

B. TECH. PROJECT REPORT

On

FINGERPRINT MATCHING USING A DEEP LEARNING BASED APPROACH

By

Smit Patel



**DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE**

May 2022

Fingerprint Matching using a Deep Learning Based Approach

A PROJECT REPORT

*Submitted in partial fulfillment of the requirements for the award of
the degrees*

of
BACHELOR OF TECHNOLOGY
in
ELECTRICAL ENGINEERING

Submitted by:
Smit Patel

Guided by:
Dr. Surya Prakash
(Associate Professor, Computer Science and Engineering)



DISCIPLINE OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
May 2022

CANDIDATE'S DECLARATION

I hereby declare that the project entitled “**FINGERPRINT MATCHING USING A DEEP LEARNING BASED APPROACH**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Electrical Engineering’ completed under the supervision of **Dr. Surya Prakash, Associate Professor, Computer Science and Engineering, IIT Indore** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.




27/05/22

Signature and name of the student(s) with date
Smit Patel

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the student is correct to the best of my knowledge.



Signature of BTP Guide(s) with dates and their designation
Dr. Surya Prakash
Associate Professor, CSE

Preface

This report on "Fingerprint Matching using a Deep Learning Based Approach" is prepared under the guidance of Dr. Surya Prakash.

In this research, I have employed a very popular Deep Learning architecture namely Vision Transformer in a Siamese Network setting to learn the similarity between a given pair of fingerprint images. Multiple experiments with variations in the model have been performed and the data has been recorded. A custom algorithm has been implemented which is used to determine the threshold for predicting the label of the outcome. I have endeavoured to ensure every part and every detail of the pipeline has been covered succinctly.

I have endeavoured to the best of my abilities and knowledge to elucidate the content in a lucid manner. Moreover, I have included figures and charts to make the report more illustrative.

Smit Patel

B.Tech. IV Year

Discipline of Electrical Engineering

IIT, Indore

Acknowledgements

Thank you first and foremost to my thesis supervisor Dr. Surya Prakash for giving me the opportunity to do research in this unique field. Without his kind support and valuable guidance, this report would not have been possible. Thank you for always showing listening to my ideas patiently and providing valuable feedback on them.

Thank you to Mr. Vivek Singh Baghel (PhD scholar in CSE) for his constant guidance throughout the process, for introducing me to the concepts and methodologies in this field and for helping me during my work's critical stages.

I also take this opportunity to thank the researchers at IIT Kanpur for creating a good quality dataset and giving access to us for my thesis.

I am also thankful to my family and friends for their constant motivation and for helping to proofread and critique my thesis. I definitely would not have been able to do this without you all.

Smit Patel

B.Tech. IV Year

Discipline of Electrical Engineering

IIT, Indore

FINGERPRINT MATCHING USING A DEEP LEARNING BASED APPROACH

Abstract

As we move towards a technological driven era, the traditional methods of data or personnel verification are becoming redundant and easier to crack. Biometric authentication has emerged as a very promising technique as it uses features of human body which are unique to every individual. In an effort to adopt recent developments in Machine Learning (ML) and Natural Language Processing (NLP) and apply them to the domain of biometric verification, I propose a novel Vision Transformer (ViT) based Siamese Network (SN) framework for fingerprint matching. Our primary focus is holistic and a end-to-end pipeline has been constructed and implemented using an ensemble of task-specific algorithms to procure the best possible result from the model. I have also endeavoured to identify specific problems on the application of ViT to our problem statement and introduced two major modifications, Shifted Patch Tokenization (SPT) and Localized Self Attention (LSA) to tackle those shortcomings effectively. I propose two variations for the model, namely Intermediate-Merge (IM) Siamese Network and Late Merge (LM) Siamese Network and test the performances on a fingerprint dataset from IIT Kanpur.

Keywords: Fingerprint Matching, Vision Transformer, Siamese Networks, Deep Learning

Contents

1	Introduction	6
1.1	Motivation of the Work	7
1.2	Novelty of the work	8
1.3	Fingerprint Nomenclature	9
1.4	Thesis Organisation	11
2	Literature Review	12
3	The Algorithms of Focus	14
3.1	Transformers	14
3.2	Vision Transformers	17
3.3	Siamese Network	20
3.4	Threshold Calculation	21
3.5	Shifted Patch Tokenization	22
3.5.1	Problem	23
3.5.2	Solution	23
3.6	Locality Self-Attention	25
3.6.1	Problem	25
3.6.2	Solution	26
4	Experimental Pipeline and Setup	29
4.1	Dataset	29
4.1.1	Dataset Alignment	29
4.1.2	Dataset Preprocessing	31
4.1.3	Dataset Augmentation	33
4.2	Vision Transformer Model	34

4.3	Siamese Setting	35
4.3.1	Late Merge Siamese Network	35
4.3.2	Intermediate Merge Siamese Network	36
5	Observations and Results	38
6	Conclusion and Future Work	42
6.1	Conclusion	42
6.2	Scope for Future Work	43

List of Figures

1.1	Various biometric features used for authentication systems	7
1.2	The singular and minutiae points in a fingerprint	9
1.3	The original fingerprint and its corresponding thinned-ridge template	10
3.1	An Encoder block of Transformer (<i>Courtesy: [24]</i>)	15
3.2	Attention Mechanism simplified	16
3.3	Overview of a Vision Transformer (<i>Courtesy: [11]</i>)	18
3.4	Equal Error Rate Plot (<i>Courtesy: [21]</i>)	21
3.5	Overview of Shifted Patch Tokenization (<i>Courtesy: [27]</i>)	24
3.6	Locality Self-Attention Mechanism (<i>Courtesy: [27]</i>)	26
3.7	Overview of Shifted Patch Tokenization (<i>Courtesy: [27]</i>)	27
4.1	Original image and image after Singular point alignment	30
4.2	Fingerprint template before and after Preprocessing	31
4.3	True and Imposter Fingerprint Image Pair	32
4.4	Gaussian Noise and Random Brightness Image Augmentation	33
4.5	(a) Late-Merge Siamese Net (b) Intermediate-Merge Siamese Net . . .	35
4.6	Pipeline of the proposed experiment	37

List of Tables

5.1	Comparative study between Intermediate Merge and Late Merge Siamese Network	39
5.2	Performance of our model on various batch sizes	40
5.3	Performance of the model on various embedding sizes	40
5.4	Performance of our approach with and without the proposed modifications	41

Chapter 1

Introduction

The project for my thesis is centered around a very popular yet important aspect of today's technology-driven era. In such times, protection of identity and confidential data becomes extremely potent. The first means to do so were a simple lock and key. As technology advanced, the keys were replaced by passwords and locks were replaced by sophisticated authentication algorithms. Computerization of verification system also indicate that spurious authentication can lead to damage on multiple fronts including data, finance, privacy and so on. Credential replication, malware and retracing techniques have made it possible to retrieve passwords and abate the novelty of such methods [32].

In order to overcome the limitations of the traditional password protection regime, researchers shifted to biometric verification [33]. In such systems, a unique biological characteristic of an individual is used to verify one's identity. Biometrics comprise a broad spectrum of inputs including face, iris, heartbeat, sound, fingerprint and so on. Figure 1.1 gives a comprehensive list of biometric features which are currently being used for recognition systems. Amongst all of these, the most popular and widely-used biometric authentication feature are fingerprints. Fingerprints are a unique identification feature pattern for humans, consisting of ridges on fingers that let people grip things with their hands. The fingerprint scanner is at the heart of this automated verification system, and it's in charge of capturing images based on the

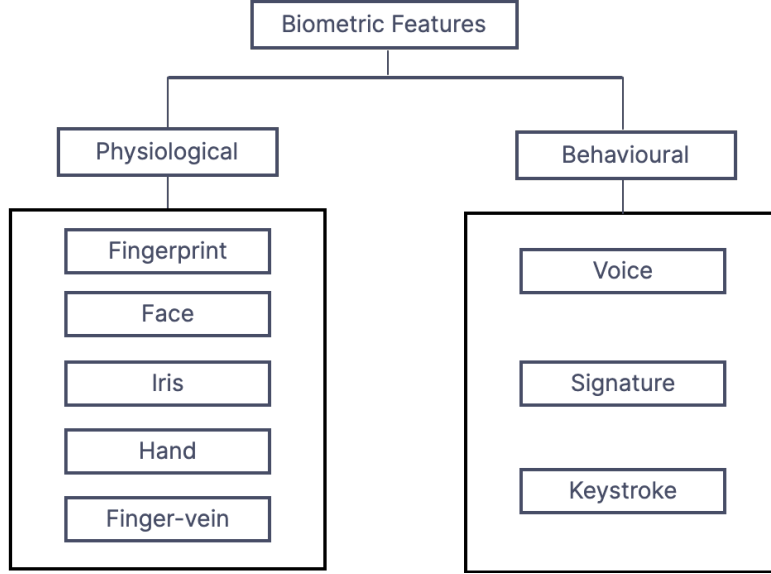


Figure 1.1: Various biometric features used for authentication systems

valleys and ridges of human fingers and matching them to stored patterns. Fingerprint matching is popular because of its scalability, convenience and ease of usage. My Bachelor thesis involves conducting systematic research on efficient fingerprint matching approaches, particularly in Deep Learning (DL) and use the learnings as motivation for developing a novel end-to-end fingerprint matching system.

1.1 Motivation of the Work

In the conventional fingerprint matching process, tedious preprocessing is involved in order to match two fingerprint images. Select data points are focused using computer vision algorithms or physical techniques and on the pretext of the similarity of the preselected features, the algorithm determines a match. These methods are not only outdated, but also computationally inefficient and not very reliable [40].

On the contrary, when machine learning (ML), particularly Deep Learning (DL), is employed for fingerprint matching, a majority of preprocessing is eliminated and the results procured are state of the art. This is because the biometric features which are extracted and learnt using Deep Learning models have superior discriminative

ability for inter-class samples and high similarity for intraclass samples as compared to standard models. That is, I am giving the model the freedom to determine the most suitable features for learning and make a more advised prediction. Moreover, DL based approaches are driven by data they are trained on, hence given a quality dataset, DL algorithms tend to effectively fit and replicate the results meticulously.

The motivation for choosing this particular topic as a part of my BTech thesis is two fold:

- There is great potential for biometrics in the state of the art breakthroughs which have been researched for numerous problem-statements. Our work is an effort to tap the potential and channelize it to the domain of fingerprints.
- The conventional fingerprint matching processes have been extensively discussed in the literature; however, the Deep Learning based approaches haven't been explored much for fingerprint matching.

1.2 Novelty of the work

This research conducted by us has several features which makes it novel in the domain of biometric, particularly fingerprint research. To the best of our knowledge, the following are the salient unique attributes of our research.

- I am the first to employ and implement Vision Transformers for Fingerprint Matching
- I am the first to ever use Vision Transformers in a Siamese setting for learning similarity between the two inputs and propose variations in its configuration.
- I have incorporated crucial modifications (SPT and LSA) which are inspired from [27], that tackle the problems faced by the model due to limited dataset size and increase accuracy.



Figure 1.2: The singular and minutiae points in a fingerprint

- I have proposed an end-to-end pipeline for fingerprint match detection.
- I have created custom and problem statement-focused dataset augmentation flow by drawing insights from similar works.

1.3 Fingerprint Nomenclature

Before diving deep into the research, there are some vital features in a fingerprint which have been used extensively and play a crucial role in the pipeline. They are as follows:

- **Singular Points:**

Singular point is defined as the topmost point of the innermost curving ridge. Noticeably, this will also be the point where the corresponding ridge curvature



Figure 1.3: The original fingerprint and its corresponding thinned-ridge template

reaches maximum in the whole template.

- **Minutiae Points:**

Minutiae points are specific feature points in a fingerprint image. Minutiae points have characteristic that these are either locations of ridge bifurcation or of a ridge ending in the fingerprint. Minutiae points are widely used to determine the uniqueness of a fingerprint image. As shown in the Fig 1.2, the green square indicates the singular point where as the blue circular outlines depict the minutiae points of the fingerprint.

- **Ridges:**

The curved lines in a fingerprint template image is referred to as ridges. While most ridges are continuous curves originating and terminating at edges of the fingerprint image the others either terminate at specific points called ridge endings or perhaps a ridge separates into two to form specific points referred to as ridge bifurcation. ‘These points are of significance, as location of these points correlates to uniqueness of fingerprint images. Usually we first thin the ridges of any user biometric template to a single-pixel thick ridges so as to process the template with more ease and precision. Fig 1.3 displays the ridges of a standard fingerprint image on the left and the same print image with thinned-ridges on the right for improving the data quality.

1.4 Thesis Organisation

In this section, I introduce the theme and subject of the thesis and the motivation driving the hypothesis. A separate subsection has also been dedicated in this section to highlight the novelty of the work and project the same as an original research having immense scope in the future. the organization of the rest of the thesis is as follows. In the following section, I introduce the key algorithms which have been used in the pipeline with focus on the fundamental concepts of Vision Transformers (ViT), which form the backbone of the entire model.

The next chapter talks about the entire pipeline of the proposed approach and divides each block into a separate section. The experimental setup and configuration for every block has been succinctly covered in the same chapter. I follow it up with the results and prime observations which have been derived from this experiment. Finally, I have presented the conclusion and the scope for future work on this lie of research.

Chapter 2

Literature Review

Ever since biometric authentication and verification has come into the limelight, researchers all around the world are deeply invested in making the algorithms more secure, reliable and failproof. Fingerprint matching, being at the heart of biometric authentication is evolving with time and rapidly advancing technology. Typically, there are two main components for fingerprint matching, fingerprint dataset feature extraction, and fingerprint similarity detection.

Initially, common fingerprint features such as minutiae points, ridges or singular points were extracted as potential features and matching was performed by comparison directly. [35] proposed a fingerprint matching solution where the minutiae points are refined and verified for matching. In order to enhance the performance, the authors of [23] combine level 3 features such as ridges and pores with fingerprint patterns and minutiae points to perform a holistic comparison using more number of fingerprint details. [22] proposed a hybrid approach where minutiae points and features from the texture of fingerprint are used for determining the similarity.

With the advent of Artificial Intelligence, research shifted their focus from traditional features mapping to more sophisticated machine learning and deep learning algorithms. A fingerprint-based genetic algorithm has been proposed by [38] which finds the optimal vector transformation between two different fingerprints. [15] de-

vice a Hidden Markov Model approach for detecting the similarity between two prints. Liu et.al. designed a deep learning model, namely Finger ConvNet in [29] as a novel approach to improve the speed and accuracy for fingerprint matching.

There are numerous machine learning approaches which have been employed for this problem statement. In particular, Siamese Networks stand out and outperform most existing approaches. Siamese Networks (will be discussed in detail in 3.3) basically comprise two exactly same arms which are trained in parallel. The model in each arm differs with the task at hand. In this regard, [7] proposed a Multi-Scale dilated Siamese Neural Network fingerprint matching. [2] applied adversarial learning and used multi-sensor data in a siamese setting for the same.

Vision Transformer was introduced by the Google Brain team in [11] as a powerful architecture for image classification. Since then, it has been adopted and modified for various problem tasks such as image encoding [41] and motion object detection [3]. This research intends to harness the superior image feature learning capability of Vision Transformers and apply it to the task of fingerprint matching. To the best of my knowledge, I am the first to experiment with and employ Vision transformers for similarity detection, which is fingerprint matching in our case.

Chapter 3

The Algorithms of Focus

3.1 Transformers

The Transformer is essentially an encoder-decoder model that was proposed in [39]. The Transformer architecture is one of the most widely used architectures in Natural Language Processing (NLP) and is drawing a lot of attention in other domains owing to its superior capabilities.

In a box, the Transformer consists of a stack of Encoders and the same number of decoders working sequentially. The Encoders are identical in structure, but they do not share each other's weights. The encoder is in charge of walking through the input time steps and encoding the whole sequence into a context vector, which is a fixed-length vector. Whereas, The decoder is in charge of reading from the context vector while stepping through the output time steps. Since I will only be employing the encoder part of the Transformer architecture, our focus is directed towards the same.

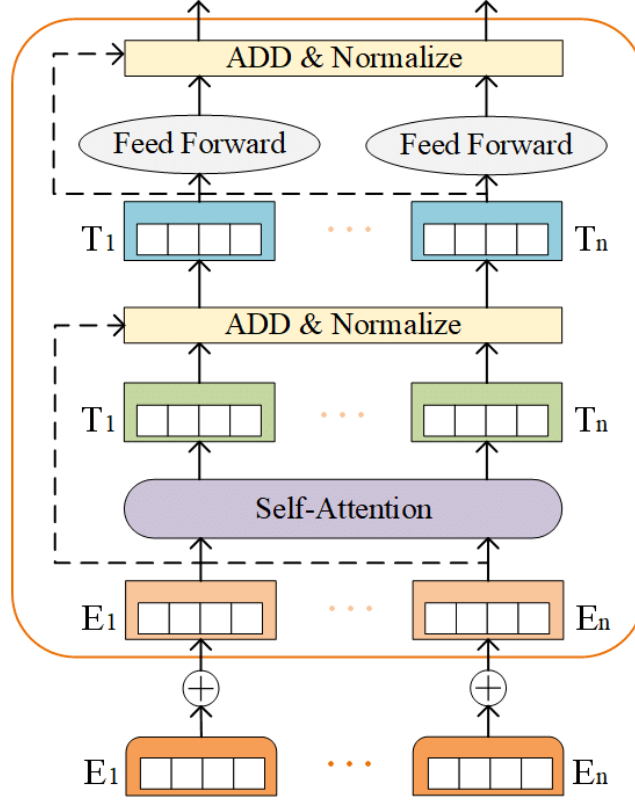


Figure 3.1: An Encoder block of Transformer
(*Courtesy: [24]*)

Each encoder can be broken down into independent two sub-components, the Attention module and the Feed Forward Network. The Attention module is basically an extension to seq-2-seq models that was proposed to overcome the drawback of the model with long sequences. The attention mechanism allows output to concentrate on input while producing output, and in the meanwhile the self-attention model allows inputs to interact by calculating attention of other inputs with respect to the input vector in focus. A simplified representation of the working of attention module is depicted in Fig 3.2. The Attention mechanism is explained as steps in the following.

1. Firstly each encoder input vector is multiplied with the three weight matrices $W(Q)$, $W(K)$, and $W(V)$. For each input vector, this matrix multiplication will yield three vectors: the key vector denoted by K , the query vector denoted by Q , and the value vector denoted by V . follows. A learnable linear projection is applied to each token to obtain Query, Key, and Value

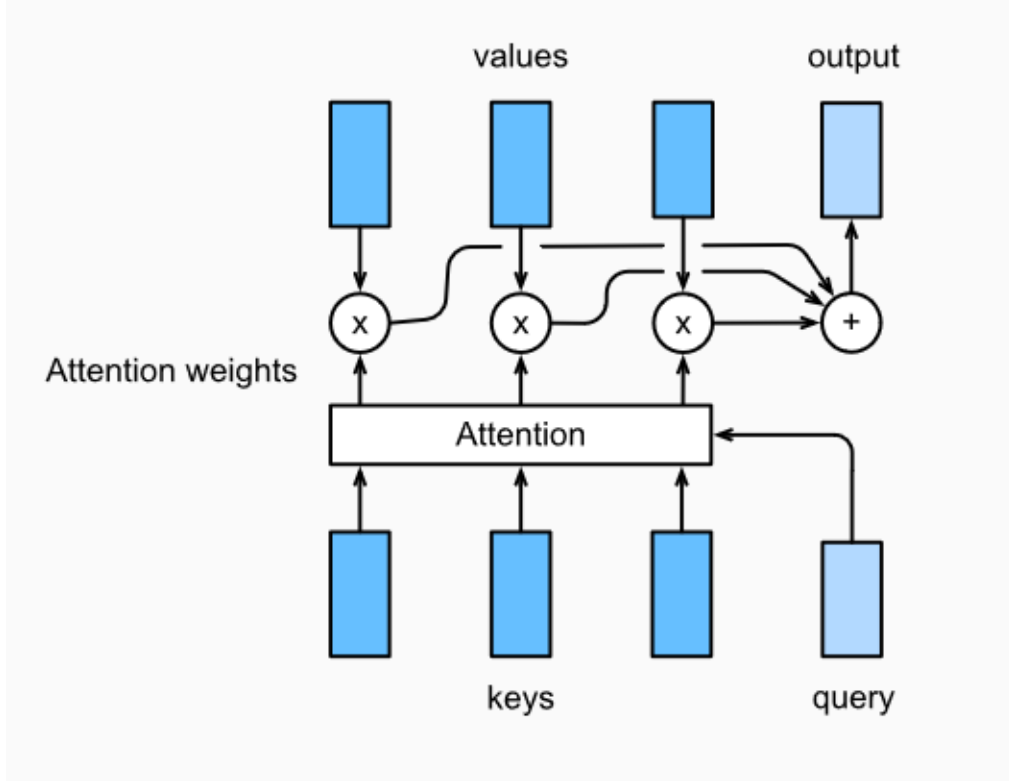


Figure 3.2: Attention Mechanism simplified

2. The current input's Query vector is multiplied with the key vectors from other inputs to generate a score, which is the second stage of determining self-attention. In other words, the similarity matrix is calculated, that is, $R \in R(N + 1)(N + 1)$, which indicates the semantic relation between tokens through the dot product operation of Key and Query. The off-diagonal components represent inter-token relations and the diagonal components of R represent self-token relations:

$$R(x) = xE_q(xE_k)^T \quad (3.1)$$

3. In the third stage, the score obtained in the previous step is divided by the square root of the key vector's dimensions (dk). For instance, the key vector in the original paper has a dimension of 64, thus that will be 8. The reason for this is that as the dot products grow larger, some self-attention scores become very little when the softmax function is used in the future

4. The softmax function will be applied to all self-attention scores calculated in relation to the query word in the fourth phase and the output vector of the function is multiplied with the value vector of the word in focus.
5. Finally, I sum the weighted value vectors obtained in the previous step to get the self-attention output for the word in focus.

Every input sequence is passed through the attention module. Mathematically, the entire attention mechanism can be summarized to the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (3.2)$$

The Transformer encoder [39] essentially consists of alternating layers of MLP and self-attention blocks. Layer normalization (LN) is applied before every block, and residual connections exist after every block. A detailed layout of an encoder block is shown in Fig 3.1. Mathematically, the working of the encoder is summerized by the following expressions:

$$z_0 = [x_{class}; x_p^1 \mathbf{E}; x_p^2 \mathbf{E}; \dots; x_p^N \mathbf{E}] + \mathbf{E}_{pos}, \quad \mathbf{E} \in R^{(P^2.C) \times D}, E \in R^{(N+1) \times D} \quad (3.3)$$

$$z'_l = SA(LN(z_{l-1})) + z_{l-1}, \quad l = 1 \dots L \quad (3.4)$$

$$z_l = MLP(LN(z'_l)) + z'_l, \quad l = 1 \dots L \quad (3.5)$$

$$y = LN(z_l^0) \quad (3.6)$$

3.2 Vision Transformers

Vision Transformers were first introduced in the research paper [11]. Vision Transformers outperformed the state-of-the-art models in Image classification on popular datasets.

Earlier, transformers would only be applied on sequences or audio signals as they are extremely compute heavy, especially their quadratic complexity when computing

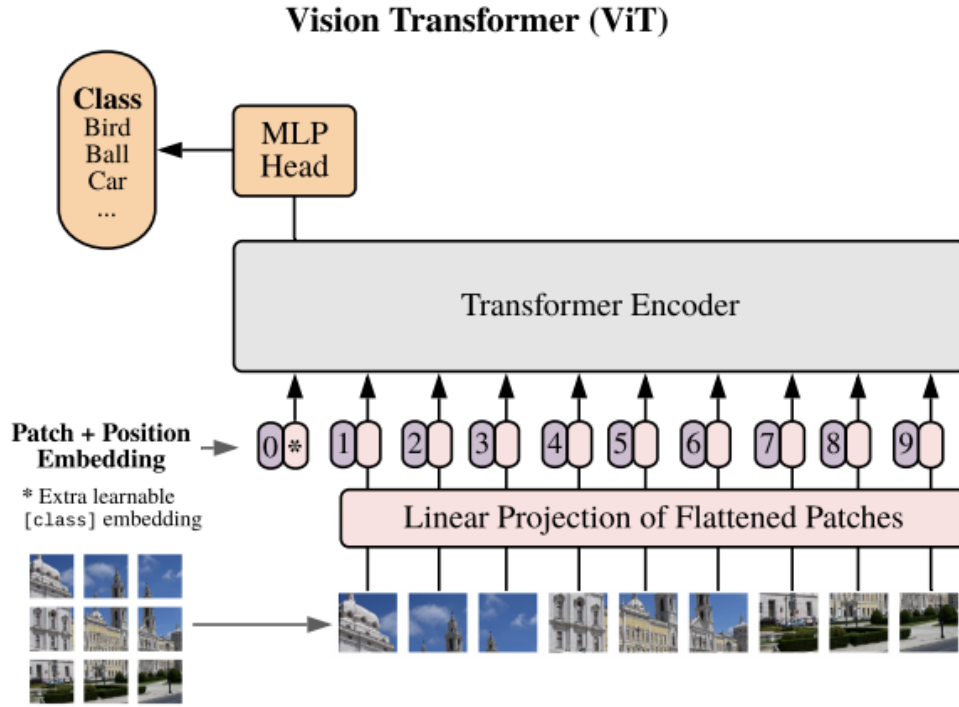


Figure 3.3: Overview of a Vision Transformer
(*Courtesy: [11]*)

the attention matrix. Hence in case of images, where each image is a n -dimensional matrix, the computation only gets worse. For instance, let's take a standard $28 \times 28 \times 1$ MNIST image [8]. In this case, the length of the vector would be 28×28 , and we'd need a $(28 \times 28)^2$ matrix to compute attention score, which is costly even for the best GPU's.

Vision Transformers (ViT) were introduced to tackle this problem for images. The paper [11] divided the image down into square portions in order to execute a form of lightweight "windowed" attention. To get here from NLP, each patch in the image problem is analogous to a word in the language problem. For producing a linear patch projection for feeding into the transformer, these patches are flattened and passed via a single Feed Forward layer. The Vision Transformer only uses the Encoder setup from the original architecture, and the internal organization is more or less the same to retain the performance. The embedding matrix E , as indicated

in the parent paper, is randomly generated and contained in this Feed Forward layer.

One issue with Transformers is that the order of a sequence is not automatically guaranteed because data is sent in at once rather than timestep-wise as in RNNs and LSTMs. To overcome this, the original Transformer paper [39] recommends using Positional encodings/embeddings, which place the inputs in a specific order. Hence for ViT, the positional embedding matrix E_{pos} is randomly generated and added to the concatenated matrix containing the patch and embedding projections. Moreover, to enable the output of a single probability rather than a sequence of vectors, [11] adopted the approach from BERT [9] and concatenated a learnable $[class]$ parameter with the patch projections. If the $[class]$ token is not used during the process, only positional embedding is added to the output of visual tokens. The formulation to apply tokenization to the patch embedding layer is given in the equation below:

$$S_{pe} = \begin{cases} [x_{cls}; S(x)] + E_{pos} & \text{if } x_{cls} \text{ exists} \\ S(x) + E_{pos} & \text{otherwise} \end{cases}$$

where $x_{cls} \in R^{dS}$ is a class token and $E_{pos} \in R^{(N+1)dS}$ is the learnable positional embedding.

The Transformer Encoder’s outputs are then used to predict the image class probability using a Multilayer Perceptron Layer (MLP). The input features effectively capture the core of the image, making the MLP head’s classification task easier. Multiple outputs are provided by the Transformer but the MLP head only receives the output relating to the particular $[class]$ embedding and the other outputs are disregarded. The output gives the probability distribution of the classes of the image. A comprehensive overview of the structure of a Vision Transformer is given in Fig 3.3.

3.3 Siamese Network

A Siamese Neural Network is a modification of artificial neural networks (ANNs) that are used widely for machine learning. It is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. Essentially, the two arms have the same configuration with the same parameters and weights and parameter updating is mirrored across both sub-networks. There are three main types of Siamese Networks that are classified on the basis of their merging position in the architecture [12]: **(1)** Late-Merge Siamese Networks, **(2)** Intermediate-Merge Siamese Networks and **(3)** Early-Merge Siamese Networks. I will be using two of these three variations of Siamese Networks for our experiment.

Traditionally, ANNs learn about the input and their corresponding output class probabilities using the softmax equation. On the other hand, Siamese NNs learn about the semantic similarity between the two inputs using distance metrics based learning and output their similarity score. Since code clone detection is essentially computing the similarity between two code fragments, the Siamese Neural Network architecture holds promise. Moreover, several research studies have shown that Siamese Networks yield better embeddings [16]. They are also well-suited for tasks which have few examples per class due to pairwise learning and not point-wise classification as in other Artificial Neural Networks (ANNs) [20]. Siamese Networks are widely used in recognizing handwritten checks, automatic detection of faces in camera images, and matching queries with indexed documents [16] [34].

In our research, I have used a Vision Transformer architecture in place of a traditional Fully Connected CNN or RNN so that the power of ViT can be harnessed for similarity detection. Basically, we input two image vectors in the siamese model and procure an output probability at the tail which is used to determine the label of the image pair, i.e. Genuine Pair (label as 1) or a Imposter Pair (label as 0/-1).

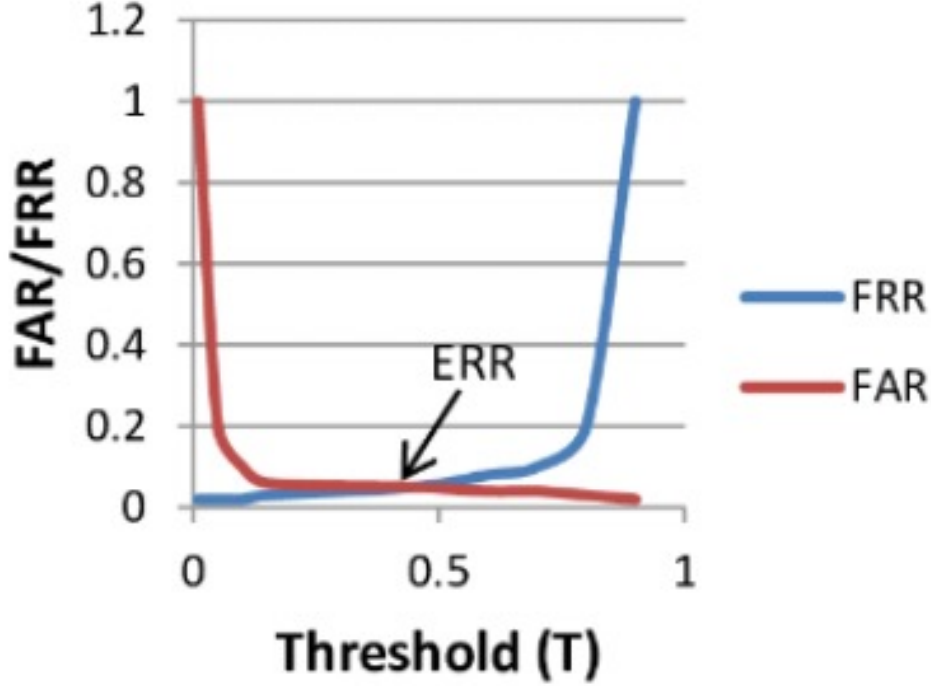


Figure 3.4: Equal Error Rate Plot
(Courtesy: [21])

3.4 Threshold Calculation

As mentioned in the section 3.3, I procure an probability output from the Siamese network, whose value varies between 0 and 1. I needed a threshold which determines the closed continuous range of values that can be interpreted as true and false label. The most common and rudimentary technique is to set the threshold to 0.5. Therefore, any probability below 0.5 is comprehended as false and one above it gets the label as true. This technique is only suitable for a data whose output have a uniform distribution. Since in most cases, the data has an erratic distribution, we have to resort to a method which is more empirical and mere hit and trial.

Assuming T is the threshold. Hence, output probability (P) $\geq T$ implies the pair of images are identical else they are not. I have designed a custom method to empirically calculate the threshold. Initially T is set to 0.5 for the first epoch. The following are the steps involved in calculating T for the model:

- Firstly, for every epoch the ROC Curve [19] Loci of all the outputs are derived.

The ROC Curve essentially is a plot between the TPR Positive Rate (TPR) of the model vs the False Positive Rate (FPR) of the model.

- The point on the ROC curve where $abs(TPR + FPR - 1)$ is minimum is interpreted as T for that epoch during training. This point is also known as the **Equal Error Rate (EER)** threshold. In the given Fig 3.4, The Equal Error Rate threshold is the intersection of the FRR (which is $1 - TPR$ for our ROC Curve) and FPL loci.
- The point procured from the above step is added to a list of T's where T_i is the threshold calculated for epoch i .
- The mean of the list of thresholds is taken as the threshold for the next epoch.

The threshold T_{last} is considered the final threshold for the model. Mathematically for every epoch i ,

$$T_i = roc_curve(min(abs(TPR + FPR - 1)) \quad (3.7)$$

$$T_{i+1} = mean(list[T_i]) \quad (3.8)$$

3.5 Shifted Patch Tokenization

In a original ViT pipeline, the input images are divided into patches that are then linearly projected into tokens. Originally, ViT had to precede pre-training on a large-size dataset such as JFT-300M.

Since ViT's don't use Convolutional Neural Networks (CNN), ViT structurally lacks locality inductive bias (weights or assumptions of model to learn target and generalize beyond train data) as compared to CNNs, and they require a very large magnitude of training data to obtain satisfactory visual representation and compensate to an acceptable extent for the low bias. the magnitude of number of elements for a sufficiently large data is in millions. In our case, since our dataset is limited

to few hundred test subjects, using the traditional ViT approach yields in low accuracy. Hence some modifications have been implemented to overcome this issue and improve the accuracy of the entire model. These modifications have been inspired from [27]. There are two main problems arising due to the small size of our dataset. The problem and the solution incorporated to tackle it have been discussed in detail.

3.5.1 Problem

The first setback to the model is a lack of quality tokenization. ViT linearly projects each patch to a visual token by dividing a given image into non-overlapping patches of identical size (16,16 in our case). Each patch is given the same linear projection here. This is beneficial as tokenization of the ViT has the permutation invariant characteristic, allowing for good patch embedding, that is, it is impermeable to the variance in the permutation of the patches and the embedding will be uniform for all the patches of the image.

On the other hand, non-overlapping patches allow visual tokens to have a tiny receptive area. Non-overlapping patch tokenization typically has a narrower receptive field than overlapping patch tokenization with the same down-sampling ratio. ViT tokenizes the patches with too few pixels due to small receptive fields. This means that the spatial relationship between neighbouring pixels of two overlapping patches is not well captured, which hinders learning as the data which will be fed to the transformer will have missing information. Since I did not have enough data for the model to generalize, hence this conundrum needs to be addressed.

3.5.2 Solution

Shifted Patch Tokenization (SPT) [27] has been introduced in the proposed pipeline to combat the issue of low receptive fields during patch tokenization process. The concept of SPT was adopted and derived from Temporal Shift Module (TSM) [28]. TSM is effective temporal modeling which basically shifts some temporal channels

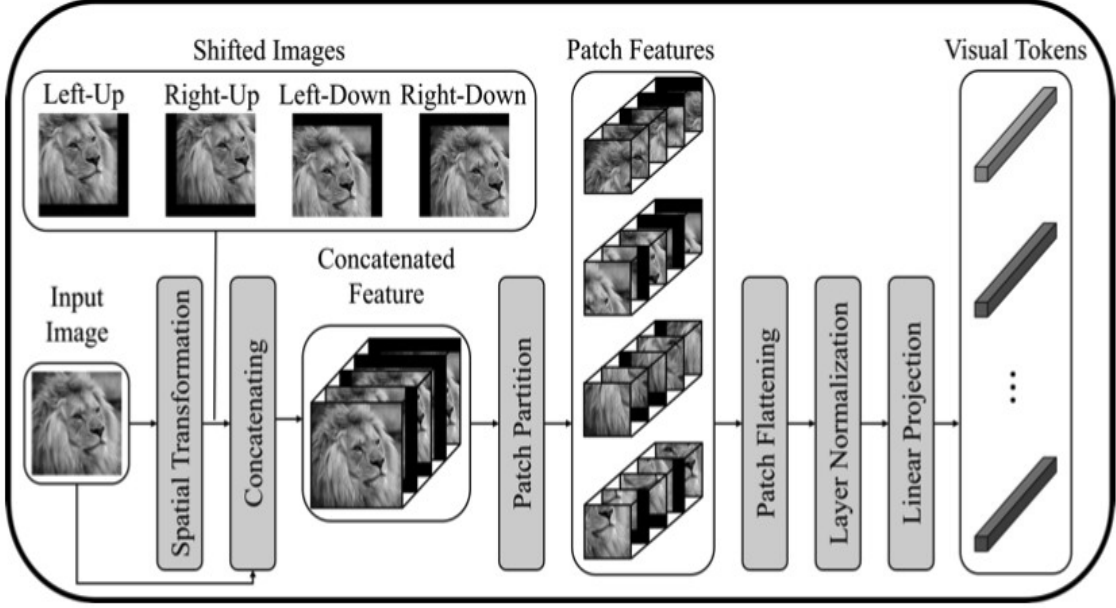


Figure 3.5: Overview of Shifted Patch Tokenization
(*Courtesy: [27]*)

of features. Inspired by this, effective spatial modeling has been proposed that tokenizes images that are shifted spatially together with the input image. In this method, the core idea remains the same, but a modification has been made so that more spatial information of the patch can be embedded into the visual tokens and thereby, increase the locality inductive bias for the Vision Transformer model.

For image in the dataset, first the image is shifted in its four diagonal directions by a fixed number of pixels. This process generates four new images, which are then taken and concatenated with the original un-shifted image along the dimension of image channels. For our dataset, I use grayscale images, hence the number of channels of the original image is 1. After concatenating, the number of channels of the new combined image becomes 5. Following this, non-overlapping patches of size (p, p) are extracted from the image. then for embedding the patches into visual tokens to input to the transformer, three processes are sequentially performed on the procured patches, patch flattening, layer normalization and finally linear projection into visual tokens. A graphical representation of the idea and the entire process has

been given in the Fig 3.5. The whole process is formulated as Eq. 3.9 :

$$S(x) = LN(P([xs^1s^2...s^{N_s}]))E_S \quad (3.9)$$

In the above equation, $s^i \in R^{HWC}$ represents the i-th shifted image according to S and $E_S \in R^{P^2.C.(N_s+1)xd_S}$ indicates a learnable linear projection. d_S represents the hidden embedding dimension of the encoder, and N_S denotes the number of images shifted by S .

Mathematically, Equation 3.10 depicts the image dimension transformation after concatenation of shifted images to the original image as mentioned above and equation 3.11 talks about the final dimension of the input vector to the ViT after image flattening and patch projection. Here, p denotes the patch size, $image_h$ and $image_w$ denote the height and width of the original image respectively.

$$[1, image_h, image_w, 1] => [1, image_h, image_w, 5] \quad (3.10)$$

$$[1, image_h, image_w, 5] => [1, (\frac{image_h}{p}, \frac{image_w}{p}), (p, p, 5)] \quad (3.11)$$

3.6 Locality Self-Attention

3.6.1 Problem

The second problem that was encountered was the inefficient functioning of the self-attention mechanism. Because image data has a far larger feature dimension than natural language text sequences and audio signals, the number of embedded tokens is always high, even for a small patch of (16, 16, 1) size image. As a result, the distribution of attention scores in the tokens' attention module smooths out.

In other words, there is an issue where ViTs are unable to attend to significant visual tokens on a local level. This can result in a lot of redundant attention that

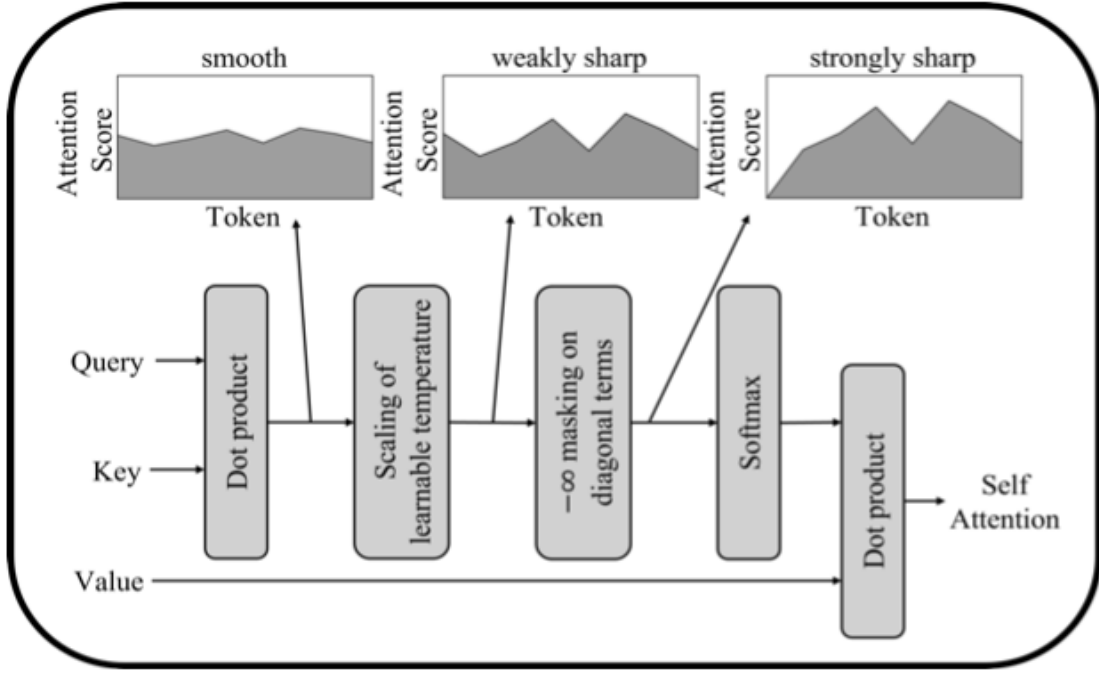


Figure 3.6: Locality Self-Attention Mechanism
(Courtesy: [27])

can't focus on a certain class which holds important information. ViT can easily focus on the background and miss the shape of the target class as a result of this superfluous attention.

3.6.2 Solution

To combat this issue, Locality Self-Attention (LSA) mechanism has been introduced. Graphically, the mechanism can be explained by Fig 3.6. Generally, a softmax function can control the smoothness of the output distribution through temperature scaling [17]. LSA primarily sharpens the attention score distribution by learning the softmax function's temperature parameters. Moreover, the self-token relation is discarded, which forcibly suppresses the diagonal components of the similarity matrix computed by Key and Query when attention score is calculated. This masking increases the attention relatively between different tokens, making the distribution of attention scores much sharper. This can be clearly inferred from the given Fig 3.6. As a result, LSA increases the local inductive bias of the encoder by making its attention module more locally focused.

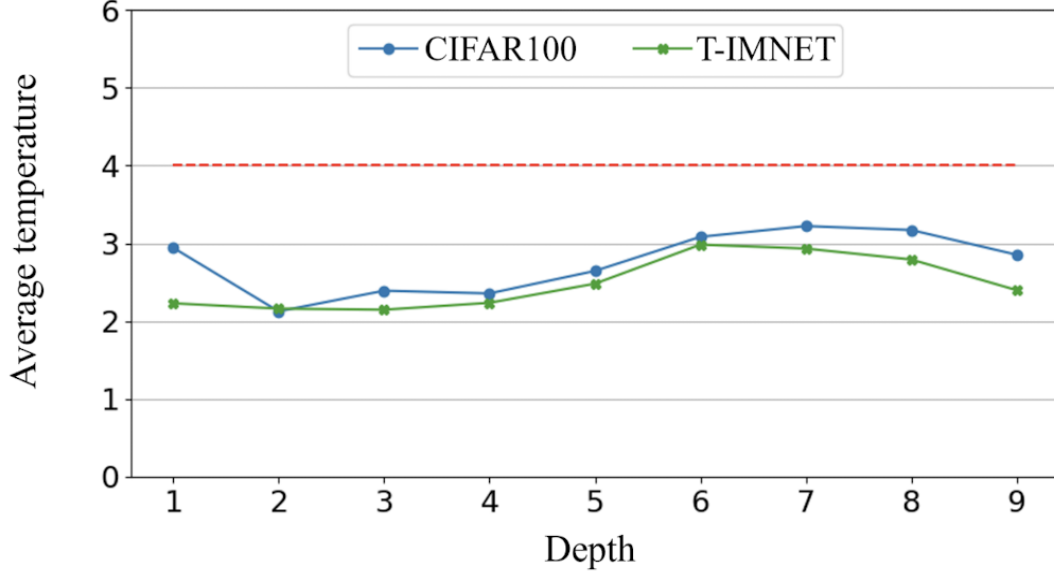


Figure 3.7: Overview of Shifted Patch Tokenization
(*Courtesy: [27]*)

The core of LSA is diagonal masking and learnable temperature scaling, a brief introduction of which has been given in the previous paragraph.

Diagonal Masking

By essentially eliminating self-token relations from the softmax process, diagonal masking helps to provide greater scores to inter-token interactions. Specifically, diagonal masking forces $-\infty$ on diagonal components of R of Equation 3.1. As a result, ViT’s attention is drawn away from its own tokens and toward other tokens.

Learnable Temperature Scaling

The second technique used in LSA is the learnable temperature scaling, which enables ViT to calculate the softmax temperature on its own during the learning process. Fig 3.7 shows the average learned temperature based on depth when the softmax temperature is used as the learnable parameter in Eq. 3.12. It can be noticed in the plot that constant temperature of traditional ViT is higher than the average learned temperature. In general, lower the temperature of softmax, sharper

the score distribution. As a result, the learnable temperature scaling sharpens the attention score distribution. Based on Eq. 3.12, the LSA with both learnable temperature scaling and diagonal masking applied is defined mathematically by:

$$L(x) = \textit{softmax}(R^M(x)/\tau)x E_v \quad (3.12)$$

where τ is the learnable temperature.

In other words, LSA overcomes the smoothing problem of the attention score distribution in the attention module of ViT.

Chapter 4

Experimental Pipeline and Setup

4.1 Dataset

I have used a fingerprint dataset which has been released by the researchers at IT Kanpur. In the dataset, fingerprint impressions of around 1378 different subjects were taken. For every subject, four identical fingerprints were extracted, that is, 4 prints from the same finger of the same subject having different orientations were taken. The nomenclature for denoting the fingerprint image is $\langle \text{subject}_i\text{-fingerprint}_j \rangle$ where i is the id of the subject and j is the id of the fingerprint of subject i .

4.1.1 Dataset Alignment

On reviewing and analysing the database, I noticed that the identical fingerprints have the same features but have varied translations with respect to each other. For optimizing the quality of the dataset, all identical samples should be invariant to rotation and translation.

To address this issue, I have proposed and implemented a solution involving translating and aligning the images based on their singular points [5]. As described in Section 1.3, Singular points are defined as points where the orientation field on the

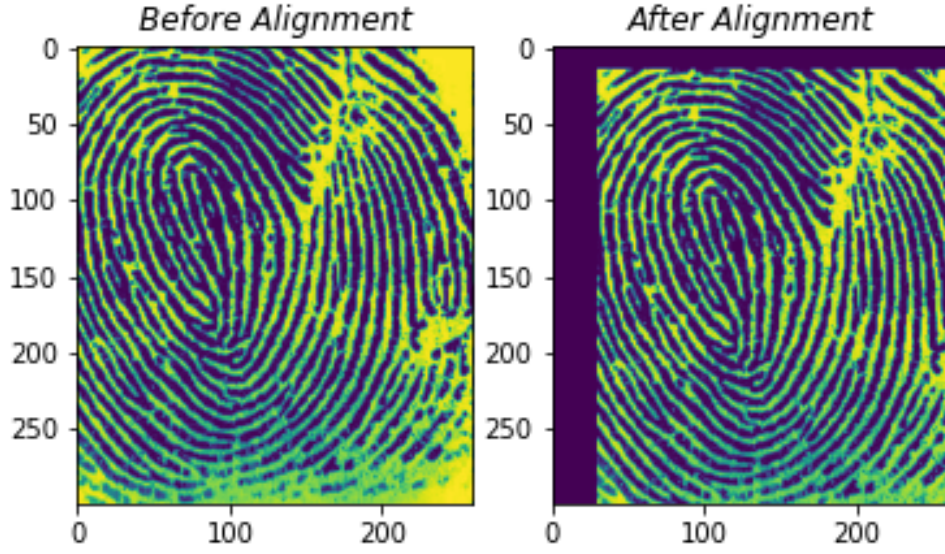


Figure 4.1: Original image and image after Singular point alignment

fingerprint contours is discontinuous or where the ridge curvature is maximum. It is important to note that commonly, there is only one singular point for every finger, hence they are popularly used as a feature for registration and identification.

In the proposed alignment algorithm, the singular point coordinates of all identical fingerprints are extracted and stored (so 4 points for every subject). For images of every subject, the arithmetic mean of all the coordinates of identical prints are calculated and stored separately. Essentially, for every set of 4 prints of the same subject, the new designated singular point is the average calculated above. Mathematically for every subject i ,

$$(x_{i_{new}}, y_{i_{new}}) = \left(\sum_{j=1}^{j=4} \frac{x_{ij}}{n}, \sum_{j=1}^{j=4} \frac{y_{ij}}{n} \right) \quad (4.1)$$

where $j \in [1, n]$ and $n = 4$ in our case.

After obtaining the updated singular points, the images are translated such that the new singular point of the image lies at the coordinates procured. As seen in Fig 4.1, the original image is on the left and the translated image according to the

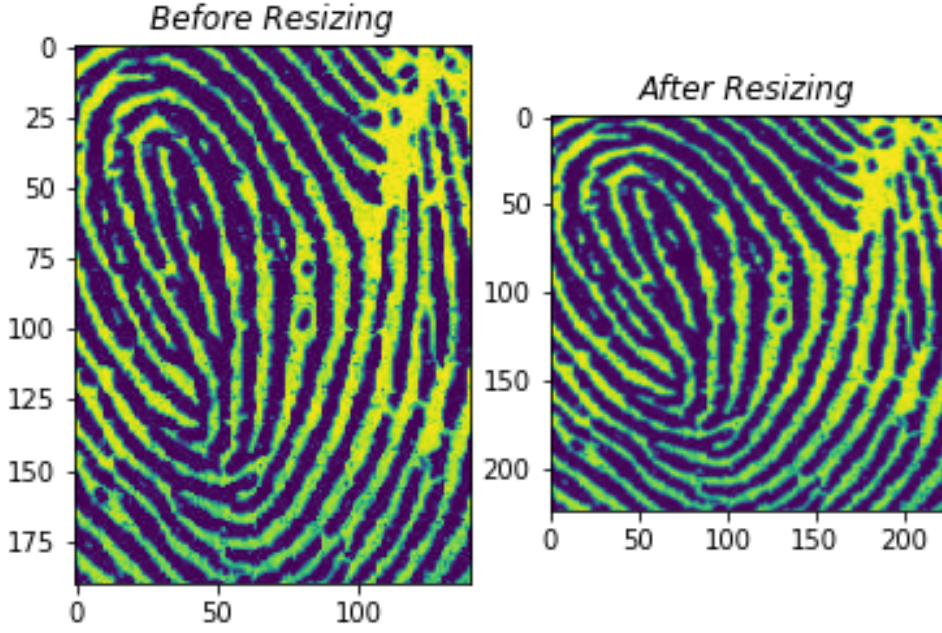


Figure 4.2: Fingerprint template before and after Preprocessing

aforementioned algorithm is on the right. The undefined portion of the image after alignment (the dark portion in Fig 4.1) is cropped out to avoid learning from pixels by the model which do not exist.

4.1.2 Dataset Preprocessing

In the previous subsection, since the translation of images is not uniform for images, I noticed that the size of images in the dataset becomes erratic and cannot be used directly for training the model. For resolving this issue, I first determined the dimensions of the minimum portion in an image which is common to all images in the entire dataset. The portion (size of (190, 140) in our case) is exacted from every image regardless of its index.

Following this, the image is resized to the dimensions (224, 224) as prescribed in [11] using inter-area interpolation method systematically [26]. Inter_area interpolation essentially is a bilinear interpolation function that does the image resampling using

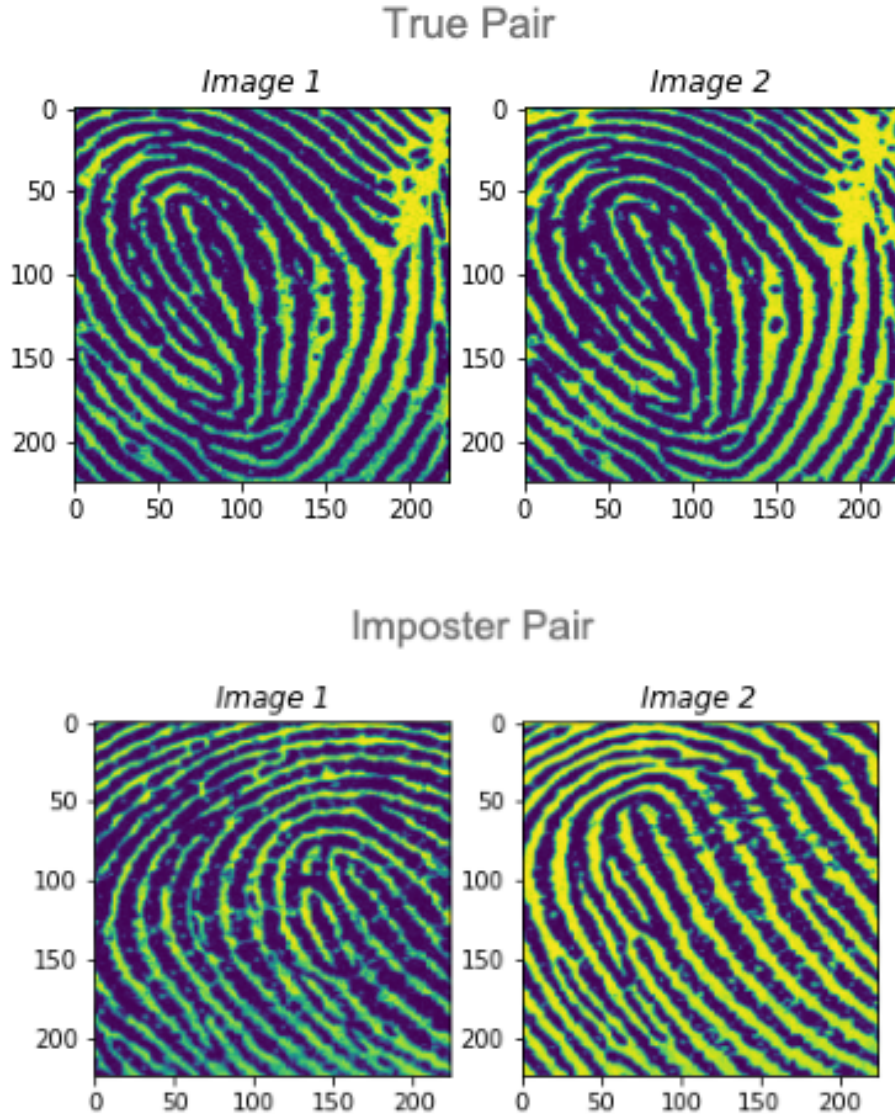


Figure 4.3: True and Imposter Fingerprint Image Pair

pixel area relation similar to `inter_nearest` interpolation. Hence the final dimensions of the image that is obtain after these steps is (224, 224). Fig 4.2 represents the input and output image for dataset preprocessing step in the pipeline.

Following this, the dataset is passed through a function which generates genuine and imposter pairs of images in the ratio of 1 : 1. For generating genuine pairs, I paired each image for a subject to its identical part, so for every subject, about 6 genuine pairs were generated. On the other hand, for generating imposter pairs, image of any subject were paired to any other image of a different subject, ensuring

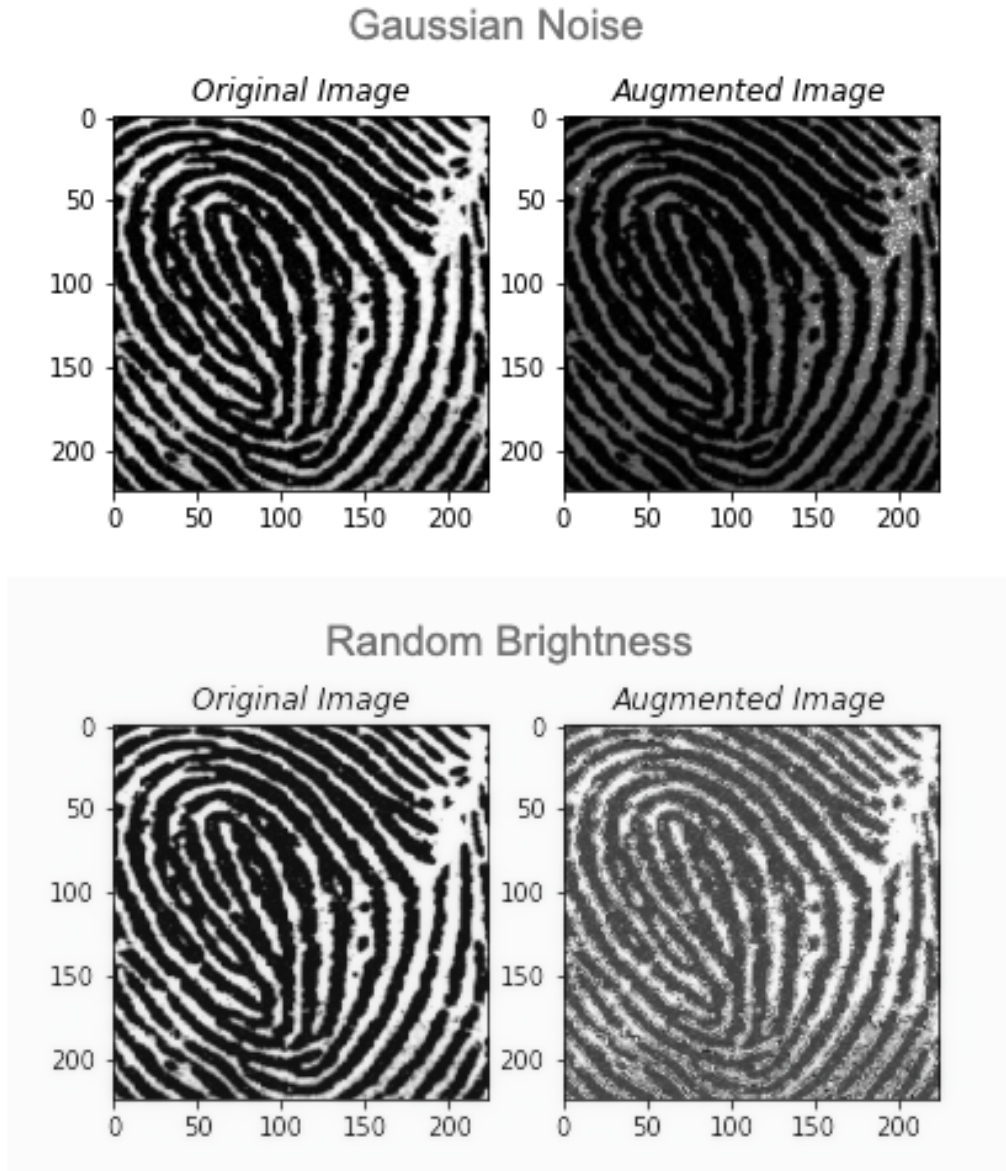


Figure 4.4: Gaussian Noise and Random Brightness Image Augmentation

that the number of pairs created are same as true pairs. An example of true and imposter pair from the dataset is given by Fig 4.3. A label of 0 for imposter pair and 1 for genuine pair is introduced corresponding to the type of pair created.

4.1.3 Dataset Augmentation

Since Deep Learning Models are usually data hungry [1], different methods are applied to increase the size of the dataset while maintaining the quality of the original set. Data augmentation refers to approaches for increasing the amount of data by

adding slightly changed copies of current data or creating new synthetic data from existing data. When training a machine learning model, it functions as a regularizer and helps reduce overfitting [37]. It is also closely related to oversampling data analysis.

Our pipeline allows application of any two of these three image augmentation approaches selected at random such that no two techniques are the same for a single pair. The augmentation techniques implemented are:

- Addition of Random Gaussian Noise [30]
- Random change in Image Brightness
- Random change in Image Contrast

An example of original and their corresponding images have been given by Fig 4.4. Each technique has probability of 1.0 for executing and the new pairs which are created are assigned the same label as that of the parent pair and are appended to the original image pair dataset.

4.2 Vision Transformer Model

Typically, a input vector embedding is passed to the bottom-most Encoder in a transformer. An abstraction that is common to every encoder is that it receives an embedding from its previous encoder directly below, of the same length. The size of this hyperparameter is usually set during fine-tuning, however in NLP, it is the length of the longest sequence in the training dataset. For our research I have experimented with two embedding dimensions, 768 and 512.

In the shifted-patch tokenization of the ViT pipeline, there are multiple approaches for converting the patches to visual tokens. I have employed and experimented with a Dense layer of Neural Networks and a Convolutional Layer for the purpose. the

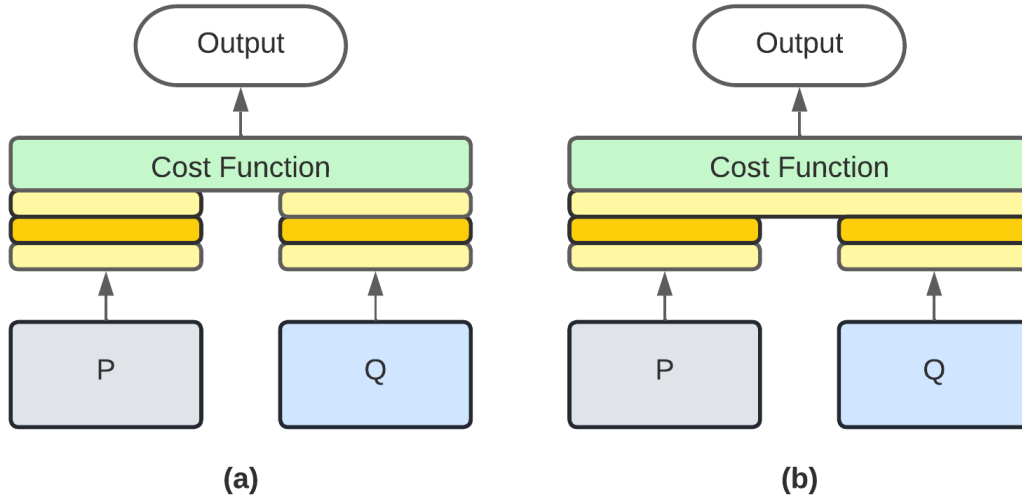


Figure 4.5: (a) Late-Merge Siamese Net (b) Intermediate-Merge Siamese Net

output from both the techniques is the aforementioned embedding with a fixed size. I stack 8 encoders and employ 12 attention heads in our ViT architecture. Moreover, the hidden dimension in the final Multi-Perceptron Layer (MLP) is twice the dimension of the input embedding for every encoder and the non-linear activation function used in the MLP is GeLU [18].

4.3 Siamese Setting

I have experiment with two different types of Siamese settings. Apart from the structural difference and the output layer of MLP, the parameters and the overall configuration of both the settings is kept exactly the same. The basic overlay of the two networks is given by Fig 4.5.

4.3.1 Late Merge Siamese Network

In this setting of Siamese Network, the left and right arms of the network remain distinct throughout the model. Hence for the two input vectors, let's say $V1$ and $V2$, two outputs are received, say $out1$ and $out2$. I could not use Cross-Entropy Loss to train the model for learning similarity between the two outputs, hence I employed

Contrastive Loss Function. This loss function takes two embedding vectors as input and equated the Euclidean distance between them if the label is 1 or behaves like Hinge Loss Function[13] if label is 0. A threshold is applied on the output of the model to classify the code snippets. The expression for Contrastive Loss function is given in Equation 4.2 where y is the label, $f1$ and $f2$ are the output vectors, m is the margin which is set at 2 and $d = ||f1 - f2||_2$ is the euclidean distance between $f1$ and $f2$.

$$L(f1, f2, y) = d * y + (1 - y) * \max(m - d, 0)^2 \quad (4.2)$$

It is important to note that the output layer of Late Merge Network has two neurons, i.e. it outputs two different probabilities. For empirical purposes, we also train our model using Cosine Embedding Loss function. The formula for the same is given in Equation 4.3.

$$L(f1, f2, y) = \begin{cases} 1 - \cos(f1, f2), & y = 1 \\ \max(0, \cos(f1, f2) - m), & y = -1 \end{cases} \quad (4.3)$$

4.3.2 Intermediate Merge Siamese Network

On the other hand, in this architecture, the two inputs $V1$ and $V2$ are given separately to the model. They follow the pipeline separately while sharing the weights up until the penultimate layer in the Dense Network of the Siamese architecture. Before the output layer of the Siamese network, the absolute difference of the two internal vectors is calculated and then fed to the final layer. Since the output is a single vector, this architecture also uses Binary Cross-Entropy Loss function for training the model. Mathematically, the loss function can be depicted in Equation 4.4

$$L(|f1 - f2|, y) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (4.4)$$

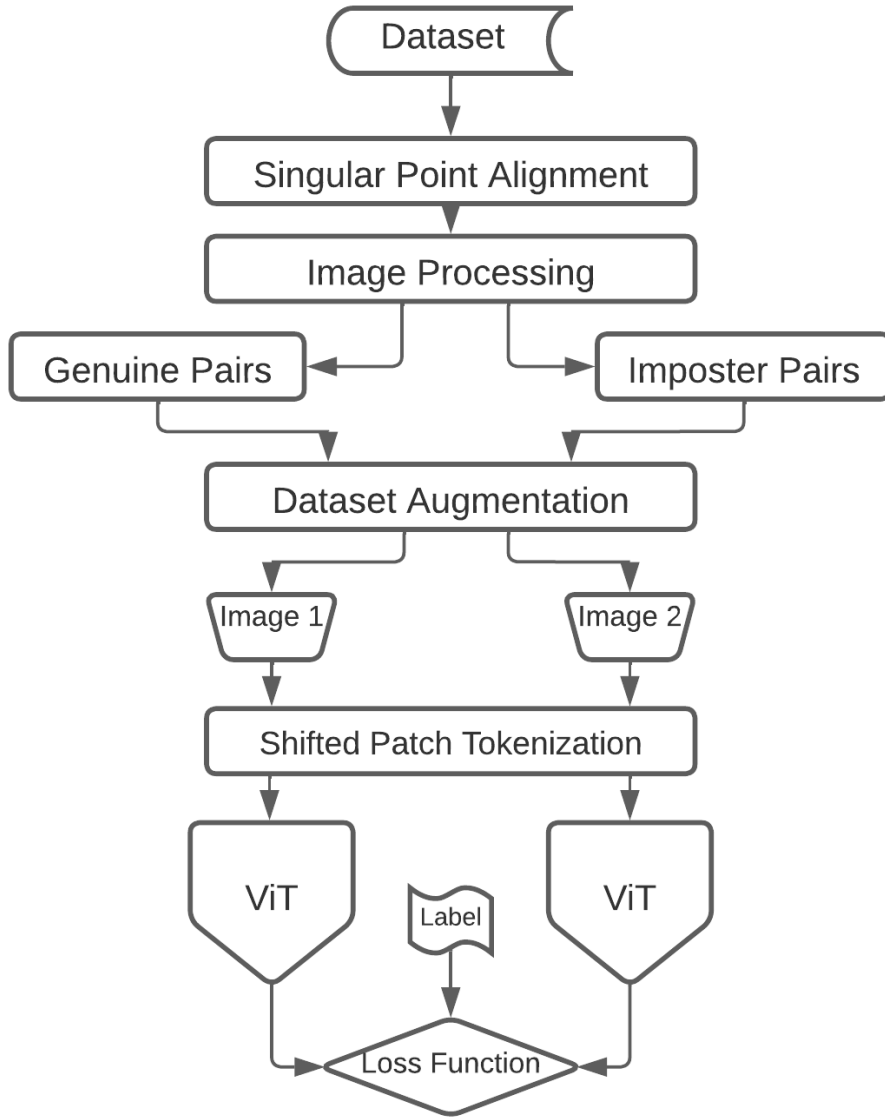


Figure 4.6: Pipeline of the proposed experiment

In the case of Intermediate Merge SN, the output layer has a single neuron, i.e., it outputs a single probability.

The entire experimental pipeline of the proposed approach is depicted by Fig 4.6.

Chapter 5

Observations and Results

The implementation and testing for the proposed research has been done in Python and the framework used for building and running the model is Pytorch.

There are a lot of hyperparameters and functions involved when training a complex DL model. Since experimenting with every permutation and combination of every parameter is not a feasible approach, some parameters are fixed based on empirical data and research conducted for the same [6]. In the model, I split the dataset into train and test data in the ratio 80-20. The optimizer used for training the model is AdamW [31] as it is an improvement over Adam by incorporating variable weight decay. I employ a scheduler which reduces the learning rate on formation of plateau for validation accuracy. The model has been trained for 10 epochs on Nvidia Tesla V100 32gb GPU with learning rate of 0.001 and multiple batch size configurations.

As mentioned in Section 4.3, I train the model on two types of Siamese settings, Intermediate-Merge (I-M) and Late-Merge (L-M) Siamese Network. It is possible to train both these networks with two loss functions each, Contrastive Loss and Cosine Embedding Loss for I-M and BCE Loss and BCEWithLogits Loss for L-M. Essentially, BCEWithLogits loss combines the BCELoss with a Sigmoid layer in one single class. This makes it more numerically stable than using a plain Sigmoid followed by a BCELoss as one can take advantage of the log-sum-exp trick for

Table 5.1: Comparative study between Intermediate Merge and Late Merge Siamese Network

Siamese Network	Loss Function	Accuracy	TPR	TNR
L-M	Cosine Embedding	0.572	0.387	0.723
L-M	Contrastive	0.697	0.817	0.628
I-M	BCE	0.5	0.25	0.691
I-M	BCEWithLogits	0.52	0.032	0. 677

numerical stability by combining the operations into one layer.

As it can be inferred from Table 5.1, Siamese model using Contrastive Loss function outperforms all models having other loss functions. Note that, all other parameters were kept the same during this experiment. In particular, it is evident that Late Merge Siamese better perform significantly better than Intermediate Merge Siamese Network, regardless of the loss function used. I believe this is due to the fact that, merging at a later stage helps the model exploit the input images separately thereby allowing the model to be more discriminative while learning features [12]. On the other hand, merging at an intermediate position allows the model to initially exploit the input image features and then these fused features are used further. This reduces the discriminative ability of the ViT in focus. For L-M Siamese setting, I choose Contrastive Loss over Cosine Embedding Loss [4] as the former provides superior results then the latter. Hence, I choose and propose L-M SN with Contrastive Loss for the approach.

Hyperparameters play an important role in determining the final performance of the model [36]. In particular, it is extremely crucial to select a proper batch size [14]. It is seen from Table 5.2 that when the model is trained with a batch size of 128, it produces the best results. In general, using larger batch sizes leads to degradation of significant degradation in the quality of the model as these methods tends to converge to sharp minimizers which lead to poor generalization [25]. This phenomena accounts for the lesser accuracy of the model with batch size 256. On

Table 5.2: Performance of our model on various batch sizes

Batch Size	Accuracy	TPR	TNR
32	0.470	0.617	0.408
64	0.575	0.696	0.448
128	0.697	0.817	0.628
256	0.607	0.704	0.562

Table 5.3: Performance of the model on various embedding sizes

Embedding Size	Batch Size	Accuracy
256	128	0.618
512	128	0.697
768	128	0.652
1024	128	0.607

the other hand, using smaller batch sizes causes the gradient estimation to become noisy and have large variance. Moreover, using a smaller batch sizes causes over-fitting [10] on the mini-batch distribution instead of the actual dataset distribution which results in decreased accuracy. This explains the poor performance of the model with 32 (very poor) and 64 batch size.

Keeping all parameters constant, I also tested for the most suitable embedding size for the model. The results are given in Table 5.3. Even though 768 is the most widely used embedding for transformers, in our approach, it is not the most suitable embedding. The results indicate that empirically, 512 is the most optimal embedding size for our model.

Table 5.4: Performance of our approach with and without the proposed modifications

Shifted Patch Tokenization	Locality Self Attention	Accuracy
No	No	0.654
No	Yes	0.670
Yes	No	0.688
Yes	Yes	0.697

Lastly, I test the relevance of the adopted modifications, namely SPT and LSA for my proposed approach. Table 5.4 shows that SPT alone increases the accuracy of the model by approximately 3.4% and in combination with LSA, the overall performance of the model is elevated by 4.3%. This proves that the proposed modifications tackle and overcome the aforementioned problems in our Vision Transformer model with significant success.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Deep Learning is one of the most promising areas of research which has shown a lot of potential especially in areas such as robotics, cryptography, and computer graphics. However it has been relatively less explored in the field of biometrics. With the advent of powerful models like Vision Transformers, there is a lot of scope for research and innovation in biometric cryptosystem research. As part of this thesis project, I have proposed a completely novel pipeline employing Vision Transformers in multiple Siamese settings for the task of fingerprint matching. I have also incorporated Shifted Patch Tokenization and Locality Self Induction as novel modifications for our problem statement. I am the first to introduce such a pipeline and harness the power of Vision Transformers for similarity detection. Our research aspires to provide a starting point and foundation for innovation in fingerprint matching.

6.2 Scope for Future Work

There is a lot of scope for future extensions of this work. Some of the potent ones include implementing the proposed pipeline on other biometric similarity detection problems where inter-class difference is not significantly small. Another interesting direction could be adopting transfer learning approach and fine-tuning a pre-trained model on the same dataset. Additionally, the current dataset pipeline and SPT can be transferred to a more powerful architecture such as BERT and BART for better performance. In addition to the current research, the proposed pipeline can be implemented on other popular fingerprint benchmarks for comparison and analysis. I strongly look forward to submitting and publishing this research in its refined form to a renowned conference/journal and enable it to serve as a potential direction for research in biometrics in the future.

Bibliography

- [1] Charu C Aggarwal et al. Neural networks and deep learning. *Springer*, 10:978–3, 2018.
- [2] Adhwa Alrashidi, Ashwaq Alotaibi, Muhammad Hussain, Helala AlShehri, Hatim A AboAlSamh, and George Bebis. Cross-sensor fingerprint matching using siamese network and adversarial learning. *Sensors*, 21(11):3657, 2021.
- [3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6836–6846, 2021.
- [4] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1371–1380, 2020.
- [5] Asker M Bazen and Sabih H Gerez. Systematic methods for the computation of the directional fields and singular points of fingerprints. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):905–919, 2002.
- [6] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.
- [7] Anurag Chowdhury, Simon Kirchgasser, Andreas Uhl, and Arun Ross. Can a cnn automatically learn the significance of minutiae points for fingerprint

- matching? In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 351–359, 2020.
- [8] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.
- [9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [10] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [12] Mustansar Fiaz, Arif Mahmood, and Soon Ki Jung. Deep siamese networks toward robust visual tracking. In *Visual Object Tracking with Deep Neural Networks*. IntechOpen, 2019.
- [13] Claudio Gentile and Manfred KK Warmuth. Linear hinge loss and average margin. *Advances in neural information processing systems*, 11:225–231, 1998.
- [14] Evgin Goceri and A Gooya. On the importance of batch size for deep learning. In *Int Conf on Mathematics (ICOMATH2018), An Istanbul Meeting for World Mathematicians, Istanbul, Turkey*, 2018.
- [15] Hao Guo. A hidden markov model fingerprint matching approach. In *2005 International Conference on Machine Learning and Cybernetics*, volume 8, pages 5055–5059. IEEE, 2005.

- [16] Anfeng He, Chong Luo, Xinmei Tian, and Wenjun Zeng. Towards a better match in siamese network based visual object tracker. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.
- [17] Yu-Lin He, Xiao-Liang Zhang, Wei Ao, and Joshua Zhexue Huang. Determining the optimal temperature parameter for softmax function in reinforcement learning. *Applied Soft Computing*, 70:80–85, 2018.
- [18] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [19] Zhe Hui Hoo, Jane Candlish, and Dawn Teare. What is an roc curve?, 2017.
- [20] Shota Horiguchi, Daiki Ikami, and Kiyoharu Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *IEEE transactions on pattern analysis and machine intelligence*, 42(5):1279–1285, 2019.
- [21] AA Ilugbusi and OA Adetunmbi. Development of a multi-instance fingerprint based authentication system. In *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pages 1–9. IEEE, 2017.
- [22] Anil Jain, Arun Ross, and Salil Prabhakar. Fingerprint matching using minutiae and texture features. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 3, pages 282–285. IEEE, 2001.
- [23] Anil K Jain, Yi Chen, and Meltem Demirkus. Pores and ridges: High-resolution fingerprint matching using level 3 features. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):15–27, 2006.
- [24] Dan Jiang and Jin He. Tree framework with bert word embedding for the recognition of chinese implicit discourse relations. *IEEE Access*, 8:162004–162011, 2020.

- [25] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
- [26] Deepika Koundal and Savita Gupta. *Advances in Computational Techniques for Biomedical Image Analysis: Methods and Applications*. Academic Press, 2020.
- [27] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492*, 2021.
- [28] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7083–7093, 2019.
- [29] Yonghong Liu, Baicun Zhou, Congying Han, Tiande Guo, and Jin Qin. A novel method based on deep learning for aligned fingerprints matching. *Applied Intelligence*, 50(2):397–416, 2020.
- [30] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019.
- [31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [32] Simon Marechal. Advances in password cracking. *Journal in computer virology*, 4(1):73–81, 2008.
- [33] Václav Matyáš and Zdeněk Říha. Biometric authentication—security and usability. In *Advanced communications and multimedia security*, pages 227–239. Springer, 2002.

- [34] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Siamese network features for image matching. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 378–383. IEEE, 2016.
- [35] Salil Prabhakar, Anil K Jain, Jianguo Wang, Sharath Pankanti, and Ruud Bolle. Minutia verification and classification for fingerprint matching. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 25–29. IEEE, 2000.
- [36] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. Tunability: importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019.
- [37] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.
- [38] Xuejun Tan and Bir Bhanu. Fingerprint matching by genetic algorithms. *Pattern Recognition*, 39(3):465–477, 2006.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [40] Jucheng Yang, Shanjuan Xie, Sook Yoon, Dongsun Park, Zhijun Fang, and Shouyuan Yang. Fingerprint matching based on extreme learning machine. *Neural Computing and Applications*, 22(3):435–445, 2013.
- [41] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2998–3008, 2021.