JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16 July 2020

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report ✓ | Paper

Name: SAKSHI CHHIMPA Department: CSE Enrolment No 161289

Contact No. 98050-56925 E-mail. sakshi.chhimpa@gmail.com

Name of the Supervisor: Dr. Ravindara Bhatt

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): SIGNET: CONVOLUTIONAL SIAMESE NETWORK FOR WRITER INDEPENDENT OFFLINE SIGNATURE VERIFICATION

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 60
- Total No. of Preliminary pages = 12
- Total No. of pages accommodate bibliography/references = 2

Sakshi

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

SIGNET: CONVOLUTIONAL SIAMESE NETWORK FOR WRITER INDEPENDENT OFFLINE SIGNATURE VERIFICATION

Project report submitted in fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering

By

Sakshi Chhimpa (161289)

Under the supervision of

Dr. Ravindara Bhatt

To

Department of Computer Science & Engineering

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CANDIDATE’S DECLARATION

This is to certify that the work which is being presented in the report entitled “**SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification**” in partial fulfilment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science and Engineering, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to December 2019 under the supervision of **Dr. Ravindara Bhatt** (Assistant Professor, Senior Grade, Computer Science & Engineering Department).The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Sakshi

(Student Signature)

Sakshi Chhimpa, 161289

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature) Ravindara Bhatt

Dr. Ravindara Bhatt

Associate Professor (Senior Grade)

Department of Computer Science & Engineering Dated:

ACKNOWLEDGEMENT

I have put a lot of effort into this project. But this wouldn't have been possible without the kind support and help of many people. I thank all of them sincerely.

I am greatly indebted to my project supervisor **Dr. Ravindara Bhatt** for his guidance and constant supervision as well as providing us with all the information regarding the project titled - **SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification.**

I am also thankful to **Jaypee University of Information Technology** for providing me with all the latest technologies, thus, comforting me with the project.

Again, I owe my profound gratitude to my project guide, for not only helping me with the project but also developing a keen interest in the same during its progress.

Sakshi Chhimpa, 161289

Dated:

TABLE OF CONTENTS

CONTENT	PAGE NO.
DECLARATION	3
ACKNOWLEDGEMENT	4
LIST OF ABBREVIATIONS	v
LIST OF FIGURES	vi vi
ABSTRACT	viii-x
CHAPTER 1: INTRODUCTION	**1-14**
1.1 Introduction	1-3
1.1.1 What is Machine Learning?	3-4
1.1.2 Some machine learning methods	4-6
1.1.3 What is Deep Learning?	6
1.1.4 Examples of Deep Learning at Work	7
1.1.5 Neural Networks in Deep Learning	8
1.1.6 Convolutional Neural Networks	9-10
1.1.6.1 Convolutional Siamese Neural Networks	9
1.2 Problem Statement	10-11
1.3 Objective	11-12
1.4 Methodology of the proposed model	12-14
1.5 Organization	14

CHAPTER 2: LITERATURE SURVEY **15-28**

2.1 TITLE: SigNet: Convolutional Siamese Network for Writer 15-

20 Independent Offline Signature Verification

2.2 TITLE: Deep Learning Specialization 21-28

CHAPTER 3: SYSTEM DEVELOPMENT **29-38**

3.1 Dataset 29

3.2 Model Development 29-38

CHAPTER 4: PERFORMANCE ANALYSIS **39-41**

4.1 Performance analysis of the model 39

4.2 Prediction of Scores 39-41

CHAPTER 5: CONCLUSIONS **42**

5.1 Conclusion 42

5.2 Future Scope 42

REFERENCES 43

LIST OF ABBREVIATIONS

ASV	Automated Signature Verification
CNN	Convolutional Neural Network
AI	Artificial Intelligence
RNN	Recurrent Neural Networks
FCL	Fully Connected Group
VGG	Visual Geometry Group
OS	Operating System
CPU	Central Processing Unit
GPU	Graphical Processing Unit

LIST OF FIGURES

Figure No.	Title	Page No.
Figure 1.1	Evolution of Technologies	4
Figure 1.2	Machine Learning Types	4
Figure 1.3	Supervised vs Unsupervised Learning	6
Figure 1.4	A neuron vs a Neural Network	8
Figure 1.5	Simple Neural Network and DL Neural Network	8
Figure 1.6	Block Diagram of Convolutional Siamese Neural Network	10
Figure 1.7	Example of Genuine and Forged Signatures	12
Figure 1.8	Siamese Network Explanation I	14
Figure 2.1	A pair of genuine (top left) and forged (bottom left) signatures, and corresponding response maps with five different filters that have produced higher energy activations in the last convolution layer of SigNet.	16
Figure 2.2	Siamese Network Explanation II	18
Figure 2.3	Siamese Network Explanation III	20
Figure 2.4	Convolutional Neural Network Explanation	21
Figure 2.5	Layers in CNN	22
Figure 2.6	Pooling Layer in CNN	23
Figure 2.7	ConvNet Configuration	25
Figure 2.8	VGG16 Explanation	27
Figure 2.9	ResNet Explanation	28
Figure 3.1	Genuine vs Forgery	29

Figure 3.2	Code Snippet #1	30
Figure 3.3	Code Snippet #2	31
Figure 3.4	Code Snippet #3	32
Figure 3.5	Code Snippet #4	33
Figure 3.6	Code Snippet #5	34
Figure 3.7	Code Snippet #6	35
Figure 3.8	Code Snippet #7	35
Figure 3.9	Code Snippet #8	36
Figure 3.10	Code Snippet #9	37
Figure 3.11	Code Snippet #10	37
Figure 3.12	Code Snippet #11	38
Figure 3.13	Code Snippet #12	38
Figure 4.1	Code Snippet #13	40
Figure 4.2	Code Snippet #14	40
Figure 4.3	Code Snippet #15	41
Figure 4.4	Code Snippet #16	41

ABSTRACT

Signature is one of the most popular and commonly accepted biometric hallmarks that has been used since the ancient times for verifying different entities related to human beings, such as documents, forms, bank checks, individuals, etc. Therefore, signature verification is a critical task.

Signature verification is a sort of software system that compares signatures and checks for genuineness. This protects time and energy and helps to forestall human error throughout the signature method and lowers probabilities of fraud within the method of authentication. The software system generates a confidence score against the signature to be verified. Too low of a confidence score suggests that the signature is presumably a forgery.

Signature verification software system has currently become light-weight, fast, versatile and additional reliable with multiple choices for storage, multiple signatures against one ID and an enormous information. It will mechanically explore for a signature inside a picture or file.

Signature fraud isn't perpetually evident to human operators. It may be troublesome for the human eye to accurately determine fallacious signatures, because the individualintricacies of every signature vary whenever someone signs. For businesses dependent on signatures, Signature Verification is very helpful, quicker, more cost effective and correct than humans—and it habitually outperforms similar solutions in client benchmarks.

Automated Signature Verification (ASV) targets offline signature matching, that is important in varied banking operations like inward cheque clearing, manual fund transfer, and outward payment. Rather than using operators to perform manual signature comparisons on all transactions daily, the ASV engine mechanically performs this task.

The uniqueness of a signature is outlined by its collective characteristics. Signature comparisons search out the variations in 2 signatures compared aspect by aspect. The comparison of their collective properties and variations reveals the degree of potential for forgery.

Signature verification can be done both online and offline.
Capturing **online signature** needs an electronic writing pad together with a stylus, which can mainly record a sequence of coordinates of the electronic pen tip while signing.
Offline signature is usually captured by a scanner or any other type of imaging devices, which basically produces two-dimensional signature images.

Offline signature verification is one of the most challenging tasks in biometrics and document forensics.
Unlike other verification problems, it needs to model minute but critical details between genuine and forged signatures, because a skilled falsification might only differ from a real signature by some specific kinds of deformation. This verification task is even harder in writer independent scenarios which is undeniably fiscal for realistic cases.

This project models an offline writer independent signature verification task with a convolutional Siamese network. Siamese networks are twin networks with shared weights, which can be trained to learn a feature space where similar observations are placed in proximity. This is achieved by exposing the network to a pair of similar and dissimilar observations and minimizing the Euclidean distance between similar pairs while simultaneously maximizing it between dissimilar pairs. Experiments conducted on crossdomain datasets emphasize the capability of our network to handle forgery in different languages (scripts) and handwritten styles. Moreover, our designed Siamese network, named SigNet, provided better results than the state-of-the-art results on most of the benchmark signature datasets.

CHAPTER-1 INTRODUCTION

1.1 INTRODUCTION

Online check frameworks regularly perform higher than their disconnected counter segments as a result of the flexibly of reciprocal data like stroke request, composing Speed, pressure, and so forth. Be that as it may, this improvement in exhibitions comes at the cost of requiring an uncommon equipment for recording the pen-tip direction, rising its framework cost and lessening the genuine application projections.

There are a few situations where confirming offline signature is the exclusively probability like check managing and archive confirmation. Due to its more extensive application zone, this project tries to specialize in the more difficult task- automatic offline signature verification.

Offline signature verification can be addressed with- writer dependent and, writer independent approaches

The writer independent situation is best over writer subordinate methodologies, concerning a working framework, a writer subordinate framework should be refreshed (retrained) with each new essayist (underwriter). For a customer based generally framework, similar to bank, where consistently new clients will open their record this causes immense expense. While, in writer free case, a conventional framework is worked to demonstrate the disparity among the genuine and manufactured signatures. Preparing a signature check framework under a writer free situation, partitions the open endorsers into train and test sets. For a chose underwriter, signatures are coupled as comparative (real, certified) or unique (real, produced) sets. From all the tuples of one underwriter, equivalent scope of tuples comparative and divergent sets are stochastically chosen for balance of the quantity of occasions. This system is applied to all the endorsers in the train and test sets to build the preparation and test models for the classifier.

In such manner a signature verifier can be productively displayed by a Siamese system that comprises of twin convolutional systems tolerating 2 different signature pictures which are coming from the tuples which can be either alike or different. The constituting convolutional neural networks (CNN) are further, above they are connected through a cost function, which performs the computation of a distance metric between the most elevated level element outline on all sides of the system

In twin networks the parameters are shared, as a result two same pictures can't be mapped by their individual systems to totally different areas in highlight space since each system figures a similar capacity.

Many of hand-made highlights for disconnected signature check assignments mull over the worldwide signature picture for include extraction, for example, square codes, wavelet and Fourier arrangement and so forth. Some different strategies consider the geometrical and topological qualities of nearby characteristics, similar to position, digression heading, mass structure, associated part and ebb and flow. Projection and form based systems are very in style for disconnected signature check. Other than above discussed strategies, methods generalized on course profile, surroundedness highlights, framework based systems, procedures supporting geometrical minutes, and surface based alternatives have likewise picked up prevalence in signature confirmation task.

Few systematic methods which contemplate the relations among nearby highlights are additionally investigated for the comparative undertaking. On the contrary hand, Siamese like systems are a lot of mainstream for various confirmation undertakings, for example, online signature check, face confirmation and so forth. Additionally, it has likewise been utilized for one-shot learning, just as for sketch-based picture recovery task. Apparently, till now, our motivation is derived from the fact that the convolutional Siamese network has not been used for offline signature verification.

This project is totally based on the use of Convolutional Siamese network called, SigNet, which has been used for offline signature verification. In the network, in which distinction to alternative ways depending upon hand crafted attributes, has the adaptability to demonstrate conventional signature imitation strategies and plenty of alternative connected properties that wraps minute irregularity in signatures from the preparation information.

So as to get familiar with the Convolutional Siamese Network, the terms mentioned below need to be discussed.

1.1.1 What is Machine Learning?

AI is basically the utilization of man-made reasoning (AI) that gives frameworks the capacity to learn all alone and improve for a fact without being expressly modified. AI centres around the advancement of PC programs that can get to information and use it learn for themselves.

The way toward learning starts with perceptions or information, similar to models, direct understanding, or guidance, so as to search for designs in information and settle on better choices later on dependent on the models. The essential point is to permit the PCs adapt consequently without human intercession or help and change activities likewise**.**

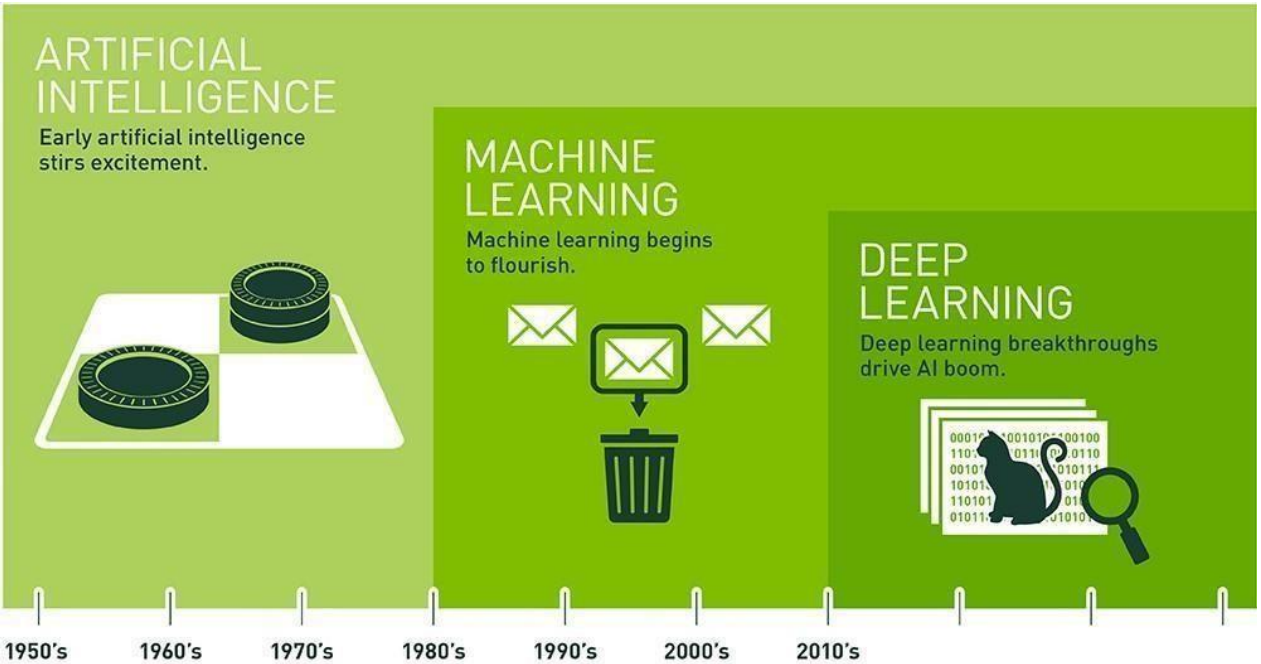

Fig 1.1: Evolution of Technologies

1.1.2 Some machine learning methods

Machine learning algorithms are unit typically categorized as **supervised** or **unsupervised.**

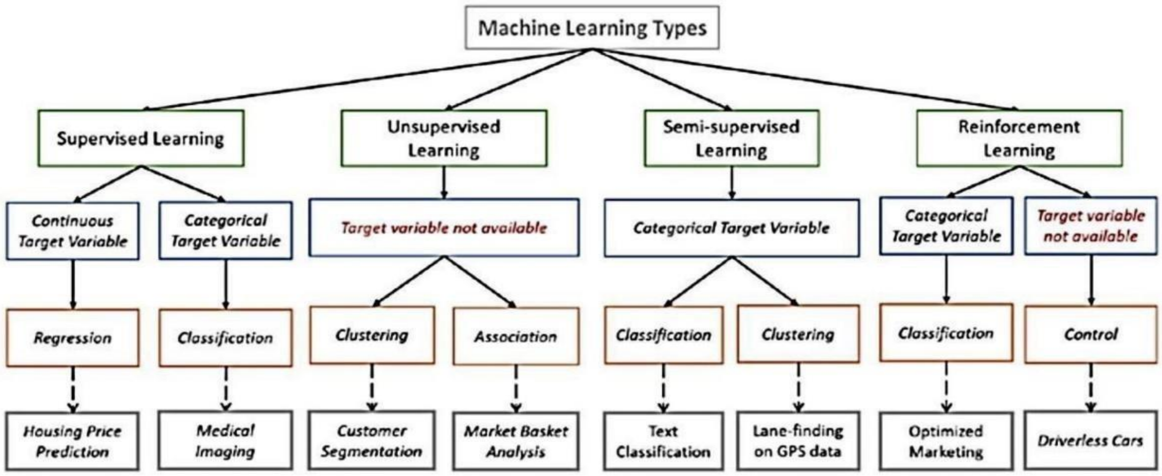

Fig 1.2: Machine Learning Types

•Supervised AI calculations can apply what has been realized inside the past to new data utilizing named guides to anticipate future occasions. Starting from the examination of a known preparing dataset, the learning calculation creates a deduced capacity to make expectations about the yield esteems. The framework can give focuses to any new contribution after adequate preparing. The learning calculation can likewise contrast its yield and the right, planned yield and discover blunders so as to alter the model appropriately.

•In differentiate, solo AI calculations are utilized when the data used to prepare is neither ordered nor signatured. Solo learning examinations how frameworks can surmise a capacity to portray a concealed structure from unlabelled information. The framework doesn't make sense of the correct yield, yet it investigates the information and can attract deductions from datasets to depict concealed structures from unlabelled information.

•Semi-directed AI calculations fall some place in the middle of regulated and unaided learning, since they utilize both named and unlabelled information for preparing — normally a modest quantity of named information and a lot of unlabelled information. The frameworks that utilization this technique can extensively improve learning precision. Typically, semi supervised learning is picked when the obtained signatured information requires talented and applicable assets so as to prepare it/gain from it. Something else, gaining unlabelled information by and large doesn't require extra assets.

•Reinforcement AI calculations is a learning strategy that associates with its condition by delivering activities and finds blunders or rewards. Experimentation search and deferred reward are the most pertinent attributes of support learning. This strategy permits machines and programming operators to naturally decide the perfect conduct inside a particular setting so as to expand its exhibition.

Straightforward prize input is required for the operator to realize which activity is ideal; this is known as the fortification sign.

Unsupervised Learning **8upervlsea Learning**

Fig 1.3: Supervised vs Unsupervised Learning

1.1.3 What is Deep Learning?

Deep learning is really a subset of AI. It in fact is AI and capacities similarly however it has various abilities.

The principle distinction among deep and machine learning is, AI models become better continuously however the model despite everything needs some direction. In the event that a machine learning model returns a wrong forecast, at that point the designer needs to fix that issue unequivocally however on account of deep learning, the model does it without anyone else. Programmed vehicle driving framework is a genuine case of deep learning.

1.1.4 Examples of Deep Learning at Work

Deep learning applications are used in industries from automated driving to medical devices.

Computerized Driving: Automotive analysts are utilizing deep learning out how to consequently identify items, for example, stop signs and traffic lights. What's more, deep realizing is utilized to identify people on foot, which helps decline mishaps.

Aviation and Defense: Deep learning is utilized to distinguish objects from satellites that find regions of premium, and recognize sheltered or perilous zones for troops.

Clinical Research: Cancer specialists are utilizing deep learning out how to consequently distinguish malignant growth cells. Groups at UCLA manufactured a propelled magnifying lens that yields a high-dimensional informational index used to prepare a deep learning application to precisely distinguish malignant growth cells.

Modern Automation: Deep learning is assisting with improving labourer security around overwhelming hardware via naturally identifying when individuals or articles are inside a hazardous separation of machines.

Gadgets: Deep learning is being utilized in robotized hearing and discourse interpretation. For instance, home help gadgets that react to your voice and realize your inclinations are fueled by deep learning applications.

1.1. 5 Neural Networks in Deep Learning

Neural systems are multi-layer systems of neurons that are utilized to group things, make expectations, and so forth. They fundamentally are a lot of calculations, displayed freely after the human mind, that are intended to perceive designs. They decipher tangible information through a sort of machine observation, signaturing or bunching crude info. The examples they perceive are numerical, contained in vectors, into which all certifiable information, be it pictures, sound, content or time arrangement, must be deciphered.

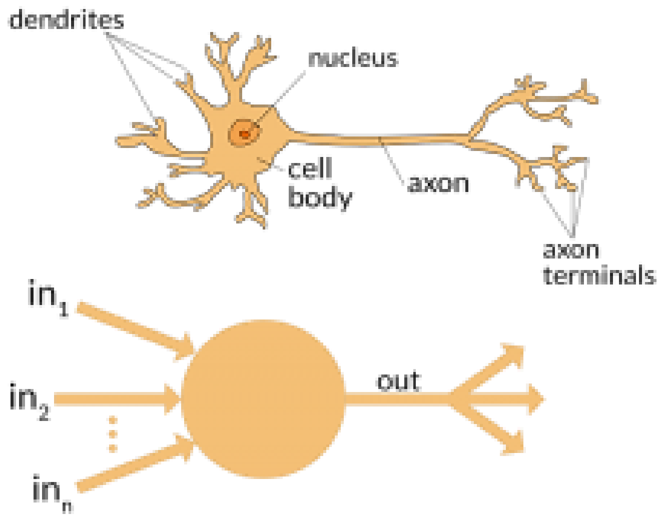

Fig 1.4: A neuron vs a Neural Network

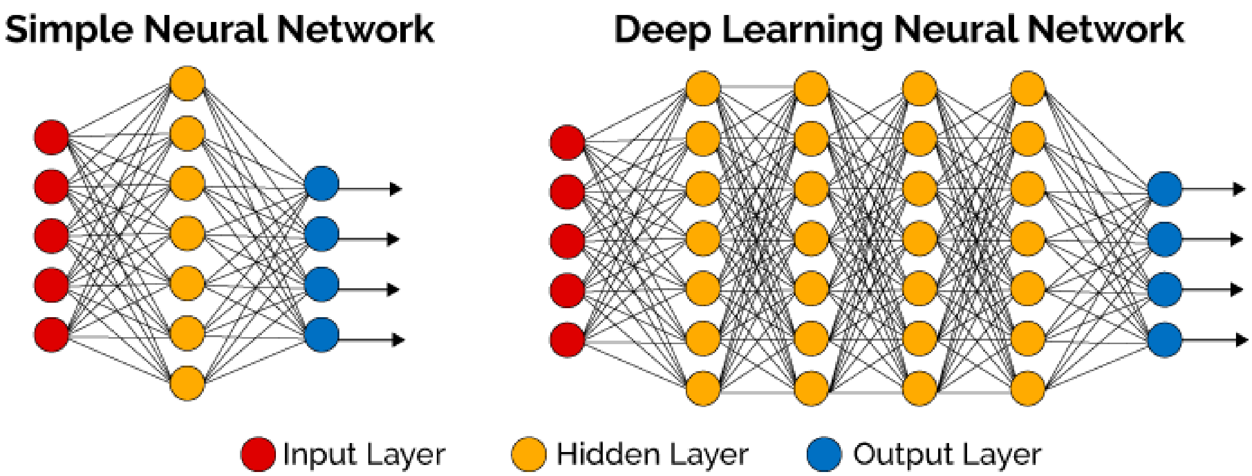

Fig 1.5: Simple Neural Network and DL Neural Network

1.1.6 Convolutional Neural Networks

In deep learning, a convolutional neural system is a class of deep neural systems, most usually applied to dissecting visual symbolism. They are otherwise called move invariant or space invariant fake neural systems, in view of their common loads design and interpretation invariance attributes

A Convolutional Neural Network (ConvNet/CNN) takes in an info picture, dole out significance (learnable loads and predispositions) to different angles/questions in the picture and have the option to separate one from the other. The pre-preparing required in a ConvNet is a lot of lower when contrasted with other grouping calculations.

1.1.6.1 Convolutional Siamese Neural Network

A Convolutional Siamese Neural Network is a kind of system that utilizes two CNN's with similar loads while working couple on two distinctive information vectors to register tantamount yield vectors. Regularly one of the yield vectors is precomputed, subsequently shaping a pattern against which the other yield vector is analysed.

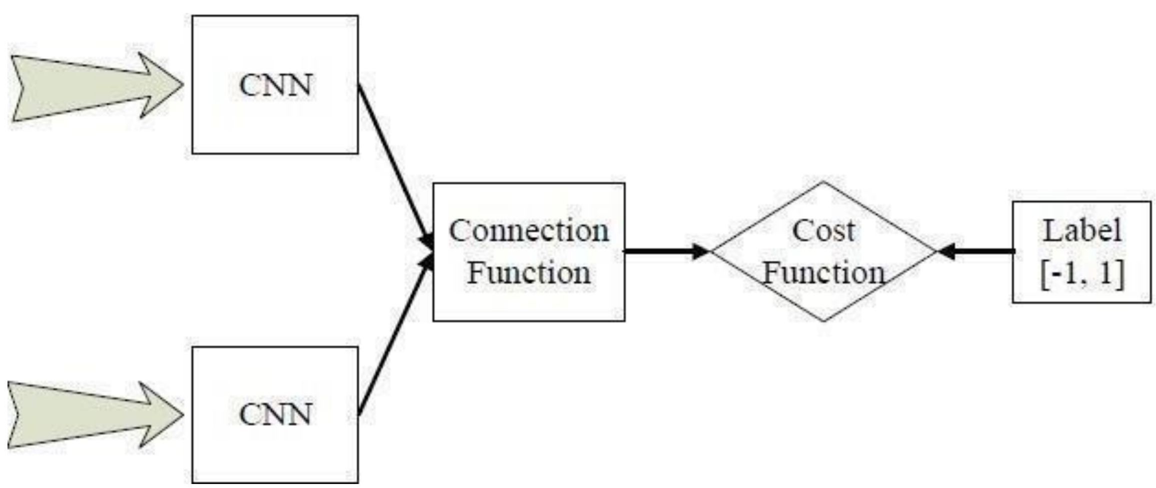

Fig 1.6: Block Diagram of Convolutional Siamese Neural Network

1.2 PROBLEM STATEMENT

For some certifiable individual recognizable proof, signature can be utilized for individual ID. It is utilized for verification or finishing up report. So as to diminish cheats in banks, signature confirmation is a lot of significant. It builds exactness and proficiency.

Since, this verification of signature is getting common day by day and most of the organizations are using this, it would be very beneficial if it could be made quite fast and less troublesome.

Also, with increase in data day by day, it is very much necessary to make offline signature verification a fast-automated process rather than performing it manually. This project puts an extra edge in making offline signature verification more precise and accurate, fast and even a smoother process, by using the modern technologies, i.e. the deep learning architecture and the convolutional Siamese neural networks (SigNet) for verifying a person's identity by his/her signature. Apart from this the project is successful in giving people an idea of how convolutional Siamese networks can be utilized in a variety of applications and thus, making everything even smarter.

1.3 OBJECTIVE

Banking and other significant areas in India have quickly received more up to date advances and computerized channels, with the hidden goal of expanding impressions and incomes. Additionally, client inclinations are moving towards advanced stages. There is a discernment, however, that the appropriation of cutting edge security rehearses has not stayed up with the pace of development of centre business empowering innovation. While in contrast with a few different divisions, banks are certainly observed to be increasingly proactive in contributing and improving security practice, such measures may at present be deficient considering the difficulties with the conventional way to deal with IT security are:

1. Proliferation of assault vectors and improved assault surface.

2. Proliferation of advanced and moving client inclination.

3. Sophistication of danger entertainers and upgraded focusing of banks.

4. Banking progressively working as a 'limit less' environment

A change in outlook has as of late been seen in assaults abusing the source, conduct, thought processes and vectors. This shows the customary multilayered protection that banks as of now have isn't satisfactory. Internationally, there is an ascent in digital security occurrences and a few of them have been enormous scope breaks, cheats and heists. The effect of such penetrates doesn't end with genuine money related misfortune at the same time, much of the time, can likewise conceivably dissolve generous brand esteem. RBI has made a stride the correct way by understanding the inborn requirement for banks to fortify their digital security act in the wake of the undeniably complex nature and quantum of assaults.
Keeping in mind these increasing threat issues with time, this project which is based on offline signature verification, i.e. the scanned images of signatures, aims at increasing the security in banking environment and in many other areas using convolutional Siamese neural networks.
Although similar type of attempt to use Siamese network for signature verification has been made in the past, but this project aims at much better results and accuracy

by making necessary changes in the hidden layers and their configuration, filter size and many other essential parameters.

1.4 METHODOLOGY OF THE PROPOSED MODEL

The project involves various steps for the successful execution and the implementation of the model as mentioned below:

Data Collection

To implement this model a dataset of signatures of different persons has been made. The dataset contains the signatures of 17 persons. Here, for all the entries, 5 genuine and 10 forged signatures are given. This results in 17 × 5 = 80 genuine and 17 × 10 = 170 forged signatures.

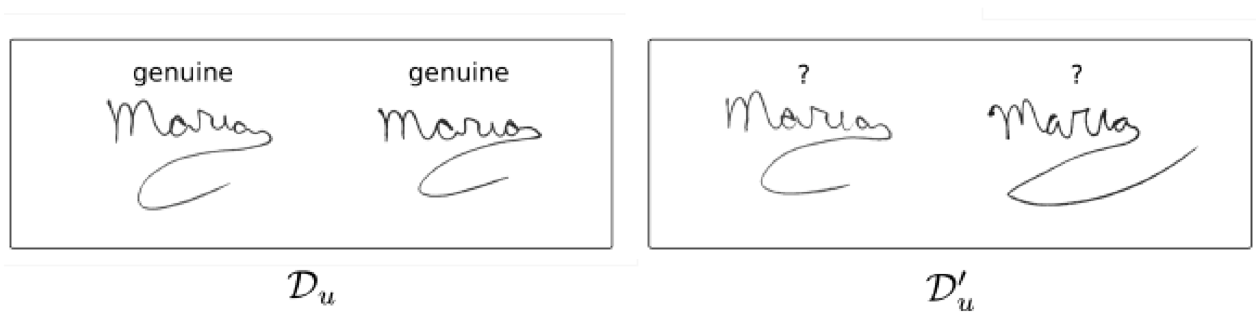

Fig 1.7: Example of Genuine and Forged Signatures

- **Grouping and Pre-handling**

For a specific underwriter, signatures are coupled as comparative (veritable, real) or different (certified, manufactured) sets. Pictures are in dim scale.

The pictures are resized to a fixed size of 155×220. Every pixel is standardized by isolating the pixel esteems with the standard deviation of the pixel estimations of the pictures in a dataset.

- **CNN and Siamese Network**

This is the fundamental part where pictures are passed as contribution to the convolutional Siamese neural systems. Siamese neural system is a class of system models that normally contains two indistinguishable subnetworks. The twin CNNs have a similar design with similar parameters and shared loads. The parameter refreshing is reflected across both the subnetworks.

These subnetworks are joined by a misfortune work at the top, which figures a closeness metric including the Euclidean separation between the element portrayal on each side of the Siamese system.

So as to at long last choose if two pictures have a place with the comparable class (certified, veritable) or a divergent class (real, manufactured), an edge work is resolved.

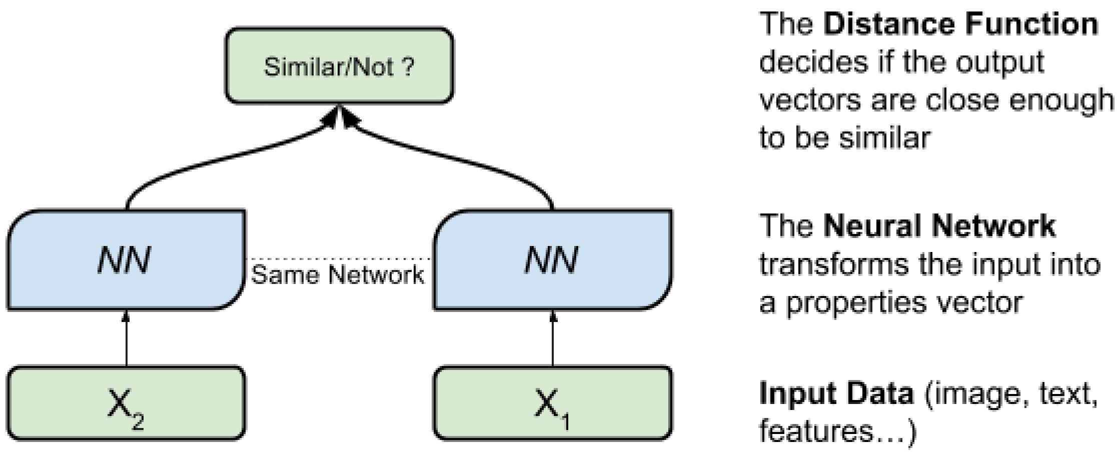

Fig 1.8: Siamese Network Explanation I

1.5 ORGANIZATION

The project contains dataset folder containing signatures of 17 persons. Here, for all the entries, 5 genuine and 10 forged signatures are given.

Also, there is a Python file named as code.py which contains the complete model code (Convolutional Siamese Neural Network) and its evaluation code.

During the execution the trained model gets stores in a H5 file names as sig.h5.

CHAPTER-2 LITERATURE SURVEY

2.1 TITLE: SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification *by Sounak Dey, Anjan Dutta, J. Ignacio Toledo, Suman K. Ghosh, Josep Llad´os, Umapada Pal*

- **CNN and Siamese Network**

Siamese systems are neural systems containing at least two indistinguishable subnetwork segments. It is significant that not just the design of the subnetworks is indistinguishable, however the loads must be shared among them also for the system to be designated "Siamese". The principle thought behind Siamese systems is that they can learn valuable information descriptors that can be additionally used to think about between the contributions of the separate subnetworks. Thus, sources of info can be anything from numerical information (for this situation the subnetworks are typically shaped by FC completely associated layers), picture information (with CNNs as subnetworks) or even successive information, for example, sentences or time signals (with RNNs as subnetworks).

To get nonlinearity amended straight units are likewise utilized. In this work, diverse convolutional pieces are utilized with sizes beginning with 11 × 11 to 3× 3. By and large, a differentiable misfortune work is picked with the goal that Gradient plunge can be applied and the system loads can be upgraded. Given a differentiable misfortune work, the loads of various layers are refreshed utilizing back spread. As the advancement can't be applied to all preparation information where preparing size is huge clump enhancements gives a reasonable choice to enhance the system.

Siamese neural system is a class of system structures that generally contains two indistinguishable subnetworks. The twin CNNs have a similar design with similar parameters and shared loads.

These subnetworks are joined by a misfortune work at the top, which processes a closeness metric including the Euclidean separation between the component portrayal on each side of the Siamese system**.** In this case, **Contrastive Loss Function** is utilised characterized as follows-

$$\mathbf{L\ (s_1, s_2, y) = \alpha\ (1 - y)\ D_w^2 + \beta y\ max\ (0, m - D_w)^2}$$

where s_1 and s_2 are two samples (here signature images), y is a binary indicator function denoting whether the two have a place with a similar class or not, α and β are two constants and m is the margin equal to 1 for our situation. $D_w = k\ //f(s_1,w_1) - f(s_2,w_2)//^2$ *is the Euclidean distance computed in the embedded feature space, f is an embedding function that maps a signature picture to real vector space through CNN, and w_1, w_2 are the learned weights for a specific layer of the underlying network.*

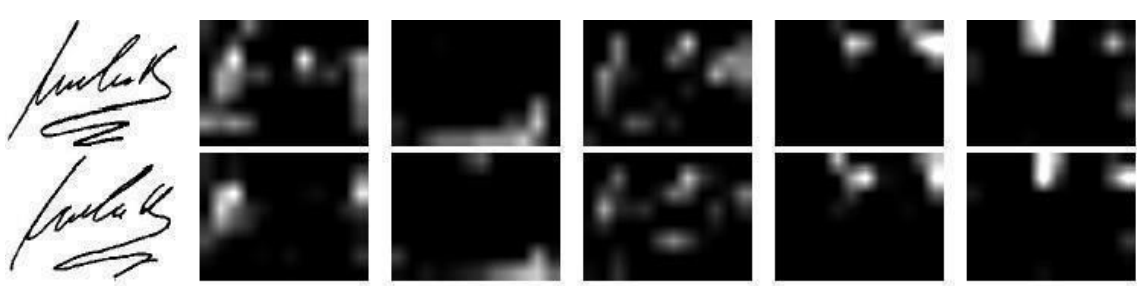

Fig 2.1: A pair of genuine (top left) and forged (bottom left) signatures, and corresponding response maps with five different filters that have produced higher energy activations in the last convolution layer of SigNet.

- ## Applications of Siamese Network

Siamese networks have wide-ranging applications. Here are some of them:

- **One-shot getting to know** - In this getting to know scenario, a new education dataset is offered to the trained (classification) community, with most effective one sample in keeping with class. Afterwards, the classification overall performance on this new dataset is examined on a separate testing dataset. As Siamese networks first research discriminative functions for a massive particular dataset, they may be used to generalize this understanding to completely new instructions and distributions as well. In (Koch, Gregory, Richard Zemel, and Rusian Salakhutdinov. "Siamese neural networks for oneshot photograph recognition." ICML Deep Learning Workshop. Vol. 2. 2015.), the writers use this capability to do one-shot learning at the MNIST dataset the use of a network educated at the Omniglot dataset (a completely different photograph dataset).
- **Pedestrian tracking for video surveillance** - In this work, a Siamese CNN community is mixed with length and role capabilities of photograph patches to track more than one persons within the field-of-view of the camera by detecting their position in every video frame, getting to know the institutions between a couple of frames and computing the trajectories.
- **Cosegmentation**

- **Matching resumes to jobs -** In this exceptional application, the network attempts to find matching task postings for applicants. In order to do this, a educated Siamese CNN network extracts deep contextual data from each the postings and the resumes and computes their semantic similarity. The hypothesis is that matching resume — posting pairs will rank better on the similarity scale than nonmatching ones.

- **Architecture**

A CNN engineering has been utilized as demonstrated as follows. For the simple reproducibility of the outcomes, a full rundown of parameters used to structure the CNN layers is introduced. For convolution and pooling layers, the size of the channels is recorded as N × H × W, where N is the quantity of channels, H is the tallness and W is the width of the comparing channel. Here, stride implies the separation between the utilization of channels for the convolution and pooling activities, and cushion shows the width of added fringes to the info. Here it is to be referenced that cushioning is important so as to convolve the channel from the absolute first pixel in the information picture. All through the system, Rectified Linear Units (ReLU) is utilized as enactment capacity to the yield of all the convolutional and completely associated layers.

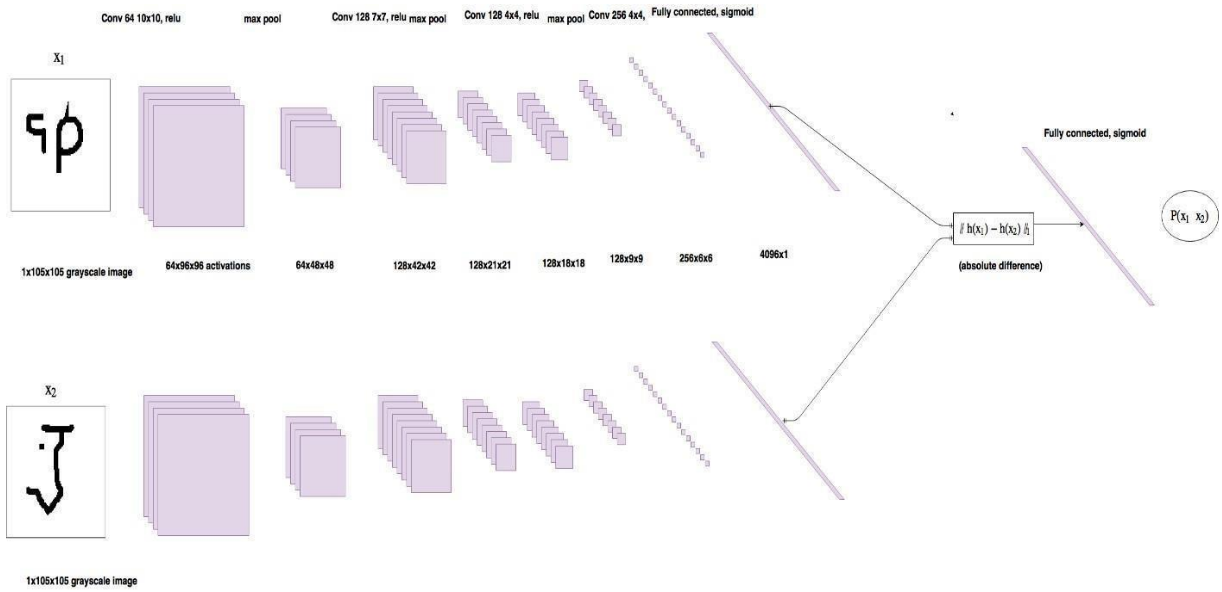

Fig 2.2: Siamese Network Explanation II

For summing up the scholarly highlights, Local Response Normalization is applied by [19], with the parameters appeared in the relating column in Table 1.

With the last two pooling layers and the principal FC completely associated layer, a Dropout is utilized with a rate equivalent to 0:3 and 0:5, individually.

The first convolutional layers channel the 155 × 220 info signature pictures with 96 bits of size 11 × 11 with a step of 1 pixel. The second convolutional layer takes as info the (reaction standardized and pooled) yield of the first convolutional layer and channels it with 256 pieces of size 5×5. The third and fourth convolutional layers are associated with each other with no mediation of pooling or standardization of layers.

The third layer has 384 pieces of size 3 × 3 associated with the (standardized, pooled, and dropout) yield of the second convolutional layer. The fourth convolutional layer has 256 parts of size 3×3. This prompts the neural system learning less lower level highlights for littler open fields and more highlights for more elevated level or increasingly dynamic highlights. The main FC completely associated layer has 1024 neurons, though the second FC completely associated layer has 128 neurons. This demonstrates the most elevated took in include vector from each side of SigNet has a measurement equivalent to 128.

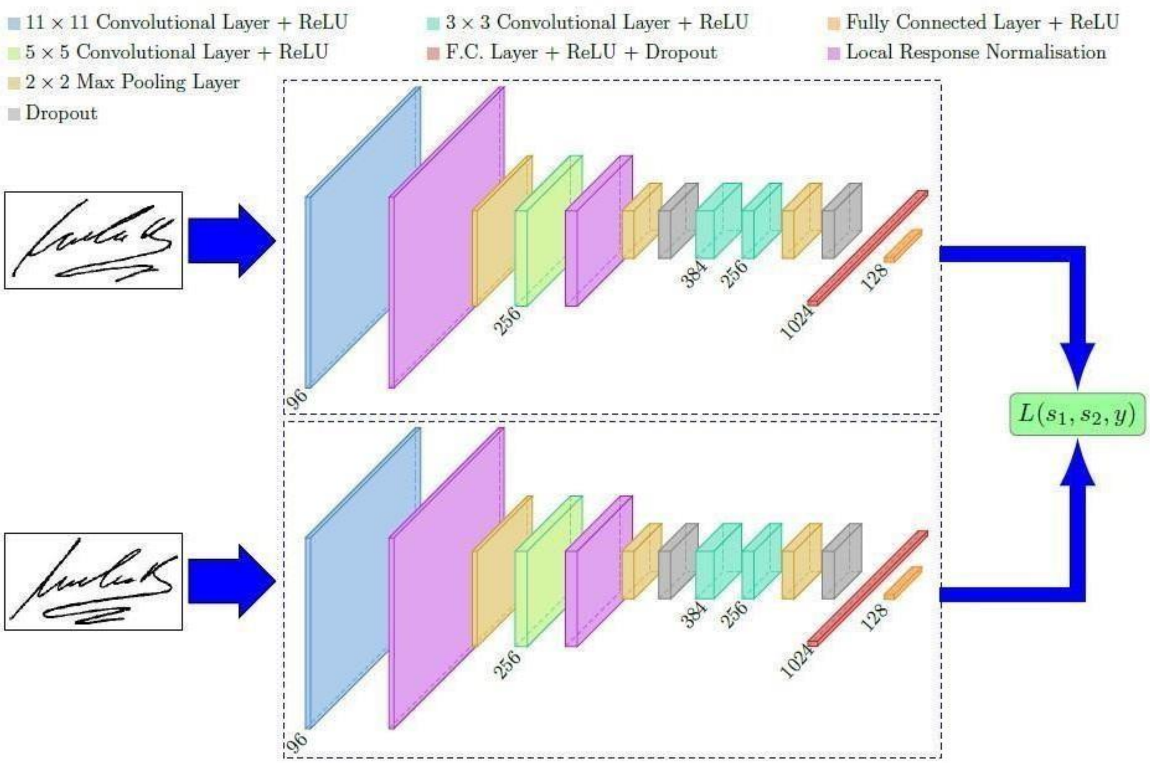

Fig 2.3: Siamese Network Explanation III

Dataset used in this Research Paper

The paper used BHSig260 signature dataset which contains the signatures of 260 persons, among them 100 were signed in Bengali and 160 are signed in Hindi. For each of the signers, 24 genuine and 30 forged signatures are available. This results in $100 \times 24 = 2; 400$ genuine and $100\times 30 = 3, 000$ forged signatures in Bengali, and $160 \times 24 = 3; 840$ genuine and $160 \times 30 = 4, 800$ forged signatures in Hindi. Even though this dataset is available together, they experimented their method separately on the Bengali and Hindi dataset.

2.2 TITLE: Deep learning Specialization by deeplearning.ai

- What is Convolutional Neural Network?

  PC vision is advancing quickly step by step. It's one reason is deep learning. If there should arise an occurrence of PC vision, convolutional neural system (condensed as CNN) consistently become possibly the most important factor in light of the fact that CNN is intensely utilized here. Instances of CNN in PC vision are face acknowledgment, picture arrangement and so forth. It is like the essential neural system. CNN likewise have learnable parameter like neural system i.e., loads, inclinations and so forth.

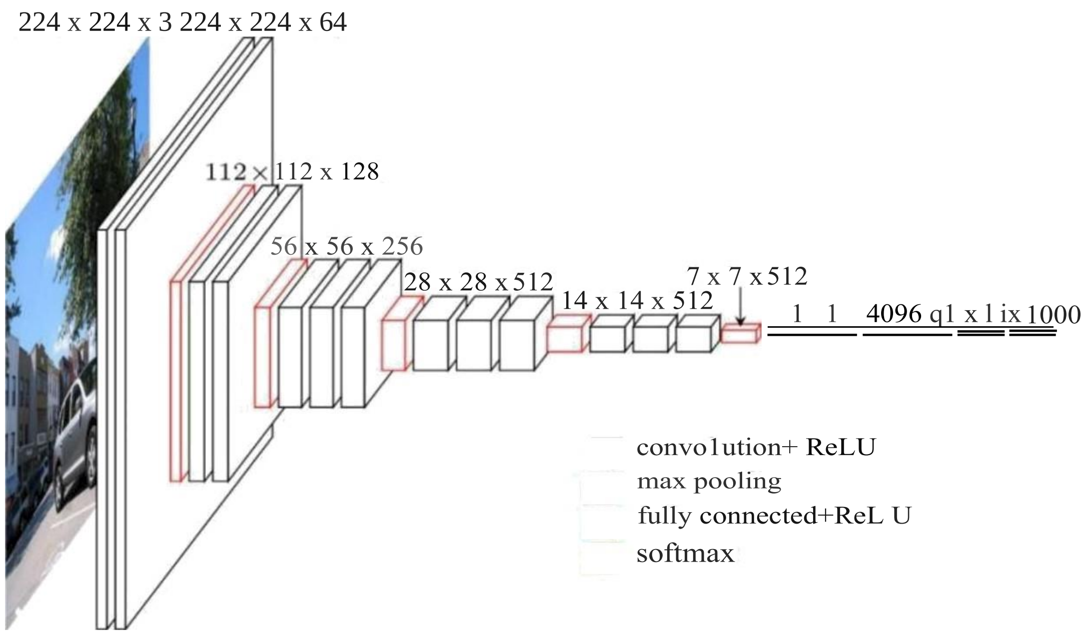

Fig 2.4: Convolutional Neural Network Explanation

Layers in CNN

o Input layer

o Convo layer (Convo + ReLU)

o Pooling layer

o Fully connected (FC) layer

o SoftMax/logistic layer

o Output layer

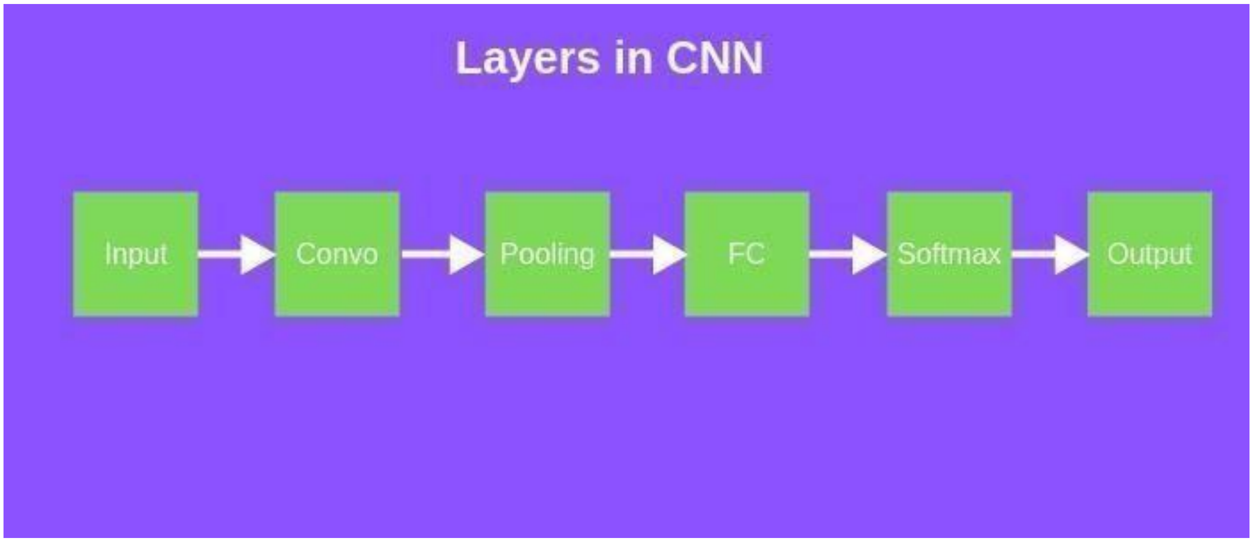

Fig 2.5: Layers in CNN

- **Input Layer:**
  Information layer in CNN ought to contain picture information. Picture information is spoken to by three-dimensional grid. You have to reshape it into a solitary section. Assume you have picture of measurement 28 x 28 =784, you have to change over it into 784 x 1 preceding taking care of into input. On the off chance that you have "m" preparing models at that point measurement of info will be (784, m).

- **Convo Layer:**
  Convo layer is at times called highlight extractor layer since highlights of the picture are get separated inside this layer. As a matter of first importance, a piece of picture is associated with Convo layer to perform convolution activity and ascertaining the spot item between responsive field (it is a nearby locale of the info picture that has a similar size as that of channel) and the channel. Consequence of the activity is single number of the yield volume. At that point the channel is slided throughout the following open field of a similar info picture by a Stride and do a similar activity

once more. The entire procedure is rehashed and again until it traverses the entire picture. The yield will be the contribution for the following layer.

- **Pooling Layer**

  Pooling layer is utilized to lessen the spatial volume of information picture after convolution. It is utilized between two convolution layers. In the event that FC is applied after Convo layer without applying pooling or max pooling, at that point it will be computationally costly. Thus, the maximum pooling is best way to lessen the spatial volume of info picture.

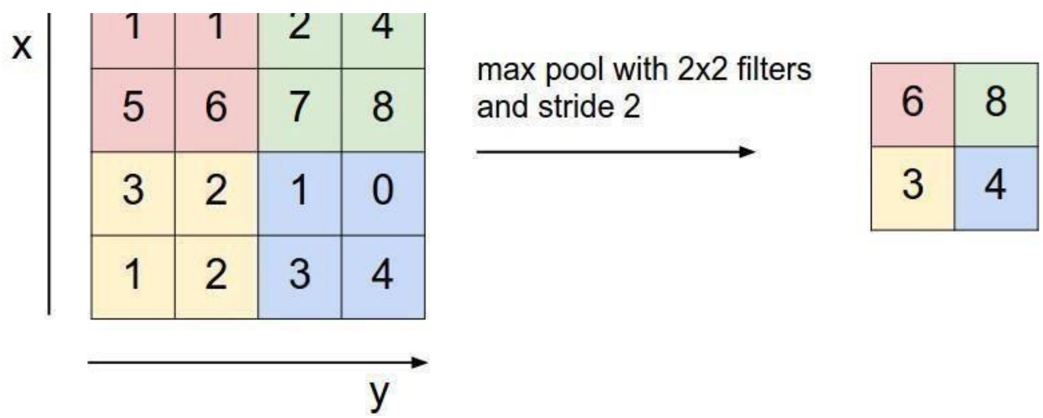

Fig 2.6: Pooling Layer in CNN

- **Fully Connected Layer**

  Fully Connected layer includes loads, inclinations, and neurons. It interfaces neurons in a single layer to neurons in another layer. It is utilized to arrange pictures between various class via preparing.

- **SoftMax / Logistic Layer**

  SoftMax or Logistic layer is the last layer of CNN. It dwells toward the finish of FC layer. Logistic is utilized for binary arrangement and SoftMax is for multiclassification.

- **Output Layer**

  Output layer contains the signature which is as one-hot encoded.

Why Convolutions?

There are fundamentally two favourable circumstances of convolutions.
Parameter Sharing
Sparsity of Connections

1. Parameter Sharing
A feature detector (for example, a vertical edge finder) that is valuable in one piece of the picture is most likely helpful in another piece of the picture.
2. Sparsity of Connections
In each layer, each output esteem relies upon few data sources.

VGG16 Network

Karen Simonyan and Andrew Zisserman researched the impact of the convolutional Network depth on its precision in the huge scope picture acknowledgment setting. They expanded the depth of their engineering to 16 and 19 layers with exceptionally little (3×3) convolution channels. They named their finding as VGG16 (Visual Geometry Group) and VGG19. The group won the first and the second place in the confinement and arrangement tracks individually at the ImageNet Challenge 2014 accommodation. The VGG16 design comprises of twelve convolutional layers, some of which are trailed by most extreme pooling layers and afterward four FC completely associated layers and lastly a 1000-way SoftMax classifier.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 2.7: ConvNet Configuration

1. First and Second Layers:

The contribution for AlexNet is a 224x224x3 RGB picture which goes through first and second convolutional layers with 64 element maps or channels having size 3×3 and same pooling with a step of 14. The picture measurements changes to 224x224x64.At that point the VGG16 applies greatest pooling layer or subtesting layer with a channel size 3×3 and a step of two. The subsequent picture measurements will be decreased to 112x112x64.

2. Third and Fourth Layer:

Next, there are two convolutional layers with 128 element maps having size 3×3 and a step of 1.

At that point there is again a most extreme pooling layer with channel size 3×3 and a step of 2. This layer is same as past pooling layer aside from it has 128 component maps so the yield will be decreased to 56x56x128.

3. Fifth and Sixth Layers:

The fifth and 6th layers are convolutional layers with channel size 3×3 and a step of one. Both utilized 256 component maps.

The two convolutional layers are trailed by a most extreme pooling layer with channel size 3×3, a step of 2 and have 256 component maps.

4. Seventh to Twelfth Layer:

Next are the two arrangements of 3 convolutional layers followed by a most extreme pooling layer. All convolutional layers have 512 channels of size 3×3 and a step of one. The last size will be diminished to 7x7x512..

5. Thirteenth Layer:

The convolutional layer yield is straightened through a FC completely associated layer with 25088 element maps every one of size 1×1.

6. Fourteenth and Fifteenth Layers:

Again we have two fully connected layers with 4096 units.

7. Output Layer:

At long last, there is a SoftMax output layer ŷ with 1000 potential qualities.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Fig 2.8: VGG16 Explanation

- **ResNet**

One of the issues ResNets explain is the celebrated known disappearing angle. This is on the grounds that when the system is excessively deep, the angles from where the misfortune work is determined effectively therapist to zero after a few uses of the chain rule. This outcome on the loads never refreshing its qualities and along these lines, no learning is being performed.

With ResNets, the angles can stream legitimately through the skip associations in reverse from later layers to starting channels.

The second issue with preparing the more deep systems is, playing out the streamlining on colossal parameter space and in this way gullibly adding the layers prompting higher preparing mistake. Lingering systems permit preparing of such deep systems by building the system through modules called remaining models as appeared in the figure. This is called **debasement issue.**

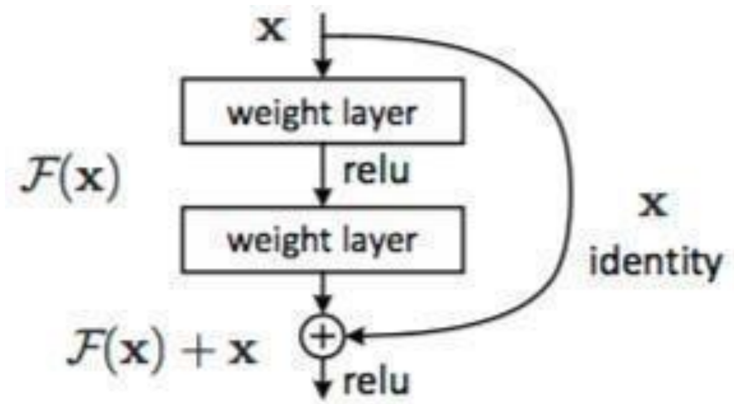

Fig 2.9: ResNet Explanation

CHAPTER-3 SYSTEM DEVELOPMENT

3.1 DATASET

To implement this model of signature verification, a dataset of signatures of different persons has been made. The dataset contains the signatures of 17 different persons. Here, for all the entries, 5 genuine and 10 forged signatures are given. This results in $17 \times 5 = 80$ genuine and $17 \times 10 = 170$ forged signatures.

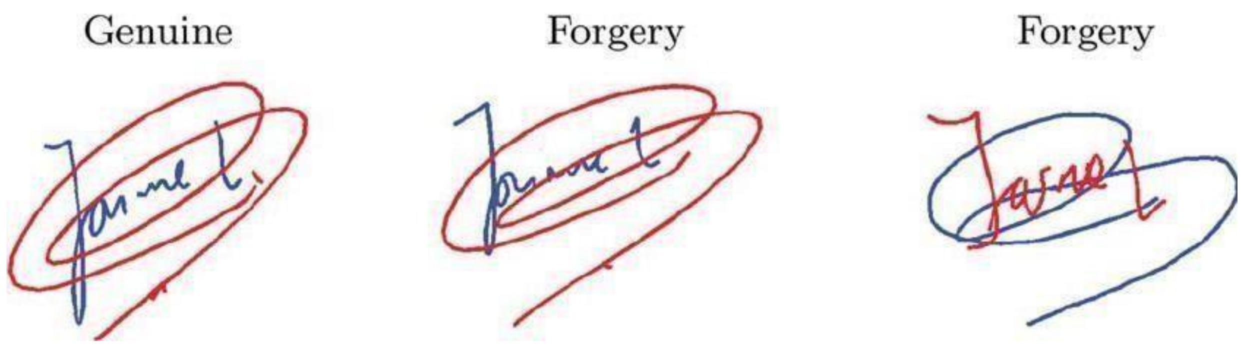

Fig 3.1: Genuine vs Forgery

3.2 MODEL DEVELOPMENT

Hardware Used

2.2 GHz Intel Core i5 5th generation CPU

8GB DDR3 RAM ○ Nvidia GeForce 920M GPU

Software Used ○ Windows 10 OS ○ Anaconda Spyder ○ Python 3 (version 3.6)

TensorFlow v1.1.2

Keras v2.2.4

NumPy and Other basic Libraries

BUILDING THE MODEL

The model has been coded in Python 3.6 with the help of necessary libraries in the Anaconda IDE.

The first part of the code involves the importing of libraries that are essential for the successful execution of the program as shown below where Keras and TensorFlow are for deep learning algorithms to be applied on images and NumPy is for normal matrix based and other calculations.

```
import tensorflow as tf
from keras.models import Sequential
from keras.optimizers import Adam, RMSprop
from keras.layers import Conv2D, ZeroPadding2D, Activation, Input, concatenate, Dropout
from keras.models import Model
from keras.layers.normalization import BatchNormalization
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import Concatenate
from keras.layers.core import Lambda, Flatten, Dense
from keras.initializers import glorot_uniform
from keras.engine.topology import Layer
from keras.regularizers import l2
from keras import backend as K
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
import sys
import numpy as np
import pickle
import os
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
import time
import itertools
import random
from sklearn.utils import shuffle
```

Fig 3.2 : Code Snippet #1

Further, the code sets the path of the dataset to be used for the training of the model. Also, the dataset imported is further divided into the training, test and validation sets as shown below.

```python

path = "./dataset1/"



folder_list = next(os.walk(path))[1]
folder_list.sort()


orig_groups, forg_groups = [], []
for directory in folder_list:
    images = os.listdir(path+directory)
    images.sort()
    images = [path+directory+'/'+x for x in images]
    forg_groups.append(images[:10])
    orig_groups.append(images[10:])



orig_train, orig_val, orig_test = orig_groups[:11], orig_groups[11:13], orig_groups[13:]
forg_train, forg_val, forg_test = forg_groups[:11], forg_groups[11:13], forg_groups[13:]
print((orig_train))

img_h, img_w =155,220

```

Fig 3.3 : Code Snippet #2

The visulaize_sample_signature function is defined to visualize and compare the genuine images with their forged ones or vice-versa.

```python
def visualize_sample_signature():
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize = (10, 10))
    k = np.random.randint(len(orig_train))
    orig_img_names = random.sample(orig_train[k], 2)
    forg_img_name = random.sample(forg_train[k], 1)
    orig_img1 = cv2.imread(orig_img_names[0], 0)
    orig_img2 = cv2.imread(orig_img_names[1], 0)
    forg_img = plt.imread(forg_img_name[0], 0)
    orig_img1 = cv2.resize(orig_img1, (img_w, img_h))
    orig_img2 = cv2.resize(orig_img2, (img_w, img_h))
    forg_img = cv2.resize(forg_img, (img_w, img_h))

    ax1.imshow(orig_img1, cmap = 'gray')
    ax2.imshow(orig_img2, cmap = 'gray')
    ax3.imshow(forg_img, cmap = 'gray')

    ax1.set_title('Genuine Copy')
    ax1.axis('off')
    ax2.set_title('Genuine Copy')
    ax2.axis('off')
    ax3.set_title('Forged Copy')
    ax3.axis('off')
```

Fig 3.4 : Code Snippet
#3

Further, model defines a function to generate a batch of data with “batch_size” number of data points.

Some of the data points will be Genuine-Genuine pairs and some will be GenuineForged pairs.

```
def generate_batch(orig_groups, forg_groups, batch_size = 32):

    while True:
        orig_pairs = []
        forg_pairs = []
        gen_gen_labels = []
        gen_for_labels = []
        all_pairs = []
        all_labels = []


        for orig, forg in zip(orig_groups, forg_groups):
            orig_pairs.extend(list(itertools.combinations(orig, 2)))
            for i in range(len(forg)):
                forg_pairs.extend(list(itertools.product(orig[i:i+1], random.sample(forg, 10))))
        gen_gen_labels = [1]*len(orig_pairs)
        gen_for_labels = [0]*len(forg_pairs)


        all_pairs = orig_pairs + forg_pairs
        all_labels = gen_gen_labels + gen_for_labels
        del orig_pairs, forg_pairs, gen_gen_labels, gen_for_labels
        all_pairs, all_labels = shuffle(all_pairs, all_labels)
```

Fig 3.5 : Code Snippet #4

The code creates pairs of Genuine-Genuine image names and Genuine-Forged image names. For every person there are 5 genuine signatures, hence in total, 5 choose 2 = 10 Genuine-Genuine image pairs are there for one person. To make Genuine-Forged pairs, every Genuine signature of a person is paired with 10 randomly sampled Forged signatures of the same person.

Thus, there will be 5 * 10 = 50 Genuine-Forged image pairs for one person.

In all, Training data contains data of 11 different persons. Hence,

Total no. of Genuine-Genuine pairs = 11 * 10 = 110

Total number of Genuine-Forged pairs = 11 * 50 = 550.

Note the lists above contain only the image names and actual images are loaded and yielded below in batches. Below code prepares a batch of data points and yield the batch.

In each batch, "batch_size" number of image pairs are loaded. These images are then removed from the original set so that they are not added again in the next batch.

```
k = 0
pairs=[np.zeros((batch_size, img_h, img_w, 1)) for i in range(2)]
targets=np.zeros((batch_size,))
for ix, pair in enumerate(all_pairs):
    img1 = cv2.imread(pair[0], 0)
    img2 = cv2.imread(pair[1], 0)
    img1 = cv2.resize(img1, (img_w, img_h))
    img2 = cv2.resize(img2, (img_w, img_h))
    img1 = np.array(img1, dtype = np.float64)
    img2 = np.array(img2, dtype = np.float64)
    img1 /= 255
    img2 /= 255
    img1 = img1[..., np.newaxis]
    img2 = img2[..., np.newaxis]
    pairs[0][k, :, :, :] = img1
    pairs[1][k, :, :, :] = img2
    targets[k] = all_labels[ix]
    k += 1
    if k == batch_size:
        yield pairs, targets
        k = 0
        pairs=[np.zeros((batch_size, img_h, img_w, 1)) for i in range(2)]
        targets=np.zeros((batch_size,))
```

Fig 3.6: Code Snippet #5

Since the project is totally based on the use of Convolutional Siamese Neural Network (SigNet), the below snapshot depicts the functions created for the implementation of base network SigNet and the functions for the contrastive loss and Euclidean distance between the two vectors in the latent space.

```python
def euclidean_distance(vects):
    x, y = vects
    return K.sqrt(K.sum(K.square(x - y), axis=1, keepdims=True))

def eucl_dist_output_shape(shapes):
    shape1, shape2 = shapes
    return (shape1[0], 1)

def contrastive_loss(y_true, y_pred):
    margin = 1
    return K.mean(y_true * K.square(y_pred) + (1 - y_true) * K.square(K.maximum(margin - y_pred, 0)))


def create_base_network_signet(input_shape):
    '''Base Siamese Network'''

    seq = Sequential()
    seq.add(Conv2D(96, kernel_size=(11, 11), activation='relu', name='conv1_1', strides=4, input_shape= input_shape,
                        init='glorot_uniform', dim_ordering='tf'))
    seq.add(BatchNormalization(epsilon=1e-06, mode=0, axis=3, momentum=0.9))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(ZeroPadding2D((2, 2), dim_ordering='tf'))
```

Fig 3.7 : Code Snippet #6

The definition of the function "create_base_network_signet" contains the full info of the layers of the neural network including the values of the required essential parameters.

```python
    seq.add(Conv2D(256, kernel_size=(5, 5), activation='relu', name='conv2_1', strides=1, init='glorot_uniform',
dim_ordering='tf'))
    seq.add(BatchNormalization(epsilon=1e-06, mode=0, axis=3, momentum=0.9))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(Dropout(0.3))
    seq.add(ZeroPadding2D((1, 1), dim_ordering='tf'))

    seq.add(Conv2D(384, kernel_size=(3, 3), activation='relu', name='conv3_1', strides=1, init='glorot_uniform',
dim_ordering='tf'))
    seq.add(ZeroPadding2D((1, 1), dim_ordering='tf'))

    seq.add(Conv2D(256, kernel_size=(3, 3), activation='relu', name='conv3_2', strides=1, init='glorot_uniform',
dim_ordering='tf'))
    seq.add(MaxPooling2D((3,3), strides=(2, 2)))
    seq.add(Dropout(0.3))
    seq.add(Flatten(name='flatten'))
    seq.add(Dense(1024, W_regularizer=l2(0.0005), activation='relu', init='glorot_uniform'))
    seq.add(Dropout(0.5))

    seq.add(Dense(128, W_regularizer=l2(0.0005), activation='relu', init='glorot_uniform'))
    return seq
```

Fig 3.8: Code Snippet #7

```
input_shape=(img_h, img_w, 1)
base_network = create_base_network_signet(input_shape)
input_a = Input(shape=(input_shape))
input_b = Input(shape=(input_shape))

processed_a = base_network(input_a)
processed_b = base_network(input_b)

# Compute the Euclidean distance between the two vectors in the latent space
distance = Lambda(euclidean_distance, output_shape=eucl_dist_output_shape)([processed_a, processed_b])

model = Model(input=[input_a, input_b], output=distance)
batch_sz = 5
num_train_samples = 10*11 + 50*11
num_val_samples=10*2 + 50*2
num_test_samples=10*4+50*4

rms = RMSprop(lr=1e-4, rho=0.9, epsilon=1e-08)
model.compile(loss=contrastive_loss, optimizer=rms)

callbacks = [
    EarlyStopping(patience=12, verbose=1),
    ReduceLROnPlateau(factor=0.1, patience=5, min_lr=0.000001, verbose=1),
    ModelCheckpoint('./sig.h5', verbose=1, save_weights_only=True)]
```

Fig 3.9 : Code Snippet #8

The below snapshot depicts the code which performs the fitting of the model on the dataset imported and sets the number of epochs for which the neural network will run. The model after successful execution on the dataset is saved into a H5 file named as sig.h5 for future use. These files contain all the weights that neural network founds to be appropriate during its training.

A function for calculating the accuracy of the model is also defined here.

```
results = model.fit_generator(generate_batch(orig_train, forg_train, batch_sz),
                              steps_per_epoch = num_train_samples//batch_sz,
                              epochs =15,
                              validation_data = generate_batch(orig_val, forg_val, batch_sz),
                              validation_steps = num_val_samples//batch_sz,
                              callbacks = callbacks)
model.save('./sig.h5')




def compute_accuracy_roc(predictions, labels):
    '''Compute ROC accuracy with a range of thresholds on distances.
    '''
    dmax = np.max(predictions)
    dmin = np.min(predictions)
    nsame = np.sum(labels == 1)
    ndiff = np.sum(labels == 0)

    step = 0.01
    max_acc = 0
    best_thresh = -1
```

Fig 3.10 : Code Snippet #9

The file sig.h5 is loaded in order to load the weights of the neural network so that they can be used while testing the model.

```
    for d in np.arange(dmin, dmax+step, step):
        idx1 = predictions.ravel() <= d
        idx2 = predictions.ravel() > d

        tpr = float(np.sum(labels[idx1] == 1)) / nsame
        tnr = float(np.sum(labels[idx2] == 0)) / ndiff
        acc = 0.5 * (tpr + tnr)
#       print ('ROC', acc, tpr, tnr)

        if (acc > max_acc):

            max_acc, best_thresh = acc, d

    return max_acc, best_thresh

model.load_weights('/Users/jyotiraditya_varma/Desktop/Jyotir_Signet/sig.h5')

test_gen = generate_batch(orig_test, forg_test, 1)
pred, tr_y = [], []
for i in range(num_test_samples):
    (img1, img2), label = next(test_gen)
    tr_y.append(label)
    pred.append(model.predict([img1, img2])[0][0])
```

Fig 3.11: Code Snippet #10

Finally, the predict_score function predicts distance score and classifies test images as Genuine or Forged.

```python
tr_acc, threshold = compute_accuracy_roc(np.array(pred), np.array(tr_y))
tr_acc, threshold


def predict_score():
    '''Predict distance score and classify test images as Genuine or Forged'''
    test_point, test_label = next(test_gen)
    img1, img2 = test_point[0], test_point[1]

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
    if test_label == 1:
        ax2.set_title('Genuine')
    else:
        ax2.set_title('Forged')
    ax1.axis('off')
    ax2.axis('off')
    plt.show()
    result = model.predict([img1, img2])
    diff = result[0][0]
    print("Difference Score = ", diff)
```

Fig 3.12 : Code Snippet
#11

```python
    if diff > threshold:
        print("Its a Forged Signature")
    else:
        print("Its a Genuine Signature")


predict_score()
```

Fig 3.13: Code Snippet #12

CHAPTER-4 PERFORMANCE ANALYSIS

4.1 PERFORMANCE ANALYSIS OF THE MODEL

A threshold d is used on the distance measure D (x_i, x_j) output by the SigNet to decide whether the signature pair (i, j) belongs to the similar or dissimilar class. The signature pairs (i, j) are denoted with the same identity as $P_{similar}$, whereas all pairs of different identities as $P_{dissimilar}$. Then, the set of all true positives (TP) at d can be defined as,

$$TP(d) = \{(i, j) \in \mathcal{P}_{\text{similar}},\ \text{with } D(x_i, x_j) \leq d\}$$

Similarly, the set of all true negatives (TN) at d can be defined as,

$$TN(d) = \{(i, j) \in \mathcal{P}_{\text{dissimilar}},\ \text{with } D(x_i, x_j) > d\}$$

Then the true positive rate TPR(d) and the true negative rate TNR(d) for a given signature, distance d is then defined as,

$$TPR(d) = \frac{|TP(d)|}{|\mathcal{P}_{\text{similar}}|},\ TNR(d) = \frac{|TN(d)|}{|\mathcal{P}_{\text{dissimilar}}|}$$

where Psimilar is the number of similar signature pairs. The final accuracy is computed as,

$$\text{Accuracy} = \max_{d \in D} \frac{1}{2}(TPR(d) + TNR(d))$$

which is the maximum accuracy obtained by varying d (belong to D) from the minimum distance value to the maximum distance value of D with step equal to 0.01.

4.2 PREDICTION OF SCORES

As discussed earlier, the code includes a function compute_accuracy_roc for calculating the accuracy of the model as shown below.

```
results = model.fit_generator(generate_batch(orig_train, forg_train, batch_sz),
                              steps_per_epoch = num_train_samples//batch_sz,
                              epochs =15,
                              validation_data = generate_batch(orig_val, forg_val, batch_sz),
                              validation_steps = num_val_samples//batch_sz,
                              callbacks = callbacks)
model.save('./sig.h5')




def compute_accuracy_roc(predictions, labels):
    '''Compute ROC accuracy with a range of thresholds on distances.
    '''
    dmax = np.max(predictions)
    dmin = np.min(predictions)
    nsame = np.sum(labels == 1)
    ndiff = np.sum(labels == 0)

    step = 0.01
    max_acc = 0
    best_thresh = -1
```

Fig 4.1 : Code Snippet
#13

```
    for d in np.arange(dmin, dmax+step, step):
        idx1 = predictions.ravel() <= d
        idx2 = predictions.ravel() > d

        tpr = float(np.sum(labels[idx1] == 1)) / nsame
        tnr = float(np.sum(labels[idx2] == 0)) / ndiff
        acc = 0.5 * (tpr + tnr)
#       print ('ROC', acc, tpr, tnr)

        if (acc > max_acc):

            max_acc, best_thresh = acc, d

    return max_acc, best_thresh

model.load_weights('/Users/jyotiraditya_varma/Desktop/Jyotir_Signet/sig.h5')

test_gen = generate_batch(orig_test, forg_test, 1)
pred, tr_y = [], []
for i in range(num_test_samples):
    (img1, img2), label = next(test_gen)
    tr_y.append(label)
    pred.append(model.predict([img1, img2])[0][0])
```

Fig 4.2 : Code Snippet
#14

Finally, the predict_score function predicts distance score and classifies test images as Genuine or Forged.

```python
tr_acc, threshold = compute_accuracy_roc(np.array(pred), np.array(tr_y))
tr_acc, threshold



def predict_score():
    '''Predict distance score and classify test images as Genuine or Forged'''
    test_point, test_label = next(test_gen)
    img1, img2 = test_point[0], test_point[1]

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (10, 10))
    ax1.imshow(np.squeeze(img1), cmap='gray')
    ax2.imshow(np.squeeze(img2), cmap='gray')
    ax1.set_title('Genuine')
    if test_label == 1:
        ax2.set_title('Genuine')
    else:
        ax2.set_title('Forged')
    ax1.axis('off')
    ax2.axis('off')
    plt.show()
    result = model.predict([img1, img2])
    diff = result[0][0]
    print("Difference Score = ", diff)
```

Fig 4.3 : Code Snippet #15

If this difference score is found to be greater than a set threshold, the image is classified as a forged signature otherwise the image is classified as a genuine one.

```python
    if diff > threshold:
        print("Its a Forged Signature")
    else:
        print("Its a Genuine Signature")


predict_score()
```

Fig 4.4 : Code Snippet #16

CHAPTER-5 CONCLUSIONS 5.1 CONCLUSION

In this project, I have introduced a structure dependent on Siamese system for offline signature confirmation. This technique gains from the information in a writer independent scenario, instead of hand crafted attributes.

This SigNet model has given extraordinary results on variety of signature containing various penmanship styles and dialects, which is extremely promising for additional exploration toward this path and better utilization of convolutional Siamese neural systems in the real world applications.

5.2 FUTURE SCOPE

Signature is one of the most recognised and normally acknowledged biometric trade signatures that has been utilized since ages confirming various elements identified with people, for example, archives, structures, bank checks, people, and so on. In this way, signature check is a basic undertaking.

References

- Research Paper: *SigNet: Convolutional Siamese Network for Writer Independent Offline Signature Verification by Sounak Dey, Anjan Dutta, J. Ignacio Toledo, Suman K.Ghosh, Josep Llad´os, Umapada Pal*
- Deep learning Specialization by *deeplearning.ai*
- https://github.com/hlamba28/Offline-Signature-Verification-using-Siamese-Network
- https://github.com/keras-team/keras/blob/master/examples/mnist_siamese.py
- R. Plamondon, S. Srihari, Online and o_-line handwriting recognition: a comprehensive survey, IEEE TPAMI 22 (1) (2000) 63–84.
- D. Impedovo, G. Pirlo, Automatic signature verification: The state of the art, IEEE TSMC 38 (5) (2008) 609–635.
- M. E. Munich, P. Perona, Visual identification by signature tracking, IEEE TPAMI 25 (2) (2003) 200–217.
- D. Bertolini, L. Oliveira, E. Justino, R. Sabourin, Reducing forgeries in writer-independent o_-line signature verification through ensemble of classifiers, PR 43 (1) (2010) 387–396.
- M. K. Kalera, S. N. Srihari, A. Xu, O_ine signature verification and identification using distance statistics, IJPRAI 18 (7) (2004) 1339–1360.
- G. Dimauro, S. Impedovo, G. Pirlo, A. Salzo, A multi-expert signature verification system for bankcheck processing, IJPRAI 11 (05) (1997) 827–844.
- M. A. Ferrer, J. B. Alonso, C. M. Travieso, O_ine geometric parameters for

automatic signature verification using fixed-point arithmetic, IEEE TPAMI 27 (6) (2005) 993–997.

- R. Kumar, J. Sharma, B. Chanda, Writer-independent o_-line signature verification using surroundedness feature, PRL 33 (3) (2012) 301–308.
- K. Huang, H. Yan, O_-line signature verification based on geometric feature extraction and neural network classification, PR 30 (1) (1997) 9–17.
- V. Ramesh, M. N. Murty, O_-line signature verification using genetically optimized weighted features, PR 32 (2) (1999) 217–233.