatures.

1. Collecting online handwritten signature via a digital tablet or pen based input device can provide very useful dynamic features such as writing speed, pen orientation and pressure in addition static shape information.

2. For our main training tasks, we used the VGG-16 CNN architecture

3. The layers we have used is : FCL(Fully Connected Layers), ReLU Nonlinearity, Convolution Layers.

Chapter 2

Signature recognition

2.1 Introduction on Signature recognition

Signature recognition and verification involves two separate but strongly related tasks: one of them is identification of the signature owner, and the other is the decision about whether the signature is genuine or forged. Also, depending on the need, signature recognition and verification problem is put into two major classes: (i) online signature recognition and verification systems (SRVS) and (ii) offline SRVS. Online SRVS requires some special peripheral units for measuring hand speed and pressure on the human hand when it creates the signature. On the other hand, almost all off-line SRVS systems relies on image processing and feature extraction techniques.

In on-line signature system, person's signature is extracted from capacitive tablet or PDA that provides x-y coordinates, pressure reading etc. These raw data values are then used to calculate various features. Signature authentication can be obtained by two ways that are static and Dynamic. Static features are independent of time while the dynamic features are time dependent. Dynamic features are extracted using electronic tablet or PDA.

Figure 1. Example of Signature Recognition

2.2 Introduction to Signature Verification

Signature verification systems aim to automatically discriminate if the biometric sample is indeed of a claimed individual. In other words, they are used to classify query signatures as genuine or forgeries. Forgeries are commonly classified in three types: random, simple and skilled (or simulated) forgeries. In the case of random forgeries, the forger has no information about the user or his signature and uses his own signature instead. In this case, the forgery contains a different semantic meaning than the

genuine signatures from the user, presenting a very different overall shape. In the case of simple forgeries, the forger has knowledge of the user's name, but not about the user's signature. In this case, the forgery may present more similarities to the genuine signature, in particular for users that sign with their full name, or part of it. In skilled forgeries, the forger has access for both the user's name and signature, and often practices imitating the user's signature. This result in forgeries that have higher resemblance to the genuine signature, and therefore are harder to detect.

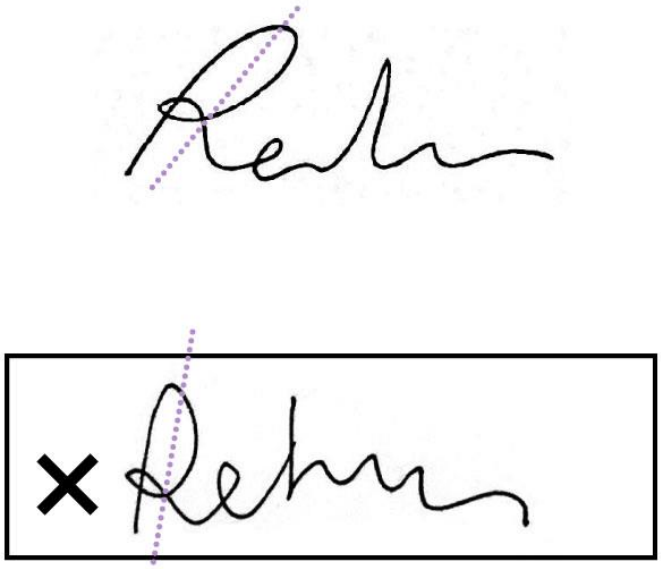


Figure 2. Example of Signature Verification

2.3 Key Challenges of Signature Verification

One of the main challenges for the signature verification task is having a high intra-class variability. Compared to physical biometric traits, such as fingerprint or iris, handwritten signatures from the same user often show a large variability between samples. This problem is illustrated in Figure 1. This issue is aggravated with the presence of low inter-class variability when we consider skilled forgeries. These forgeries are made targeting a particular individual, where a person often practices imitating the user's signature. For this reason, skilled forgeries tend to resemble genuine signatures to a great extent.

Another important challenge for training an automated signature verification system is the presence of partial knowledge during training. In a realistic scenario, during training we only have access to genuine signatures for the users enrolled to the system. During operations, however, we want the system not only to be able to accept genuine signatures, but also to reject forgeries. This is a challenging task, since during training a classifier has no information to learn what exactly distinguishes a genuine signature and a forgery for the users enrolled in the system.

Lastly, the amount of data available for each user is often very limited in real applications. During the enrolment phase, users are often required to supply only a few samples of their signatures. In other words, even if there is a large number of users enrolled to the system, a classifier needs to perform well for a new user, for whom only a small set of samples are available.

2.4 Application of Signature Verification

The majors application of Signature Verification are:

Banking and financial services

Banks and financial service institutions still put their trust in signature verification and use it for customer identification, identity verification, and authorization. Many banks and financial institutions depend on customer signatures to authorize high-value transactions. Some of them may not even be using any technological advantage to verify signatures and entirely depend on human skills.

For example, Bank checks are paper-based payment instruments that are entirely dependent on signature verification to authorize payments. Banks traditionally use human skills to make sure that their signature matches their records. However, despite being very careful, errors do take place in which bank employees might fail to identify signatures. In such an incident, a fraudulent transaction may take place.

Government services

Just like banks and financial services, government services also heavily rely on signatures for personal identification, authentication, and authorization. From lawmakers to government officials and citizens accessing government services have to sign papers to verify their identity or authorize a transaction. All the government services and processes, where signatures are required, can be streamlined with automated signature verification.

Online and mobile banking

Many banking and financial services apps have already been using behavioral biometric means which can track user behaviour to create a profile and keep the user in an authenticated state. On these apps, biometric electronic signature verification can work as a first authentication barrier, which is traditionally done by PIN, password, fingerprint, or face recognition. Since banks and financial service institutions are already known to value user signatures, transaction authorization on mobile apps can be done using signatures instead of PIN, password, selfie, or fingerprint scan.

Today smartphones, ultraportable computers, and tablets come equipped with a touchscreen. These capacitive touchscreen panels can register a slight touch. These touchscreen devices offer an opportunity to use signature verification as a method of identification, authentication, and authorization.

Chapter 3

Implementation of Deep Learning

3.1 Introduction to Deep Learning

Deep learning is a subfield of machine learning that deals with algorithms inspired by the structure and function of the brain. Deep learning is a subset of machine learning, which is a part of artificial intelligence (AI). The "deep" part of deep learning refers to creating deep neural networks. This refers a neural network with a large amount of layers — with the addition of more weights and biases, the neural network improves its ability to approximate more complex functions.

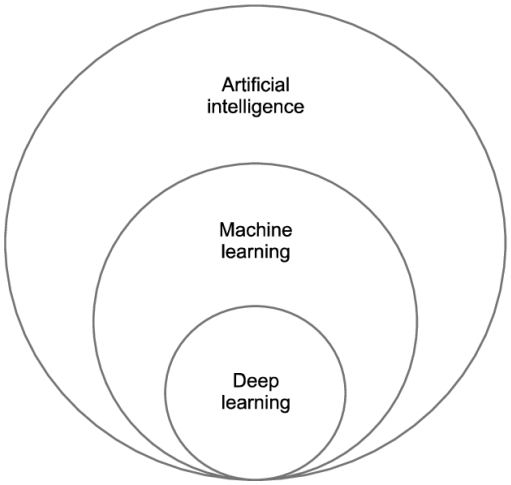


Figure 3. Deep Learning {set}

Deep learning models are capable enough to focus on the accurate features themselves by requiring a little guidance from the programmer and are very helpful in solving out the problem of dimensionality. Deep learning algorithms are used, especially when we have a huge no of inputs and outputs.

Since deep learning has been evolved by the machine learning, which itself is a subset of artificial intelligence and as the idea behind the artificial intelligence is to mimic the human behaviour, so same is "the idea of deep learning to build such algorithm that can mimic the brain".

Deep learning is implemented with the help of Neural Networks, and the idea behind the motivation of Neural Network is the biological neurons, which is nothing but a brain cell.

3.2 Neural Networks

A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development of trading systems.

An ANN usually involves a large number of processors operating in parallel and arranged in tiers. The first tier receives the raw input information -- analogous to optic nerves in human visual processing. Each successive tier receives the output from the tier preceding it, rather than the raw input -- in the same way neurons further from the optic nerve receive signals from those closer to it. The last tier produces the output of the system.

Each processing node has its own small sphere of knowledge, including what it has seen and any rules it was originally programmed with or developed for itself. The tiers are highly interconnected, which means each node in tier n will be connected to many nodes in tier n-1 -- its inputs -- and in tier n+1, which provides input data for those nodes. There may be one or multiple nodes in the output layer, from which the answer it produces can be read.

Artificial neural networks are notable for being adaptive, which means they modify themselves as they learn from initial training and subsequent runs provide more information about the world. The most basic learning model is centered on weighting the input streams, which is how each node weights the importance of input data from each of its predecessors. Inputs that contribute to getting right answers are weighted higher.

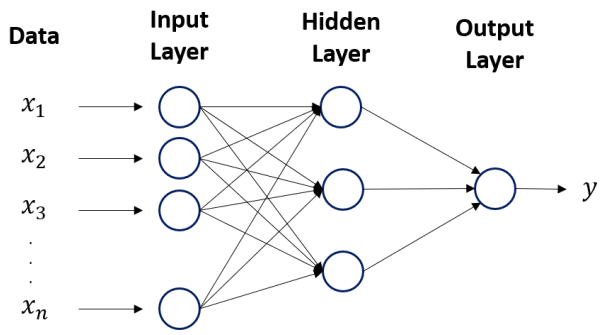


Figure 4. Example of Neural Networks

3.3 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

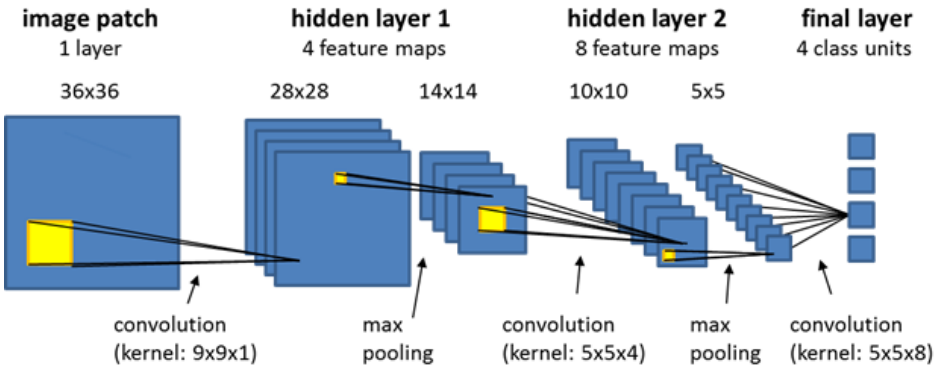


Figure 5. Convolutional Neural Networks Architecture

3.4 Siamese Neural Networks

A Siamese Neural Network is a class of neural network architectures that contain two or more identical subnetworks. 'identical' here means, they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks. It is used to find the similarity of the inputs by comparing its feature vectors, so these networks are used in many applications.

Traditionally, a neural network learns to predict multiple classes. This poses a problem when we need to add/remove new classes to the data. In this case, we have to update the neural network and retrain it on the whole dataset. Also, deep neural networks need a large volume of data to train on. SNNs, on the other hand, learn a similarity function. Thus, we can train it to see if the two images are the same (which we will do here). This enables us to classify new classes of data without training the network again.

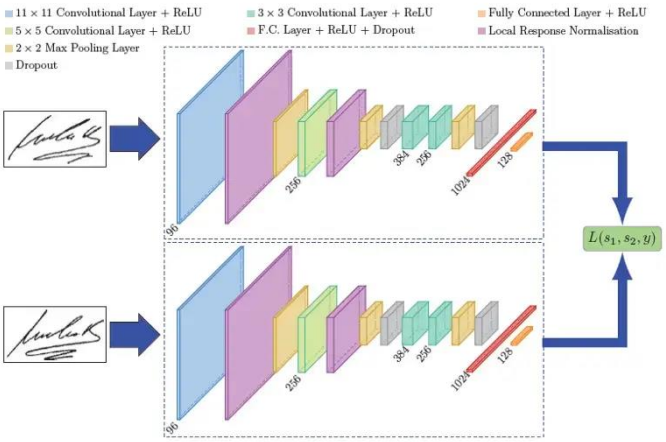


Figure 6. Siamese Neural Network used in Signet

There is two loss in this architecture which is known as: Triplet Loss and Contrastive Loss.

3.5 Deep Learning based Signature Verification

Although deep learning models have shown their superior performance in various areas, they often lack interpretability. That makes them hard to adopt for forensics and other areas that require

rigorous evidence. Therefore, explainable (or interpretable) deep learning methods have attracted more and more attention in recent years. In the field of computer vision, Simonyan et al. [5] proposed a gradient-based visual saliency method to visualize the decision-making process of neural networks. Its main idea is to show the image pixels (a saliency map) which is sensitive to the predictions of a network. This saliency map is obtained by computing the gradient of the class-specific score from a given classifier. The gradient indicates how much the change in a pixel that influences the classifier output.

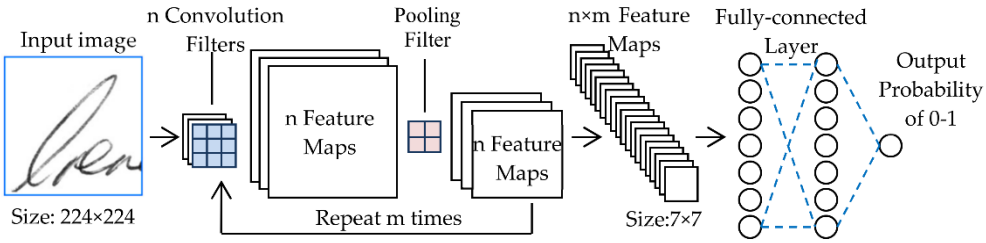


Figure 7. Signature Verification Deep Learning Architecture Model

Chapter 4

Code Implementation

This section includes all the things that we used in our model along with theiroutputs

There are many studies which proposes models to verify signatures using deep learning techniques.

4.1 Datasets

The dataset that I used in my model is from ICDAR 2011 Signature Verification Competition (SigComp2011). There are more than one languages and more than 1000 datasets in one language and from one person.

Figure 8. Some of the Signatures from the dataset.

4.2 Importing Libraries

The very first step is to import all the necessary libraries to the pre-processes to the Signature Verification to build our model.

```
import numpy as np
from PIL import Image
import pickle
import os
import pickle
import cv2
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import tensorflow as tf
import h5py

import keras
from keras.models import  Sequential
from keras.models import Model
from keras.layers import Dense, Dropout, Flatten, Activation, GlobalAveragePooling2D
from keras.callbacks import EarlyStopping
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras import regularizers

from keras.utils import np_utils

# InceptionV3 model imports
from keras.applications.inception_v3 import InceptionV3

from keras.models import load_model
```

4.3 Inception V3 for transfer learning

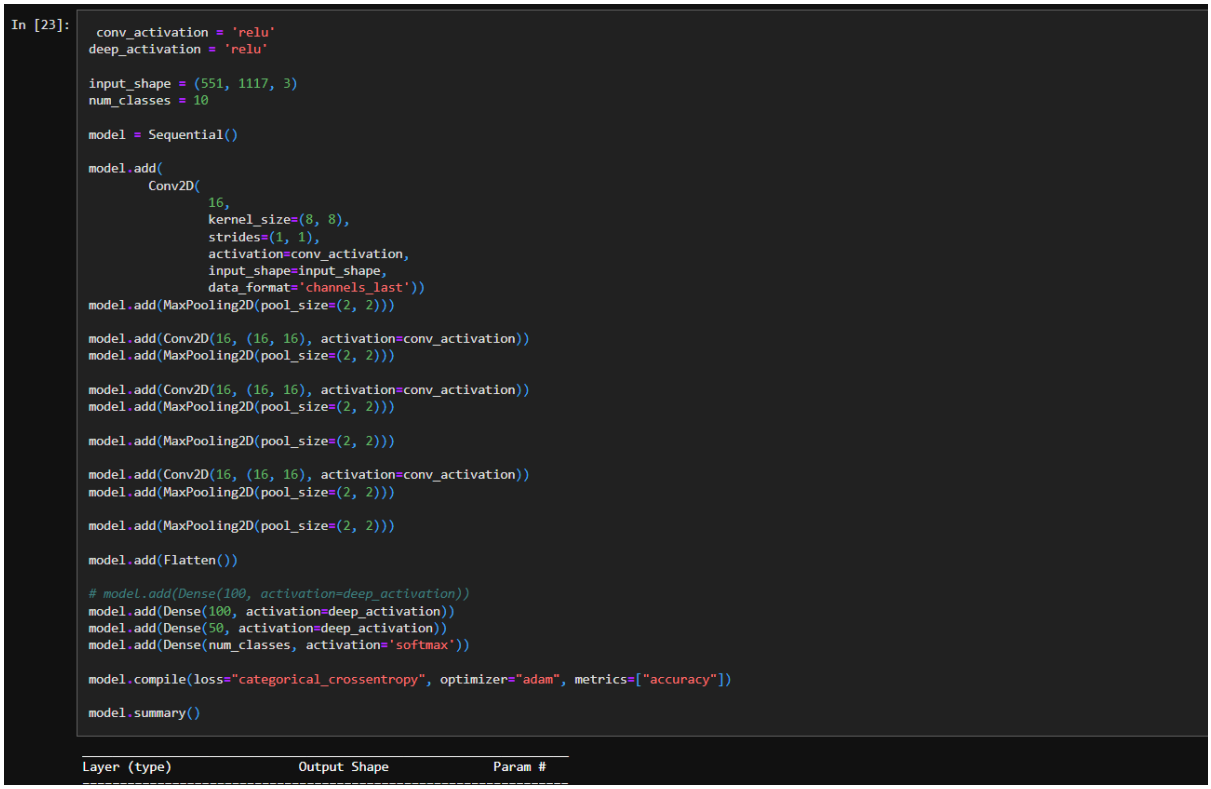
Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

```
In [20]:
# InceptionV3 model imports
from keras.applications.inception_v3 import InceptionV3
base_model = InceptionV3(include_top = False, weights='imagenet', input_shape= (551,1117,3))

Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87916544/87910968 [==============================] - 5s 0us/step

In [0]:
# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# let's add a fully-connected layer
x = Dense(128, activation='relu')(x)
# and a logistic layer -- let's say we have 10 classes
predictions = Dense(10, activation='softmax')(x)

In [22]:
m = Model(inputs=base_model.input, outputs=predictions)

for layer in base_model.layers:
    layer.trainable = False

m.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])
m.summary()
```

Figure 9. Inception V3 code for my model

4.4 Build the Model



```
In [23]:  conv_activation = 'relu'
          deep_activation = 'relu'

          input_shape = (551, 1117, 3)
          num_classes = 10

          model = Sequential()

          model.add(
                  Conv2D(
                          16,
                          kernel_size=(8, 8),
                          strides=(1, 1),
                          activation=conv_activation,
                          input_shape=input_shape,
                          data_format='channels_last'))
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(16, (16, 16), activation=conv_activation))
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(16, (16, 16), activation=conv_activation))
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(16, (16, 16), activation=conv_activation))
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Flatten())

          # model.add(Dense(100, activation=deep_activation))
          model.add(Dense(100, activation=deep_activation))
          model.add(Dense(50, activation=deep_activation))
          model.add(Dense(num_classes, activation='softmax'))

          model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

          model.summary()

          _________________________________________________________________
          Layer (type)                 Output Shape              Param #
          _________________________________________________________________
```

Figure 10. Building the model

4.5 Load and Train the Model

Here, I have loaded and trained the model for more than 7 epochs with a batch size of 32 using fit functions.

load model(provide path accordingly)

go to runtime and change the runtime type to GPU

In [0]:
```
model = load_model('tranfer_20ep_12+dropout.h5')
```

train model(optinal)

In [10]:
```
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs= 20, batch_size= 32, verbose=2)
```

```
Train on 200 samples, validate on 26 samples
Epoch 1/20
 - 36s - loss: 1.3924 - acc: 0.5450 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 2/20
 - 19s - loss: 1.3239 - acc: 0.5950 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 3/20
 - 19s - loss: 1.4369 - acc: 0.4900 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 4/20
 - 19s - loss: 1.3536 - acc: 0.5550 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 5/20
 - 19s - loss: 1.3495 - acc: 0.5250 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 6/20
 - 19s - loss: 1.3491 - acc: 0.5850 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 7/20
 - 19s - loss: 1.3246 - acc: 0.5800 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 8/20
 - 19s - loss: 1.3612 - acc: 0.5100 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 9/20
 - 19s - loss: 1.3102 - acc: 0.5750 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 10/20
 - 19s - loss: 1.3434 - acc: 0.5200 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 11/20
 - 19s - loss: 1.2986 - acc: 0.5850 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 12/20
 - 19s - loss: 1.2407 - acc: 0.5850 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 13/20
 - 19s - loss: 1.2422 - acc: 0.6250 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 14/20
 - 19s - loss: 1.2179 - acc: 0.6150 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 15/20
 - 19s - loss: 1.2225 - acc: 0.6150 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 16/20
 - 19s - loss: 1.2373 - acc: 0.6400 - val_loss: 15.7582 - val_acc: 0.0385
Epoch 17/20
```

Figure 11. Load and train the model

4.6 Output

After using SNN in our model the output we got shown below:

In [11]:
```
import matplotlib.pyplot as plt
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

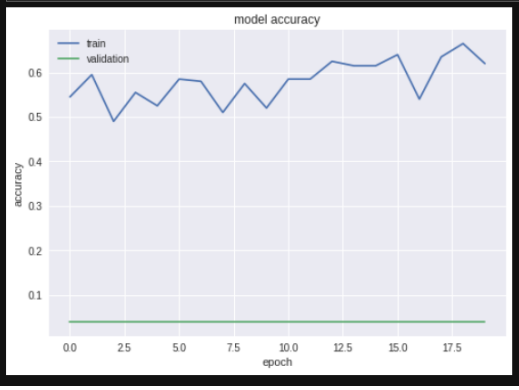


Figure 12. Model Accuracy. In the graph X-axis shows the epoch value and the Y-axis shows the accuracy value

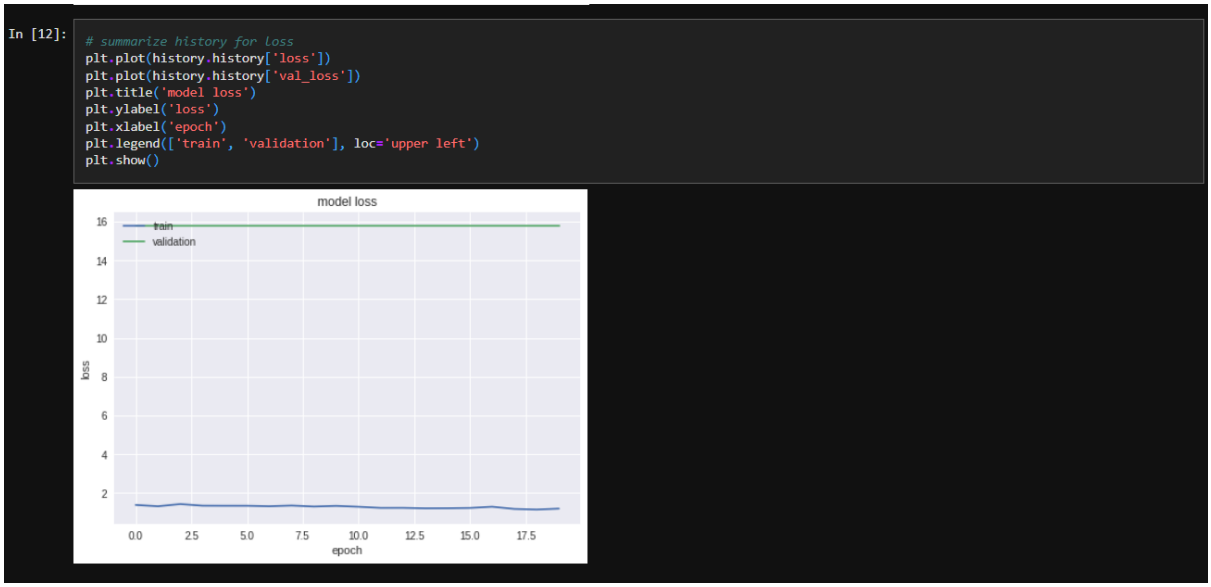


Figure 13. Model Loss. In the graph X-axis shows the epoch value and the Y-axis shows the loss values

Chapter 5

Conclusion

This project proposed a deep learning strategy for the detection and recognition and verification of Signatures which revealed a good performance under some various conditions. This project starts on generic object detection pipelines which provide base architectures for other related tasks. While distinguishing genuine signatures and skilled forgeries remains a challenging task, error rates have dropped significantly in the last few years, mostly due to advancements in Deep Learning applied to the task. Analysing the recent contributions to the field, we can notice that they concentrate in the following categories:

- **Improving classification with limited number of samples** - Given the severe constraints in practical applications, researchers have searched for ways to increase performance in cases where a small number of samples per user is available. In particular, the creation of dissimilarity-based writer-independent solutions, and metric-learning solutions have shown to be promising to address this problem.
- **Augmenting the datasets** - Related to the problem of having low number of samples per user, some researchers have focused in generating synthetic signatures, in order to increase the number of samples available for training.
- **Building model ensembles** - In order to increase classification accuracy, and the robustness of the solutions, some researchers have investigated the creation of both static and dynamic ensembles of classifiers.

Chapter 6

References

- Hameed, M. & Ahmad, Rodina & Mat Kiah, Miss Laiha & Murtaza, Ghulam. (2021). Machine learning-based offline signature verification systems: A systematic review. Signal Processing: Image Communication. 93. 116139. 10.1016/j.image.2021.116139.
- Patil, Ms.Punam & Patil, Bhushan. (2021). A Review - Signature Verification System Using Deep Learning: A Challenging Problem. International Journal of Scientific Research in Science and Technology. 295-298. 10.32628/IJSRSET207632.
- https://arxiv.org/pdf/2002.10119.pdf
- Bailing Zhang. Off-line signature verification and identification by pyramid histogram of oriented gradients. International
- Journal of Intelligent Computing and Cybernetics, 3(4):611–630, 2010.
- Srinivasan, H., Srihari, S.N., Beal, M.J. (2006). Machine Learning for Signature Verification. In: Kalra, P.K., Peleg, S. (eds) Computer Vision, Graphics and Image Processing. Lecture Notes in Computer Science, vol 4338. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11949619_68
- Elizabeth Ann Soelistio, Rafael Edwin Hananto Kusumo, Zevira Varies Martan, Edy Irwansyah, "A Review of Signature Recognition Using Machine Learning", 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), vol.1, pp.219-223, 2021.
- Neha Sharma et al 2021 J. Phys.: Conf. Ser. 1969 012044
- V. Iranmanesh, S. M. S. Ahmad, W. A. W. Adnan, S. Yussof, O. A. Arigbabu, and F. L. Malallah, “Online handwritten signature verification using neural network classifier based on principal component analysis,” The Scientific World Journal, vol. 2014, Article ID 381469, 8 pages, 2014.