CHAPTER 6

FINGER VEIN RECOGNITION USING MOTION-TOLERANT VGG16 AND LSTM ARCHITECTURES

6.1 INTRODUCTION

Minor finger movements or lighting conditions can blur the vein images during scanning, affecting recognition accuracy. A combination of VGG16 architecture (Xie C. & Kumar A., 2017) and Long Short-Term Memory (LSTM) networks (Qin H. & Wang P., 2019) are used in this work to resolve the issue of motion blurs. VGG16 is used for feature extraction and LSTMs are used for learning temporal dependencies. Hence, the system becomes more resilient to lighting fluctuations and minor finger misalignments during scanning. This hybrid architecture enhances recognition accuracy even when the finger is not held perfectly steady, as also supported by the findings of Kuzu R.S. et al. 2020.

6.2 MOTION TOLERANT FINGER VEIN RECOGNITION FRAMEWORK

The VGG16 model, which has been proven to perform better for finger vein recognition tasks, is used in this work. VGG16 architecture is modified to adapt to the motion artifacts in the captured image. To improve the learning ability of the model, the images in the dataset are labelled to eliminate details not relevant to the vein pattern. The dataset size and diversity have been increased through augmentation techniques. Figure 6.1 shows the Motion Tolerant FVR system.

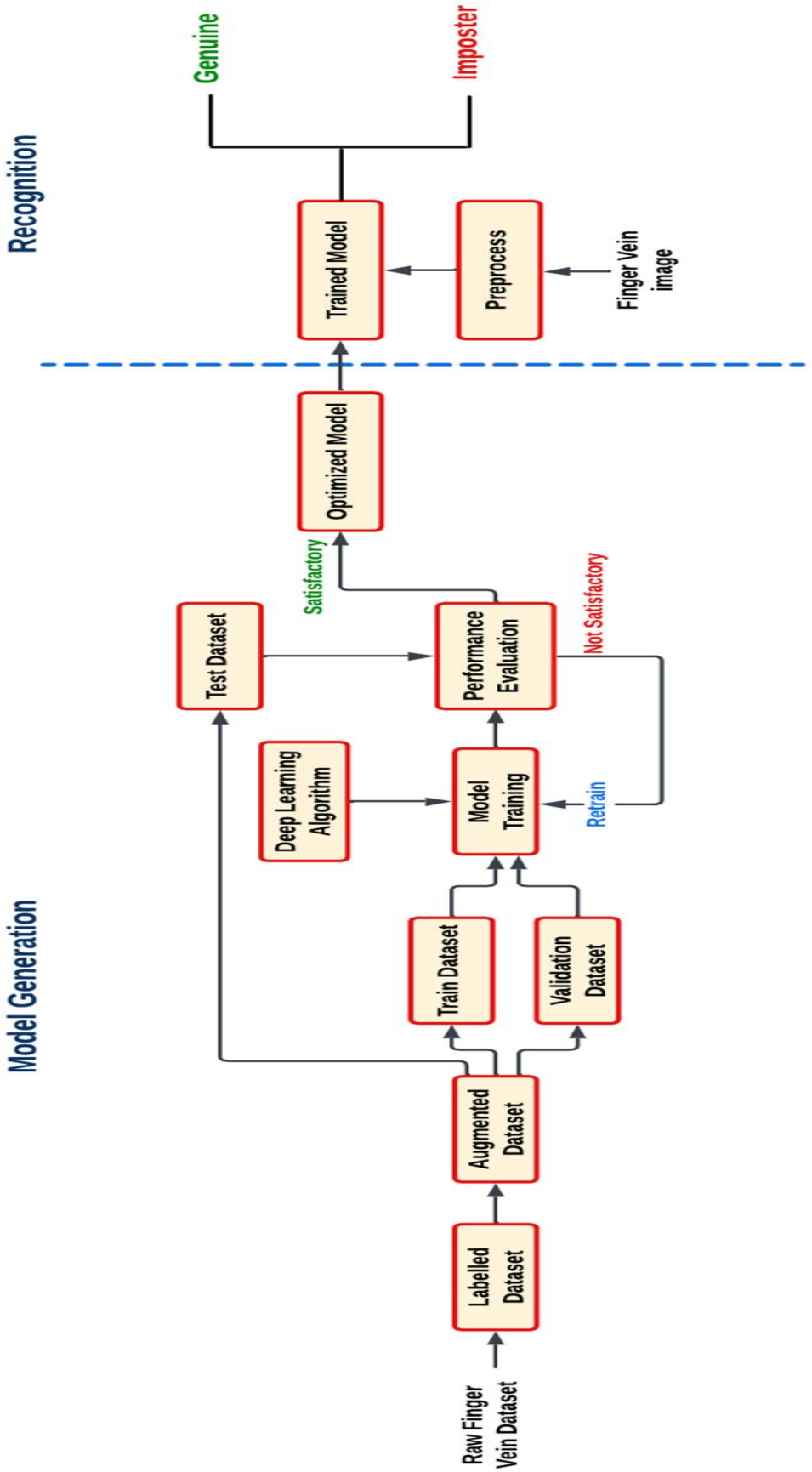

Figure 6.1 Motion Tolerant Finger Vein Recognition System

The raw finger vein images collected are separated into training and test images. The training images are labelled using a hybrid labelling algorithm. Augmentation techniques are then applied to the labelled images. The augmented images are then partitioned into training and validation sets for model development and performance evaluation.

GAN has generated images mimicking motion blur. These images are organized into sequences, where each sequence represents a series of images from the same person. Augmented images contain images that simulate different finger positions, lighting conditions, slight movement etc. Since the model is trained using these varieties of images, it learns to handle these situations in practical applications.

Motion Tolerant DL algorithm is used to train the model. The algorithm is provided with data from the training set to learn and identify underlying patterns during training. The validation set is used to adjust and optimize the parameters of the model. The effectiveness of the model is assessed using a test dataset. The model needs to be retrained with adjustments to improve the performance if the performance is not satisfactory. When the model achieves satisfactory performance with the test dataset, it can then be used for authentication. The ROI extracted from the captured FV image is fed to the trained model, and it verifies if the new image matches with an enrolled one or identifies it as an imposter. This approach ensures that the model is thoroughly trained, validated, and optimized before being deployed for actual recognition tasks.

The model is named “Motion Tolerant” model as it can handle variations in finger position during capture. This architecture uses a VGG16 for feature extraction, followed by a time-distributed layer. An LSTM layer is then used to capture temporal dependencies and handle motion artifacts. Finally, a dense layer is used for classification.

6.3 LONG SHORT-TERM MEMORY NETWORK

LSTM networks process sequential data and capture long-term dependencies within it. The LSTM uses a memory cell that can maintain information over time. This allows the network to capture and remember information over sequences. LSTMs use a repeating module that contains interacting layers, which work together to maintain and update the memory cell. It contains two memory states, the Hidden state and the Cell

state. The input, forget, and output gates regulate the flow of information by determining what to store or discard memory cell at each time step.

The variables used in the equations 6.1 to 6. 5 are:

t: Current time step in the sequence.

t−1: Previous time step.

X_t: New Input

h_{t-1}: Preceding Hidden State.

W_f, W_i, W_o: Weight.

b_f, b_i, bo : Bias.

Two memory states are maintained by LSTM: the Hidden state and the Cell state.

Hidden State (Short-Term Memory): This represents the current output of the network and carries information from the previous time step that is immediately required. The current hidden state is calculated using the equation 6.1:

$$H_t = O_t * \tanh(C_t) \quad (6.1)$$

Cell State (Long-Term Memory): This retains relevant information across many time steps and allows the network to understand long-term dependencies in the data. The current cell state is calculated using the equation 6.2:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (6.2)$$

LSTM uses the following gates to control the flow of information within the cell.

Forget Gate: This gate controls the amount of information from the previous cell state that should be eliminated. The forget gate output is represented using the equation 6.3.

$$f_t = sigmoid(W_f * [h_t - 1, x_t] + b_f) \quad (6.3)$$

Here f_t is the forget gate output at time step t. It decides how much of the previous cell state is to be retained.

Input Gate: This gate controls the amount of new information to be added to the cell state, which can be represented using the equation 6.4:

$$i_t = Sigmoid\ (W_i\ [h_{t-1}, x_t] + b_i) \quad (6.4)$$

Output Gate: This gate determines how much of the updated cell state influences the hidden state, which in turn serves as the output of the current LSTM unit. The hidden state carries time-step-specific information and is either passed to the next LSTM cell or used directly for prediction. The output of this gate is represented using the equation 6.5:

$$O_t = \text{Sigmoid}\ (W_o\ [h_{t-1}, x_t] + b_o) \quad (6.5)$$

The original dataset is labelled using the hybrid algorithm. The dataset is then enhanced using conventional transformations and using GANs. The augmented dataset is then fed into a Motion Tolerant Deep Learning Model. This model is specifically designed to handle the challenges posed by motion artifacts, which can occur when the finger is not perfectly still during the image-capturing process. This approach improves the performance of the authentication system under different real-world situations. The new FV image to be verified can be fed to the trained model, which then recognizes it as either a genuine user or an impostor user.

6.4 ARCHITECTURE OF THE MOTION TOLERANT DEEP LEARNING MODEL

The Motion-Tolerant model integrates the strengths of VGG16 and LSTM networks for feature extraction and sequential data processing. From the original VGG16 architecture, the FCLs are removed. The enormous number of parameters in FCL increases the complexity of the model. By removing them, the model becomes lighter and more computationally efficient. Instead, task-specific TD layer and LSTM layer are introduced in the new architecture. This helps in handling temporal dependencies and motion resilience. Figure 6.2 illustrates the architecture of the Motion Tolerant model.

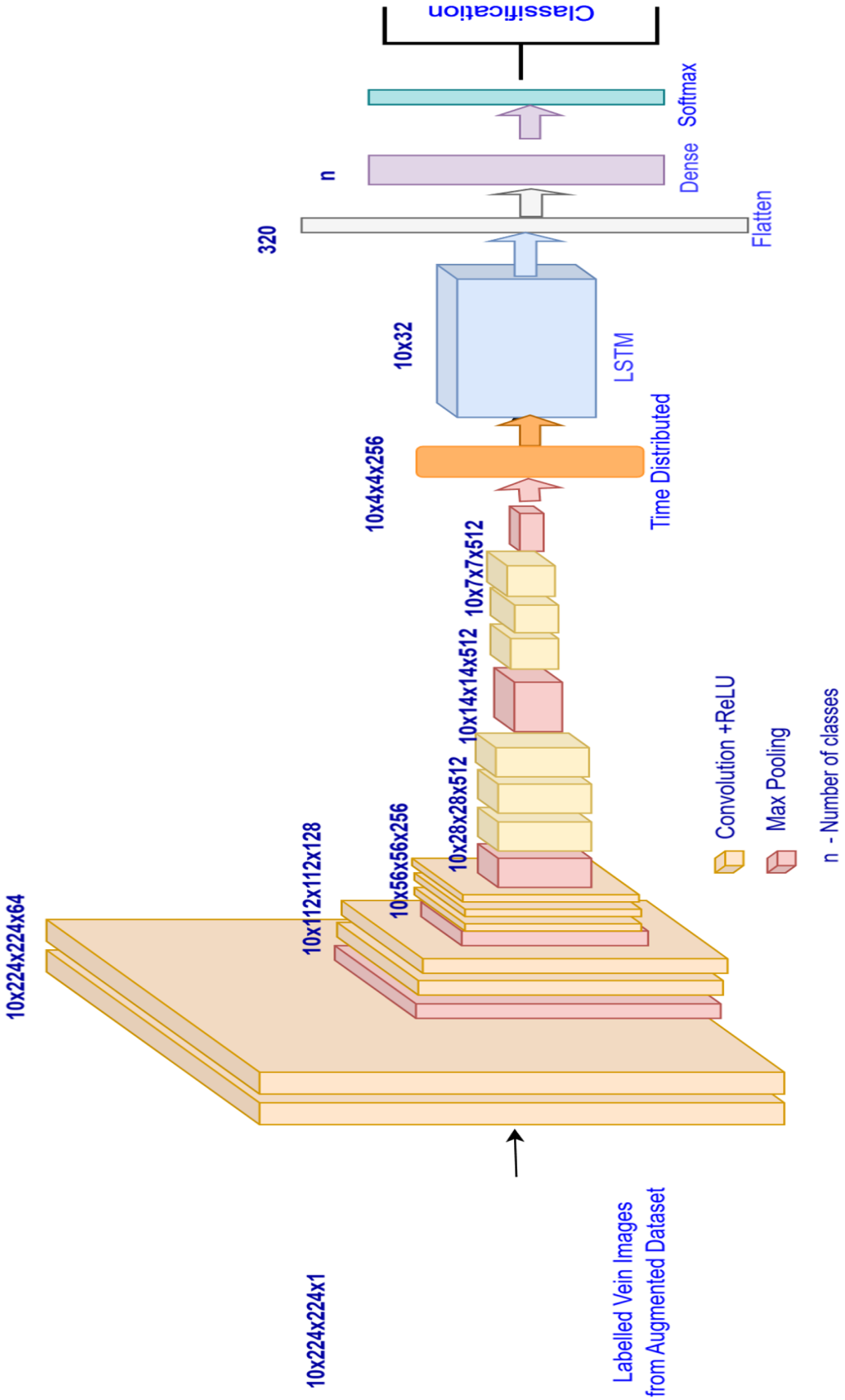

Figure 6.2 Architecture of the Motion Tolerant Finger Vein Recognition Model

The fixed-size vectors generated by the convolutional layers are processed by the Time Distributed (TD) layer. This layer treats each frame of the sequence independently. After the TD layer, an LSTM layer with 32 hidden units is added. This layer extracts the temporal dependencies between the feature vectors generated by the preceding layers. It learns the sequential patterns and associations present in the data by processing the sequence of feature vectors. This helps in mitigating motion blurs and other capture conditions.

The output is then passed through a flattened layer, which converts 2D data into a 1D vector. This 1D vector is then fed into a dense layer consisting of neurons equivalent to the number of classes. The probability distribution for the classes is predicted using a softmax activation function. The input image is recognized as the class with the maximum probability. The steps are described in the Algorithm 6.1.

Algorithm 6.1: Motion Tolerant Deep Learning Model

1. Data Preprocessing:

Group the input images into sequences, Sk

For each individual j: Sk= {Ik,1, Ik, 2…, Ik,T }

T: count of images in the sequence

Ik,t : t^{th} image in the sequence of the kth individual

2. Feature extraction:

Initialize the VGG16 model.

Remove the FCLs from the initialized VGG16 model.

for each image Ik,t in sequence Sk:

Fk,t=VGG16_Conv (Ik,t)

3. Incorporate temporal information:

Apply a time distributed layer with flattening to each feature vector:

Fk,t′=flatten (Fk,t)

4. Learn temporal relationships using LSTM:

Initialize the LSTM layer with 32 hidden units:

for each sequence Sk with flattened feature vectors {Fk1', Fk2′…, FkT}:

Hk = LSTM ([Fk1′, Fk2′…, FkT′])

5. Classification:

Initialize a FCL with c neurons and a softmax activation function. (c: number of classes)

for each hidden state sequence Hj, classify the sequence: yj=Softmax (WHj + b)

6. Model training and evaluation:

Compile the model

Loss=categorical cross-entropy

Optimizer=Adam

Train the model using the labelled and augmented dataset:

mdl.fit(sequences, labels, split =0.2, epochs = epochs)

Evaluate the performance on the validation set: mdl.evaluate (validation_set)

Table 6.1. shows the different hyperparameters used in the Motion Tolerant Model.

Table 6.1 Hyperparameter Tuning for the Motion Tolerant Model

Parameters	**Values**
Input Size	224x224x10
Activation Function	ReLU and SoftMax
Loss function	Categorical Cross Entropy
Optimizer	Adam Optimizer
Learning Rate	1e-4
Batch size	16
LSTM Units	128
LSTM Layers	1
Recurrent Dropout rate	0.2

6.5 RESULTS AND DISCUSSIONS

The performance of the Motion Tolerant DL model for FV authentication is evaluated using the THUFV dataset and the SDUMLA dataset.

6.5.1 Performance Evaluation of Motion Tolerant Model

a. THUFV Dataset

Figure 6.3 illustrates the effectiveness of a Motion Tolerant DL model during training and validation.

In the graph given in Figure 6.3, both training and validation performance improved significantly during the early epochs, with both metrics converging above 99%. This implies that the model is learning well from the data and is not underfitting.

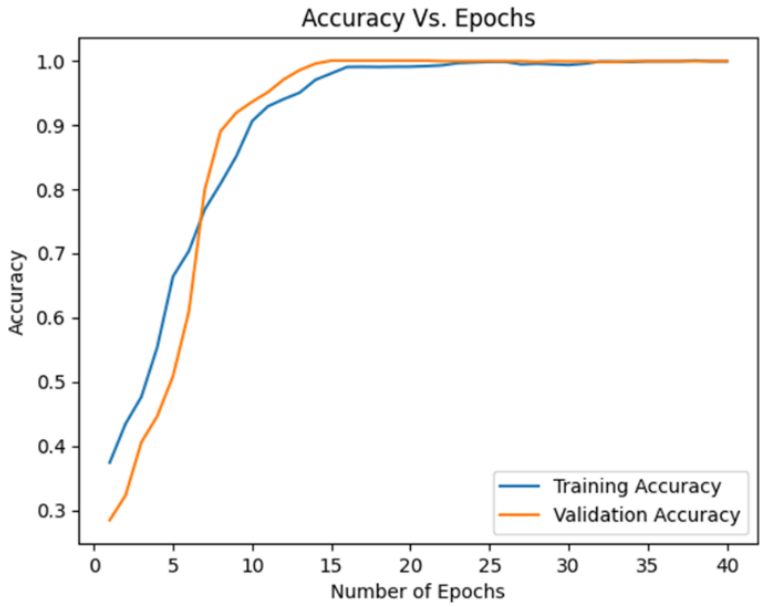

Figure 6.3 Accuracy curve of Motion Tolerant Model on Labelled-Augmented THUFV Dataset

Figure 6.4 shows that both losses decrease rapidly in the initial epochs and stabilize at minimal values, demonstrating that the model is minimizing error consistently. The validation curves also closely follow the training curves, suggesting good generalization, meaning the model is learning patterns that generalize well without overfitting. Overall, the model appears to be well-tuned with appropriate complexity and learning rates, leading to a successful training process.

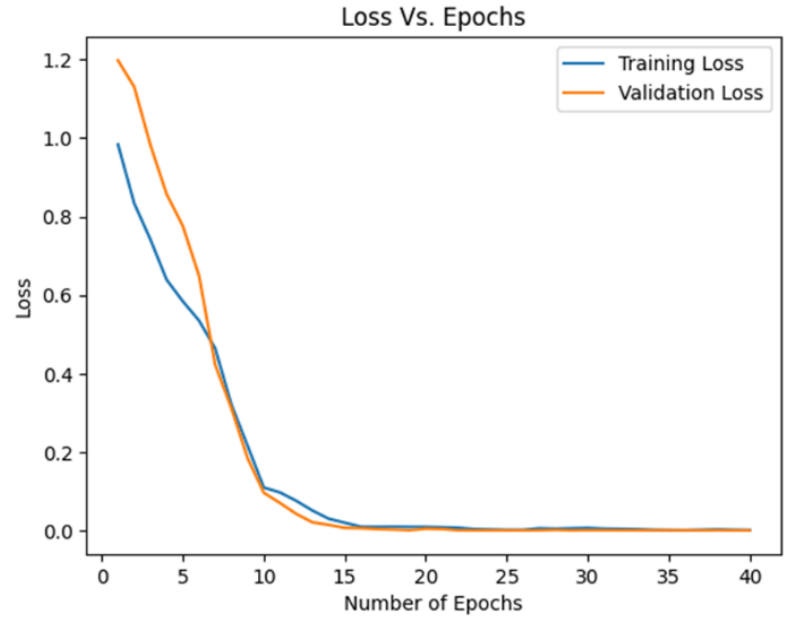

Figure 6.4 Loss Curve of Motion Tolerant Model on Labelled-Augmented THUFV Dataset

Table 6.2 compares the performance of different configurations across various measures, showing that the motion-tolerant model with labelling and augmentation consistently outperforms the other configurations.

Table 6.2 Performance Comparison of Different Model Configurations using THUFV Dataset

Configuration	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16	92.5	94.09	97.64	95.83
L-VGG16	96.34	98.29	97.87	98.08
LA-VGG16	98.6	99.6	98.97	99.29
LA-Motion Tolerant Model	**99.89**	**99.93**	**99.96**	**99.94**

Table 6.2 depicts the improvement in performance metrics of the THUFV dataset when different techniques are applied to the FV authentication system. The baseline VGG16 model obtained accuracy, precision, recall, and F1-score values of 92.5%, 94.09%, 97.64%, and 95.83%, respectively. The addition of labelling improves accuracy

to 96.34%, representing a 3.84% increase. Incorporating both labelling and dataset augmentation further enhances the model, achieving an accuracy of 98.6%, with 6.26% improvement over the baseline. The Motion Tolerant DL model, which includes labelling and augmentation, reaches an impressive accuracy of 99.89%, showing a substantial 7.39 % increase from the baseline. Improvements can be observed in Precision, recall, and F1-score also, with precision increasing to 99.93% (+5.84%), recall to 99.96% (+2.32%), and F1-score to 99.94% (+4.11%) compared to the baseline. The significant improvement in all metrics shows the effectiveness of combining labelling and data augmentation techniques with the Motion Tolerant Model.

b. SDUMLA Dataset

Figure 6.5 displays the training and validation accuracy graph for the SDUMLA-HMT dataset. The accuracy graph shows that both training and validation accuracy rapidly increase and then stabilizes, indicating effective model learning with minimal overfitting.

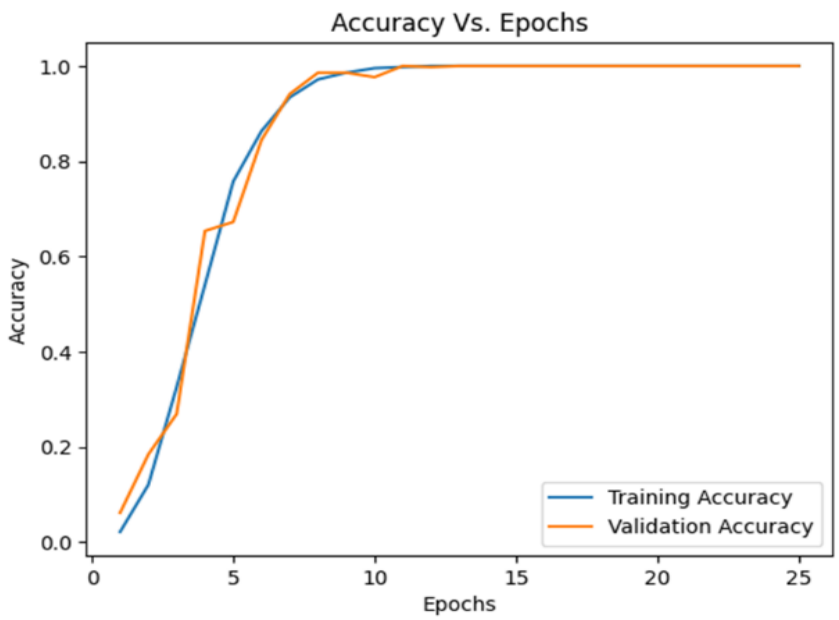

Figure 6.5 Accuracy Curve of Motion Tolerant Model on Labelled-Augmented SDUMLA Dataset

Figure 6.6 shows that both training and validation loss decrease steadily, further suggesting that the model is converging well without significant overfitting, as the losses for both sets decrease at similar rates and remain low by the end of training.

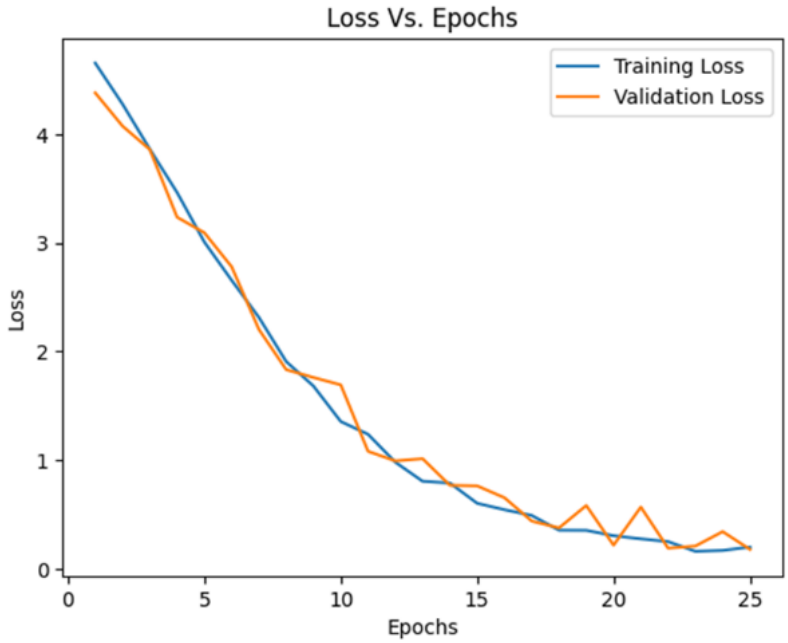

Figure 6.6 Loss Curve of Motion Tolerant Model on Labelled-Augmented SDUMLA Dataset

Table 6.3 compares the performance of the Motion Tolerant Model with the VGG16, L-VGG166 and LA-VGG16 configurations.

Table 6.3 Performance Comparison of Different Model Configurations using SDUMLA- HMT Dataset

Configuration	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
VGG16	94.31	93.60	92.07	92.83
L-VGG16	95.45	94.78	96.76	95.76
LA-VGG16	97.10	95.73	97.11	96.41
LA-Motion Tolerant Model	**99.76**	**97.42**	**99.76**	**98.58**

Starting with the base VGG16 model, the accuracy is 94.31%, which increases by 1.14% to 95.45% when labelling is added. This improvement continues as data augmentation is introduced, raising accuracy by an additional 1.65% to 97.10%.

Significant improvement can be observed with the model, which combines labelling and augmentation, achieving an accuracy of 99.76%, a total increase of 5.45% from the base model. Similarly, precision increases by 3.82%, recall by 7.69% and F1-Score by 5.75%. These results show that the motion-tolerant model with labelling and augmentation consistently outperforms the other configurations across various metrics for the SDUMLA-HMT dataset.

6.5.2 Analysis of ROC, AUC and EER

The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings.

$$\text{True Positive Rate} = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (6.6)$$

$$\text{False Positive Rate (FPR)} = \frac{False\ Positive}{False\ Positive + True\ Negative} \quad (6.7)$$

The area under the ROC curve is measured as AUC (Area Under the Curve). AUC value can range between 0 and 1. A value of 1 signifies a perfect model, and 0.5 indicates a model without discrimination capability.

$$AUC = \int_0^1 TPR(FPR)d(FPR) \quad (6.8)$$

The Equal Error Rate (EER) is the point on the ROC curve where the False Acceptance Rate (FAR) equals the False Rejection Rate (FRR). ERR is a critical metric in biometric and security systems.

$$\text{False Acceptance Rate (FAR)} = \frac{False\ Positive}{False\ Positive + True\ Negative} \quad (6.9)$$

$$\text{False Rejection Rate (FRR)} = \frac{False\ Negative}{False\ Negative + True\ Positive} \quad (6.10)$$

The ROC curve in Figure 6.7 compares the performance of the Motion Tolerant model on THUFV (blue curve) and SDUMLA (orange curve) datasets.

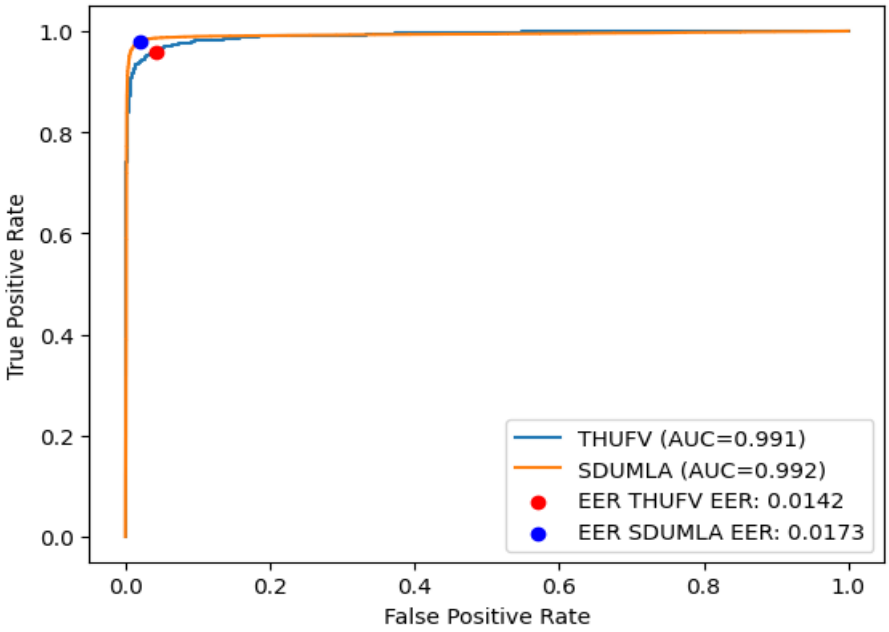

Figure 6.7 ROC curves of THUFV and SDUMLA Datasets for the Motion Tolerant Model

The EER and AUC values for the THUFV and the SDUMLA datasets are shown in Table 6.4.

Table 6.4 Details of EER and AUC Values

Dataset	EER (%)	AUC
SDUMLA- HMT	1.73	0.993
THUFV	1.42	0.991

The value of AUC signifies that the model performs well on both datasets, with THUFV having an AUC of 0.991 and SDUMLA slightly higher at 0.993. The EER is 1.42% and 1.73% for the THUFV and the SDUMLA datasets, respectively. These metrics suggest that the classifier has very low error rates on both datasets. The EER values obtained for various methods on the SDUMLA dataset are compared in Table 6.5.

Table 6.5 Comparison of EER for Different Algorithms

Model	WLD	Gabor + LBP	RLT	LMC	CNN	Resnet50	Resnet 101	CNN-CO	SegNet	**Motion Tolerant**
EER (%)	22.7	8.096	5.46	4.54	3.906	3.493	3.365	2.37	2.17	**1.73**

The results in Table 6.5 shows that Motion Tolerant model achieves the lowest EER over various advanced algorithms.

6.6 SUMMARY

The Motion-Tolerant Finger Vein Recognition architecture integrates VGG16 for robust feature extraction and LSTM networks for effective sequence learning. The new model was evaluated using the THUFV and SDUMLA-HMT datasets. The accuracy obtained for the THUFV dataset and SDUMLA dataset are 99.89% and 99.76%, respectively. High accuracy values, along with high AUC values and low EER values for both datasets, also prove the model's effectiveness in recognizing FV patterns under varying conditions in real-world applications.