CHAPTER 3

FINGER VEIN RECOGNITION USING MODIFIED UNET AND VGG16 MODELS

3.1 INTRODUCTION

Traditional FVR systems extract features manually and then machine learning techniques are used to classify these features. However, these methods faced limitations under varying environmental conditions like changes in lighting, finger positioning, and image quality. These challenges necessitate the need for advanced techniques to develop more reliable FVR systems.

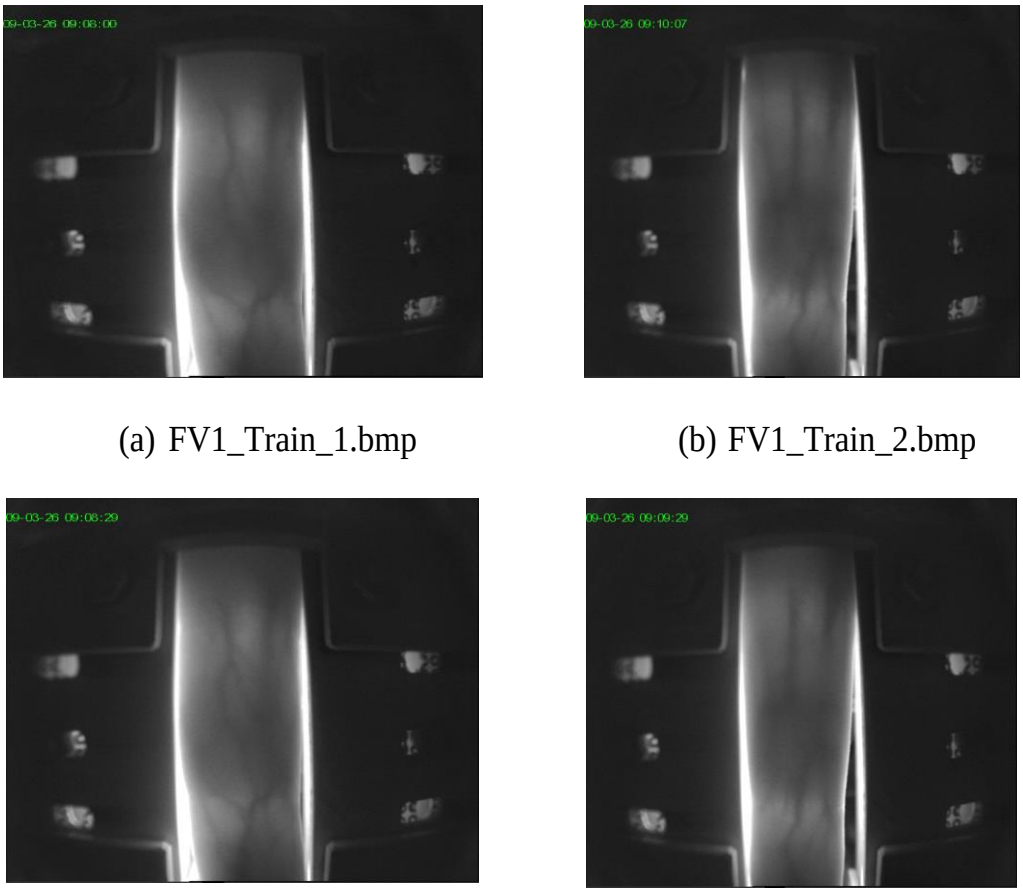
Deep Learning algorithms can be used for FV recognition tasks, which automatically extract and classify patterns from raw FV images. CNNs can be effectively used for image recognition applications as they can learn complicated patterns from large datasets (Hijazi S. et. al. 2015).

Two methods are experimented with in this work. The first method uses a modified U-Net architecture for FV image recognition. The U-Net, which was introduced for biomedical image segmentation, can extract detailed and fine structures in the images. Hence, it is selected for the FV recognition task. (Ronneberger O. et. al. 2015, Huang. H. et al. 2020). Since UNET can only extract features from, some modifications are done on UNET architecture to make it suitable for the classification task. The second method uses VGG16(Visual Geometric Group), a well-known deep CNN architecture, which is suitable for FV authentication due to its deep feature extraction capabilities, which allow it to capture fine vein patterns effectively.

3.2 DATASET DESCRIPTION

Two different datasets are used for the experiments to analyze the performance of the models across varied data environments. Each dataset was evaluated independently to ensure that the performance metrics accurately reflect the model's ability to manage distinct features of each dataset.

The model architecture was first constructed using the "Tsinghua University FV and Finger Dorsal Texture Database (THU-FVFDT)". The dataset is available at "https://www.sigs.tsinghua.edu.cn/labs/vipl/thu-fvfdt.html"(Yang W. et al., 2012). This dataset consists of raw FV images from 220 individuals. Two images are acquired from each individual during two separate sessions. The dataset contains images with nonuniform illumination. This dataset has been widely used in prior research on FV recognition. These images were collected from students and staff at Tsinghua University Graduate School, and with a size of 720x576. The samples of images are shown in Figure 3.1.



(a) FV1_Train_1.bmp (b) FV1_Train_2.bmp

(c) FV1_Test_1.bmp (d) FV1_Test_2.bmp

Figure 3.1. Sample Images from the THU-FV Dataset

The second dataset, the Shandong University Homologous Multi-modal Traits Database (SDUMLA-HMT), compiled by the Machine Learning and Applications team at Shandong University, is available at "https://time.sdu.edu.cn/kycg/gksjk.htm" (Yilong

Yin. et al., 2011) This includes a larger number of images and was used to further verify the model's performance on a more extensive database. The samples of images from the dataset are shown in Figure 3.2.

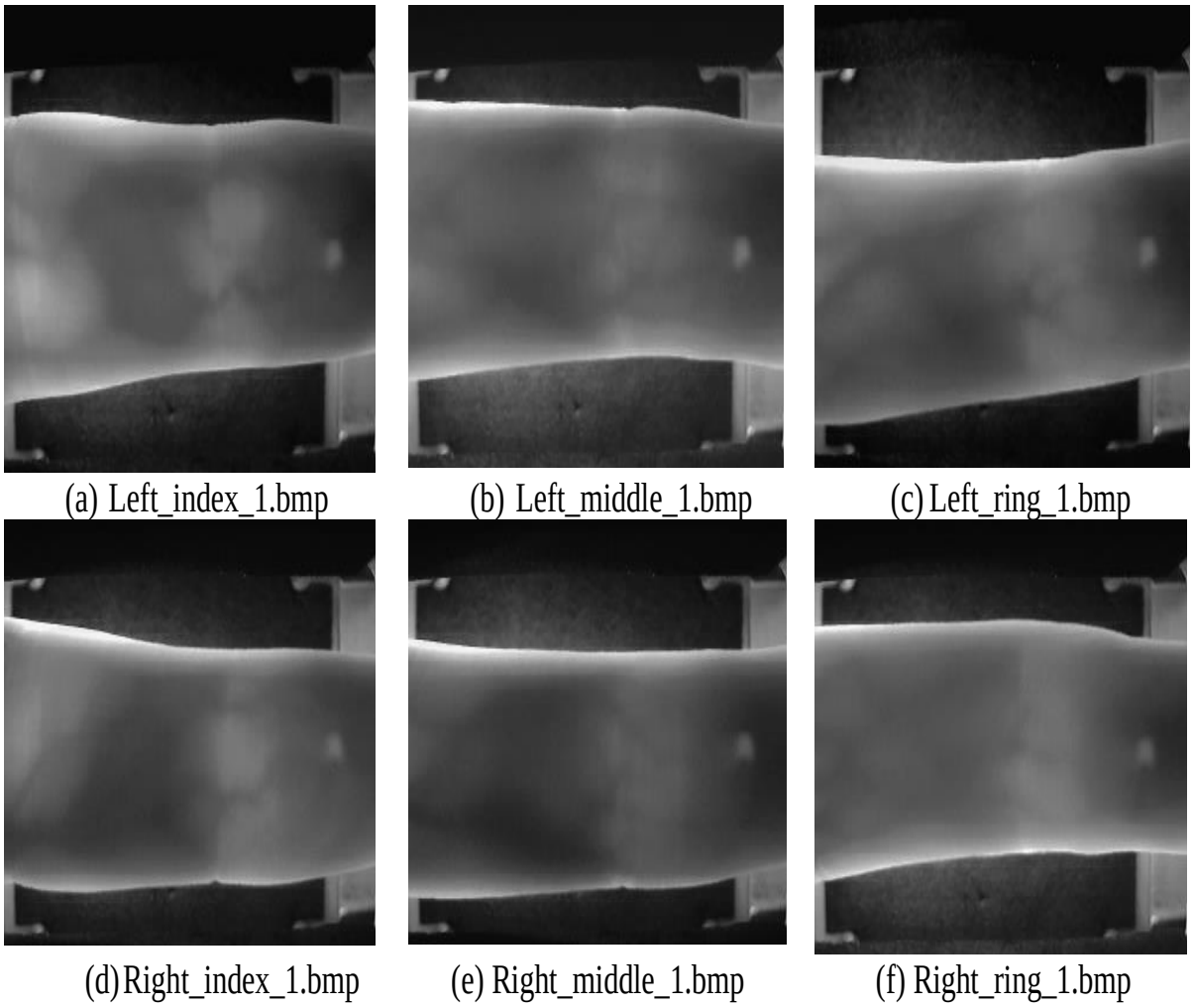


Figure 3.2 Sample Images from the SDUMLA-HMT Dataset

This dataset includes FV samples collected from 106 individuals. The vein images of the ring, middle, and index fingers from the right and left hands were captured from each person. Six images per finger are captured during dataset collection. As a result, 3,816 images were captured, each with dimensions of 320x240 pixels, and all images were provided in BMP format. An important aspect of the image-capturing process is that it was conducted without using a guide bar. The absence of a guide bar means the images are susceptible to alignment issues and inconsistent shading. These factors contribute to the classification of the SDUMLA-HMT as a low-quality FV database, presenting additional

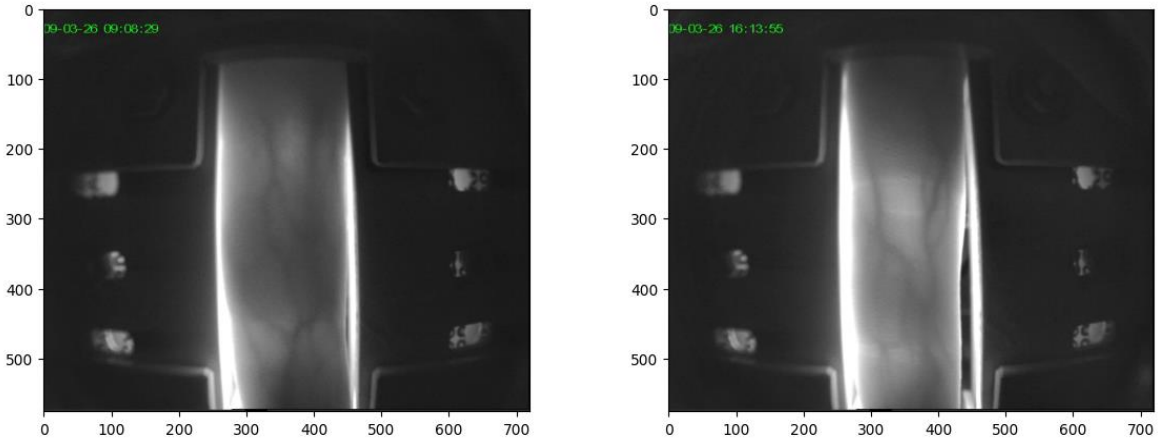
challenges for image analysis and recognition tasks. The details of the datasets are shown in Table 3.1.

Table 3.1 Dataset Description

DETAILS	DATASET	
	SDUMLA	**THUFV**
Number of images per individual	6*6	2
No. of individuals	106	220
Size of image	320*240	720*526
Total images	3816	440
Remarks	Images of the 6 fingers of 106 individuals captured 6 times	Images of 220 individuals captured in 2 sessions

3.3 PREPROCESSING

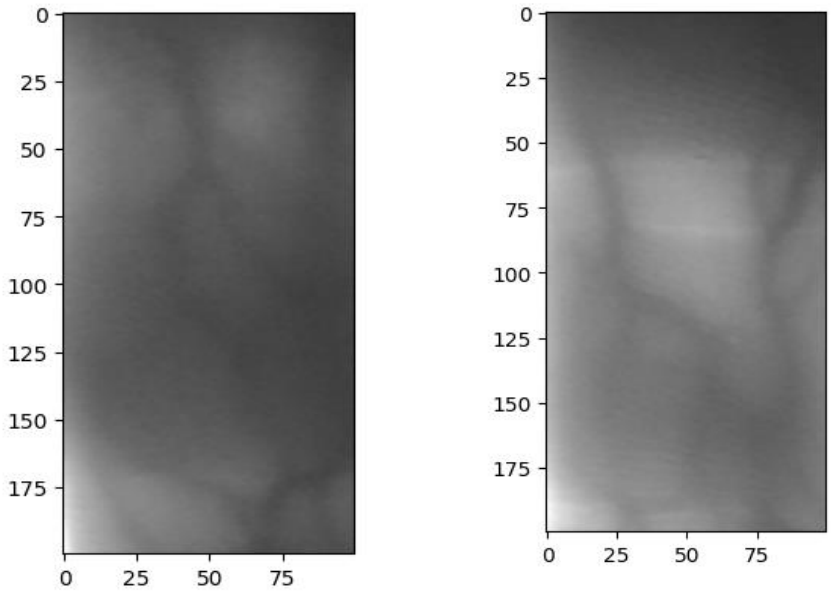
Preprocessing the images before feeding the deep learning models ensures consistency in size, scale, and quality of the image, thereby increasing the accuracy and computational efficiency of the model. The different stages in preprocessing include ROI extraction, rotation, resizing and contrast enhancement. The sample images from the THUFV dataset before preprocessing are shown in Figure 3.3.



(a) THUFV1_Train_1_1.bmp (b) THUFV1_Train_7_1.bmp

Figure 3.3 Two Sample Images from THUFV Dataset before Preprocessing

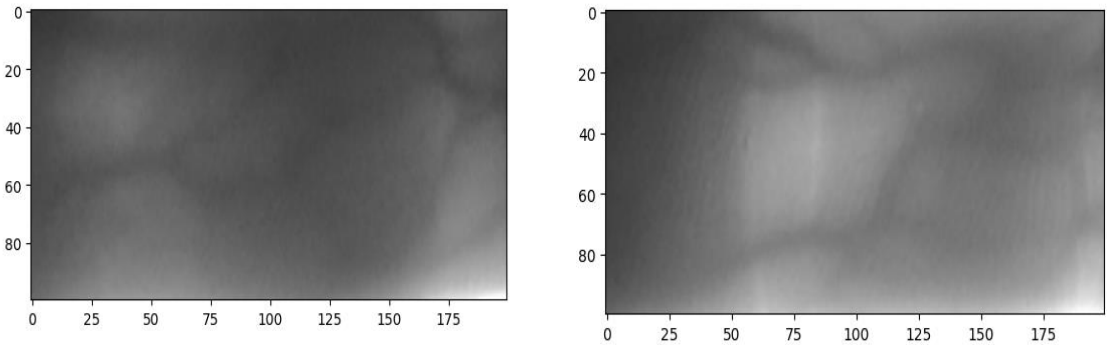
The Region of Interest (ROI) is first extracted by isolating the specific area of the image that contains the FV patterns. This helps to eliminate background noise and irrelevant data, which helps the model to focus on relevant data. This is done by cropping the image. Depending on the raw images in a specific dataset, the images are cropped to extract FV regions. Figure 3.4 represents the ROI extracted images of the samples shown in Figure 3.3.



(a) THUFV1_Train_1_1.bmp (b) THUFV1_Train_7_1.bmp

Figure 3.4 ROI Extracted Images

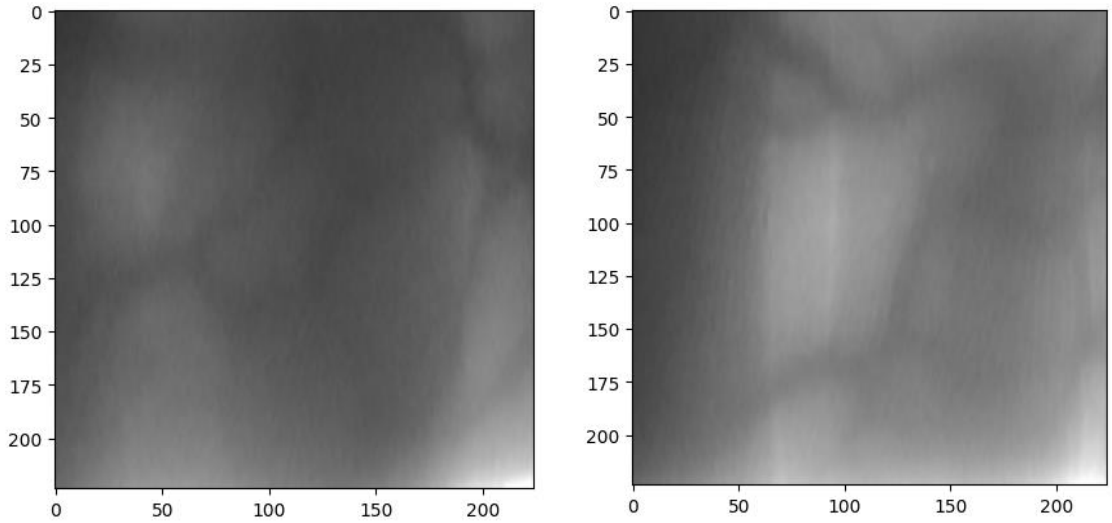
The ROI is then rotated 90 degrees to standardize its orientation, whereas for the SDUMLA dataset, this step can be avoided. Standardizing the orientation of vein patterns lets the model concentrate on the structural features of the veins. The rotated images are shown in Figure 3.5.



(a) THUFV1_Train_1_1.bmp (b) THUFV1_Train_7_1.bmp

Figure 3.5 Images Rotated by 90^{o}

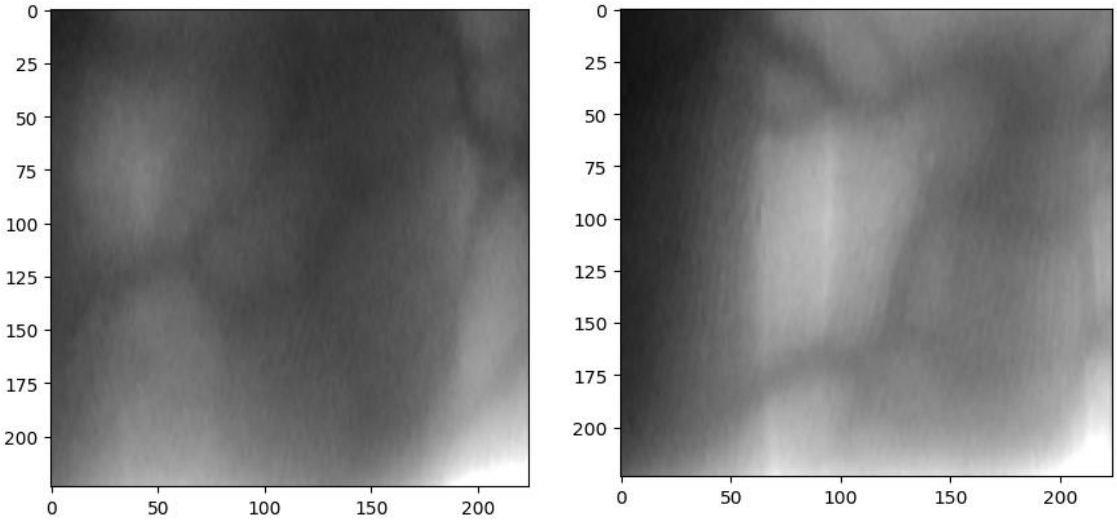
The image is then resized to a uniform dimension of 224x224 pixels. Resizing the image with equal height and width helps to apply convolution and pooling operations more efficiently. Consistent image sizes allow the model to process batches of images efficiently and minimize the risk of errors that can arise when images with varying dimensions are fed into the model. The resized images are displayed in Figure 3.6.



(a) THUFV1_Train_1_1.bmp (b) THUFV1_Train_7_1.bmp

Figure 3.6 Resized Images

To improve the clarity of vein patterns, the contrast is enhanced in the resized images. The images after contrast enhancement are shown in Figure 3.7.



(a) THUFV1_Train_1_1.bmp (b) THUFV1_Train_7_1.bmp

Figure 3.7 Contrast Enhanced Images

These preprocessing steps, ROI extraction, image rotation, resizing and contrast enhancement increase the ability of the model to concentrate on important features like vein patterns, by reducing background noise and irrelevant information. The dataset is split into 70% for training, 15% for validation and 15% for testing.

3.4 UNET ARCHITECTURE

UNET is a robust CNN architecture suggested by Olaf Ronneberger, Philipp Fischer, and Thomas Brox in 2015, originally developed for segmenting biomedical images. Its name, 'UNET', refers to the U-shaped structure of the network. The architecture is composed of an encoder (contracting path) and a decoder (expansive path). The original UNET architecture was primarily developed for image segmentation tasks, where the objective was to delineate specific structures within an image. However, in the context of FVR, there is an additional necessity to classify the segmented vein images into distinct classes for further analysis and identification. To address this, an FCL is incorporated into the UNET architecture to enable classification after the segmentation process. Thus, the UNET component is responsible for segmentation, and FCL is used for classification. The architecture of UNET is given in Figure 3.8.

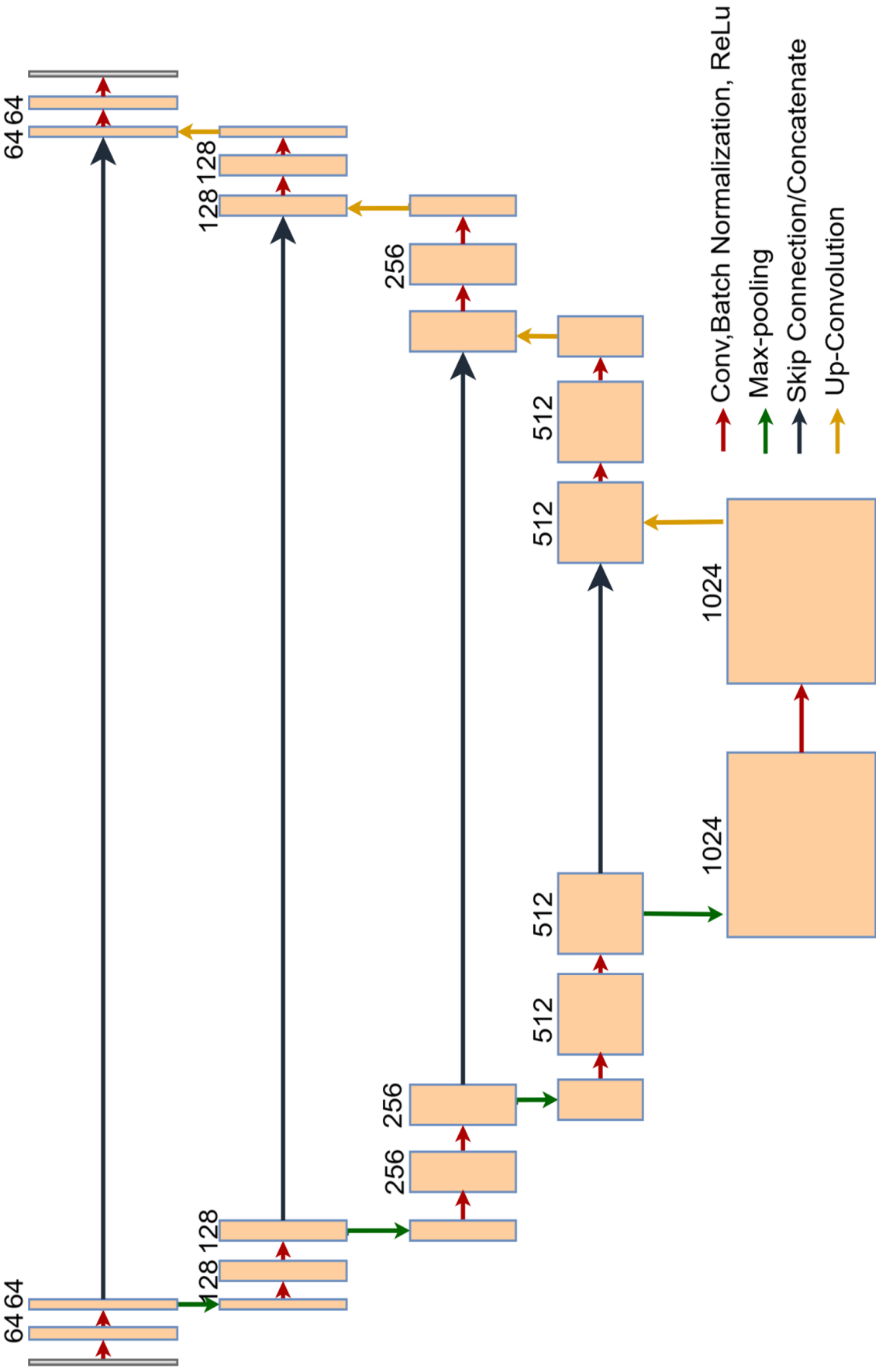


Figure 3.8 Architecture of UNET

The encoder, or contracting path, applies a sequence of convolutional layers to downsample the input image, each followed by ReLU activation functions and batch normalization, which help in learning complex features. Max-pooling layers reduce the dimension of the input image while increasing the depth of the feature maps. High-level features are extracted when the image passes through the encoder.

The decoder, or expansive path, reconstructs the image to its original dimensions through up-convolutions. Skip connections connect each stage of the encoder to the parallel stage in the decoder. This ensures that high-level features from the encoder are preserved and integrated into the decoding process. The use of up-convolutions, together with skip connections, enables the network to restore spatial details that might have been lost during the down-sampling phase.

This ability to segment and reconstruct images has made U-Net highly suitable for tasks requiring precise feature extraction, such as FV pattern recognition. With the addition of FCL for classification, the UNET model can both segment vein images and classify them. With the combination of segmentation and classification capabilities, modified UNET can be used for FV recognition tasks.

3.5 MODIFIED UNET ARCHITECTURE

The UNET architecture is modified specifically for FVR to accommodate both segmentation and identification tasks. The original UNET is designed primarily for image segmentation, where the objective is to generate detailed segmentation maps that accurately delineate objects or regions within an image. However, FVR requires not only the segmentation of vein patterns but also the capability to identify these patterns as belonging to specific individuals, making classification an essential component of the system.

In the Modified UNET one FCL is integrated using a flattened layer to the output of the encoder. This will convert the extracted features into a one-dimensional vector to perform the classification for identifying the individuals. The modified architecture, shown in Figure 3.9, retains the core elements of the original UNET while introducing enhancements to support classification.

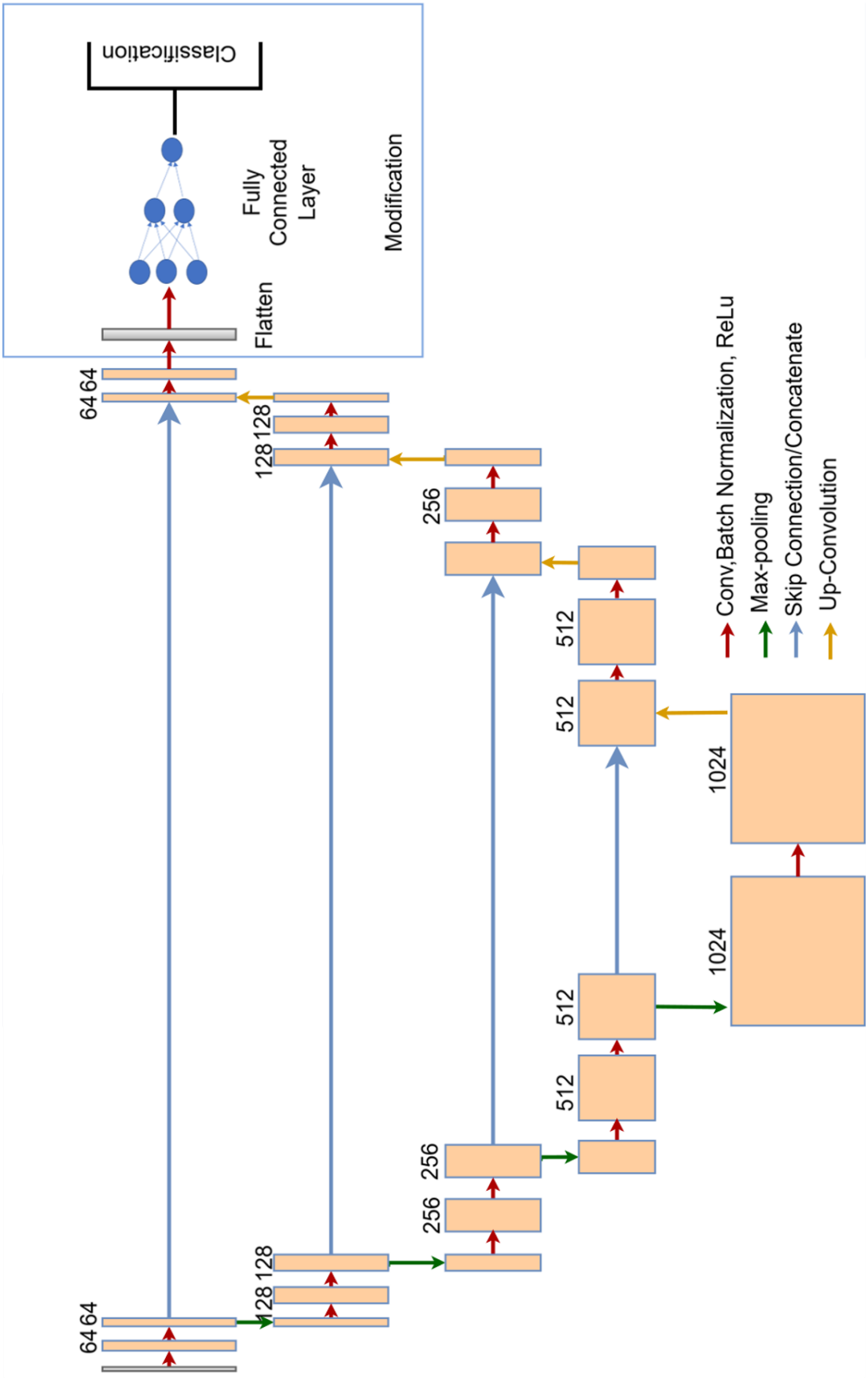


Figure 3.9 Architecture of Modified UNET for Finger Vein Recognition

The encoder captures hierarchical features from the input image by applying convolution, ReLU and max pooling. These layers encode the spatial features of the veins. The decoder then reconstructs high-resolution segmentation maps by up-sampling and refining these encoded features. High-resolution features from the encoder are integrated with the decoder's output through skip connections, ensuring spatial details are preserved during reconstruction.

The flattening operation restructures the feature maps into a linear vector format, while the FCL processes this vector to generate a classification output. This will combine the segmentation and identification tasks, enabling the modified UNET to serve as a finger vein recognition solution.

i. Contracting Path (Encoder)

The contracting path of the network is built using repeated convolutional layers and max pooling layers. It learns both the context and distinguishing features of the input image at different levels of resolution.

- **Convolution Layers**: Convolution layers are used to extract features from the input image. These layers apply convolutional filters to the input, capturing spatial hierarchies.

- **ReLU Activation**: Rectified Linear Unit (ReLU) allows the model to learn intricate and abstract features using its nonlinear behaviour.

- **Batch Normalization**: Standardizing the outputs of convolutional layers enhances training stability and speed.

- **Max Pooling**: By down-sampling, these layers shrink the spatial size of the feature maps, keeping key features intact and decreasing the computational load. Max pooling specifically selects the largest value from every single pooling window, thus providing spatial invariance.

- **Feature Extraction**: When moving down the contracting path, the network captures increasingly complex features of the image at different scales.

ii. Expansive Path (Decoder)

The decoder utilizes the encoded, compressed features to restore the image, aiming to produce a high-resolution segmentation map. A summary of the layer details is as follows.

- **Upsampling Layers:** The spatial size of the feature maps is expanded by these layers. Common techniques include up-convolution (transposed convolution) or bilinear interpolation.
- **Convolutional Layers:** These layers process the up-sampled feature maps, refining the segmentation and reconstructing the spatial dimensions to match the original input size.
- **Skip Connections:** Connections are established between matching layers of the encoder and decoder to preserve spatial information and facilitate the flow of high-resolution features from the encoder to the decoder. This aids in preserving spatial details that may be lost during the down-sampling process.
- **Concatenation Operation:** The concatenation step merges encoder features with the up-sampled outputs from the decoder. This enriches the decoder's feature maps with high-resolution information, improving the segmentation accuracy.

iii. Flatten Layer

By transforming the multi-dimensional feature maps, this layer produces a one-dimensional feature vector, which is essential for feeding the FCL for classification.

iv. Fully connected Layer (FCL)

After segmentation, the FCL operates on the extracted features to classify the segmented patterns. After the U-Net component segments and extracts intricate vein patterns from finger images, the resulting features are flattened and passed through an FCL. It is then passed through softmax layer, which classifies the features by mapping the extracted feature vector to the corresponding output classes.

Algorithm 3.1. Modified UNET Architecture

Step 1. Encoder (Contracting Path)

Step 1.1: Apply a convolutional block (Conv, Batch Normalization, ReLU) and max pooling

X1 = ReLU (BatchNorm (Conv (I, W1)))

X2 = ReLU (BatchNorm (Conv (X1, W2)))

X_pool1 = MaxPool(X2, k=2, s=2)

Step 1.2: Repeat convolutional block and max pooling

X3 = ReLU (BatchNorm (Conv (X_pool1, W3)))

X4 = ReLU (BatchNorm (Conv (X3, W4)))

X_pool2 = MaxPool (X4, k=2, s=2)

Repeat this process for four stages in the encoder

Step 2. Bottom of the U-Net

Step 2.1: Apply convolutional block without pooling

X_b1 = ReLU (BatchNorm (Conv (X_pool4, W9)))

X_b2 = ReLU (BatchNorm (Conv (X_b1, W10)))

Step 3. Decoder (Expansive Path)

Step 3.1: Apply up-convolution and concatenate with corresponding encoder output

X_up1 = UpConv (X_b2, W11, s=2)

X_concat1 = Concat (X_up1, X4)

Step 3.2: Apply convolutional block

X5 = ReLU (BatchNorm (Conv (X_concat1, W12)))

X6 = ReLU (BatchNorm (Conv (X5, W13)))

Step 3.3: Repeat up-convolution, concatenation, and convolutional block for remaining layers

X_up2 = UpConv (X6, W14, s=2)

X_concat2 = Concat (X_up2, X2)

X7 = ReLU (BatchNorm (Conv (X_concat2, W15)))

X8 = ReLU (BatchNorm (Conv (X7, W16)))

Repeat the process for 4 decoder layers

Step 4. Final Convolutional Layer

S = Conv (X_final, W_final)

S represents the segmentation map

Step 5. Flatten Layer

F = Flatten (X_final)

Step 6. Fully Connected Layer and Classification

L = Softmax (FC (F, W_fc))

L represents the classification label

Conv (X, W): Convolution operation on input X with weights W

BatchNorm(X): Batch normalization on input X

ReLU(X): ReLU activation function on input X

MaxPool (X, k, s): Max pooling operation on input X with kernel size k and stride s

UpConv (X, W, s): Up-convolution (transposed convolution) on input X with weights W and stride s

Concat (X1, X2): Concatenation of X1 and X2

Flatten(X): Flatten operation on input X

FC (X, W): Fully connected layer on input X with weights W

Softmax(X): Softmax activation function on input X

Table 3.2 shows the details of hyperparameter tuning in the UNET architecture.

Table 3.2 Hyperparameter Tuning for UNET Architecture

Parameters	**VGG16**
Input Size	224x224
Activation Function	ReLU and SoftMax
Loss function	Categorical Cross Entropy
Optimizer	Adam Optimizer
Learning Rate	1e-3
Batch size	16
Drop Out rate	0.5

3.6 VISUAL GEOMETRY GROUP 16 (VGG16)

VGG16 is a deep CNN used for image classification and feature extraction. VGG16 was developed by the "Visual Geometry Group at the University of Oxford" (Simonyan K. & Zisserman A., 2014). The VGG16 architecture for FV recognition is shown in Figure 3.10.

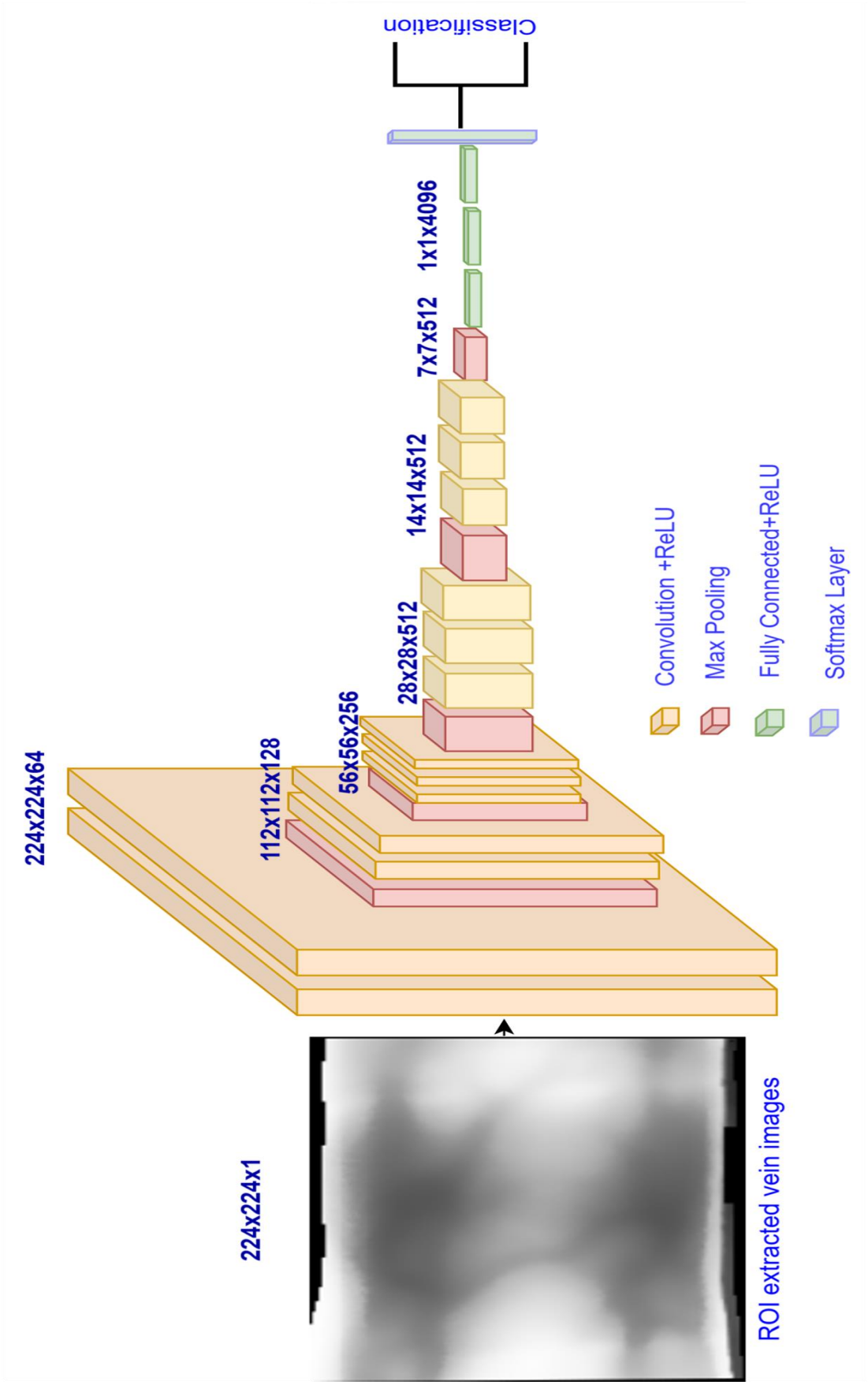


Figure 3.10 Architecture of VGG16 for Finger Vein Recognition

VGG16 architecture includes 13 convolutional layers, 5 max pooling layers, and 3 FCL, with a total of 16 weighted layers. The convolution layer uses a 3x3 filter with a stride of 1 and the same padding. The Rectified Linear Unit (ReLU) activation function helps in overcoming the gradient descent problem and is computationally inexpensive when compared to other activation functions. A 2x2 kernel with a stride of 2 is used in max pooling layers. This pooling operation reduces the dimension of the feature maps, which decreases computational load. It also helps in extracting the most significant features while discarding less important information.

The output from these convolutional and pooling operations is flattened and passed through a series of FCL. The FCL transforms the learned features into class predictions. The softmax layer then converts this into probability scores.

VGG16 was chosen over VGG19 due to several advantages that make it more suitable for FVR applications. VGG16 is made up of 16 layers compared to 19 layers in VGG19. A smaller number of layers and parameters reduces the complexity of VGG16, which makes it computationally more efficient to train and evaluate. VGG16 also reduces the risk of overfitting that will arise with the higher capacity of VGG19. VGG16 has consistently demonstrated good performance in related works for biometric recognition tasks (Wang Y., et al., 2022). Both VGG16 and VGG19 are effective at extracting hierarchical features, but the additional layers in VGG19 are unnecessary for tasks where the dataset does not require a deeper model.

Table 3.3 shows the details of hyperparameter tuning in the VGG16 architecture.

Table 3.3 Hyperparameter Tuning for VGG16 Architecture

Parameters	**VGG16**
Input Size	224x224
Activation Function	ReLU and SoftMax
Loss function	Categorical Cross Entropy
Optimizer	Adam Optimizer
Learning Rate	1e-4
Batch size	16
Drop-out rate	0.5

3.7 RESULTS AND DISCUSSIONS

The modified UNET and VGG16 models are evaluated using THUFV and SDUMLA-HMT datasets to measure the effectiveness of these models in extracting and classifying the vein patterns. The performance assessments are done using accuracy, precision, recall and F1-score metrics as given in equations 3.1 to 3.4.

a. Accuracy: It evaluates how accurately the model performs. It represents the ratio of accurate predictions to the total predictions.

$$\text{Accuracy} = \frac{(True\ Positives + True\ Negatives)}{Total\ Predictions} \tag{3.1}$$

b. Precision: It measures the proportion of the predicted positive outcomes are true positives.

$$\text{Precision} = \frac{True\ Positives}{(True\ Positive + False\ Positives)} \tag{3.2}$$

c. Recall: It measures the proportion of the actual positives that are correctly identified.

$$\text{Recall} = \frac{True\ Positives}{(True\ Positives + False\ Negatives)} \tag{3.3}$$

d. F1-Score: F1-Score combines precision and recall into a single value, to provide a balanced assessment of the model's performance.

$$\text{F1-Score} = \frac{(2(Precision * Recall))}{(Precision + Recall)} \tag{3.4}$$

Even though the dataset is balanced, the F1-score remains important as it provides deeper insight into the trade-off between recall and precision, which is important in situations where the impact of false positives and false negatives has varying consequences. In biometric authentication, a false negative (failing to recognize an authorized user) may be just as critical as a false positive (incorrectly granting access to an unauthorized user). The F1-score helps evaluate how well the model balances these two types of errors, ensuring that both precision and recall are optimized for the specific application's needs.

3.7.1 Performance Evaluation of Modified UNET

a. THUFV Dataset

The performance of the model is assessed from accuracy and loss curves on the THUFV dataset. The metrics have been stabilized by about 50 epochs, and further training did not improve the performance, hence, training has been stopped at 50 epochs.

The 'Accuracy vs. Epochs' curve in Figure 3.11 shows that the training accuracy begins at 30% and ascends to approximately 85% within the first 10 epochs, then gradually stabilizes around 88-90%. The validation accuracy follows a similar trend, starting at 50%, rising sharply, and stabilizes at 80%.

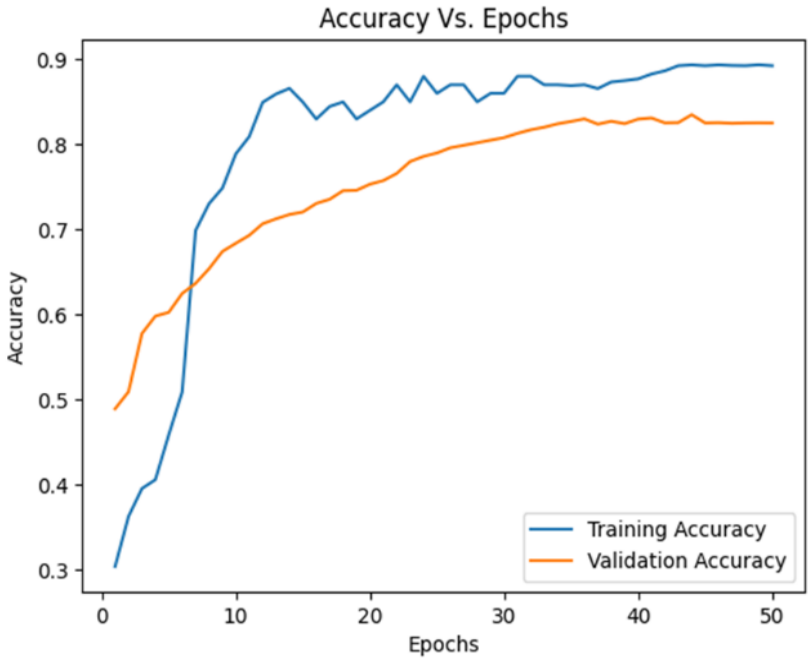


Figure 3.11 Accuracy Curve of Modified UNET on THUFV Dataset

In the 'Loss vs. Epochs' graph shown in Figure 3.12, a rapid decline in training and validation loss is observed during the initial phase.

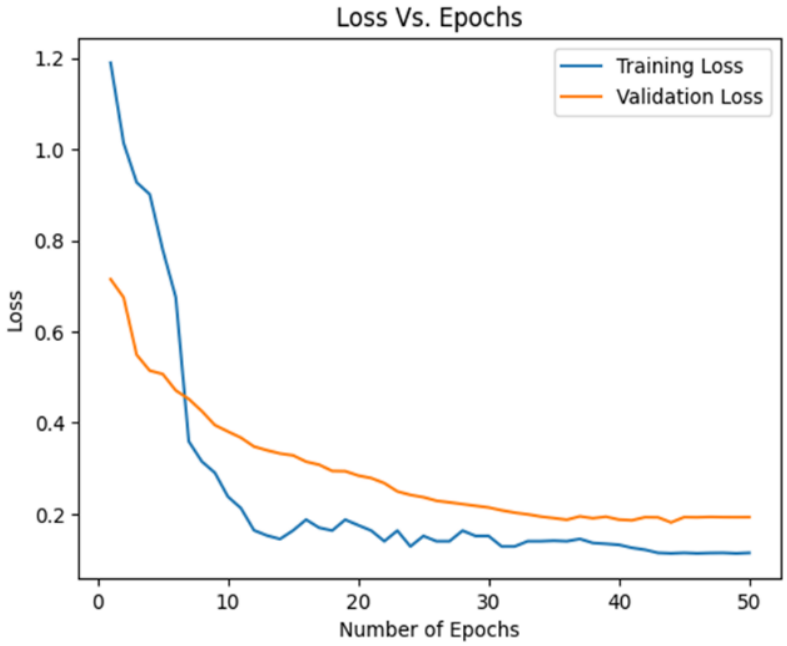


Figure 3.12 Loss Curve of Modified UNET on THUFV Dataset

The training loss drops from 1.2 to below 0.2, stabilizing around 0.1, while the validation loss decreases from 0.6 to approximately 0.2, showing consistent convergence. The gap between training and validation accuracy, along with the divergence in their respective loss values, indicates the possibility of overfitting in the model. The training accuracy consistently exceeds that of the validation set, while the training loss remains persistently lower, indicating a potential overfitting tendency in the model. The validation loss starts relatively high and decreases more slowly than the training loss. This implies that the model initially struggles more with unseen data, suggesting a need for better regularization techniques or more diverse training data.

b. SDUMLA Dataset

The accuracy and loss graphs for the modified UNET architecture are also analyzed on the SDUMLA dataset. Figure 3.13 shows the accuracy achieved during the model's training phase and its evaluation on validation data increases significantly during the initial epochs.

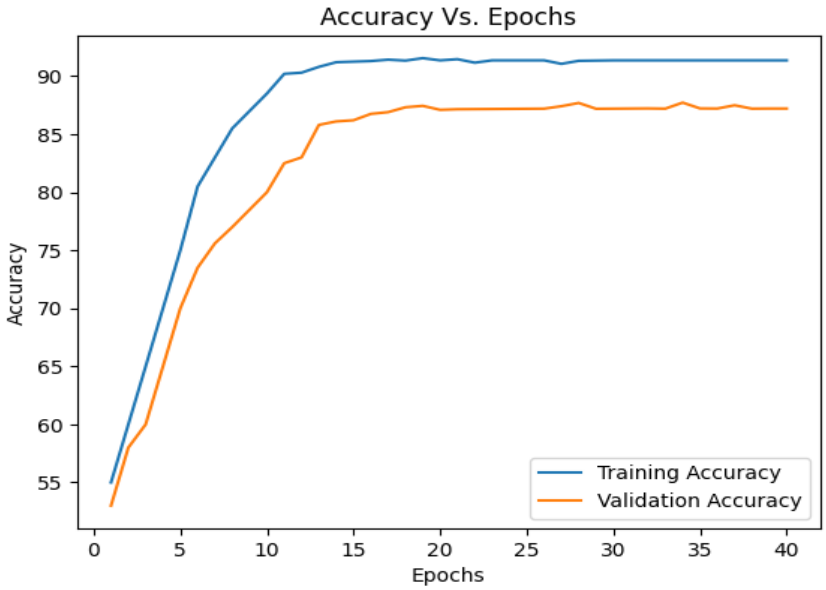


Figure 3.13 Accuracy Curve of Modified UNET on SDUMLA Dataset

The training accuracy stabilizes around 91.36%, whereas validation accuracy is around 85%. A noticeable difference between the training and validation accuracy indicates its superior performance on the training dataset relative to unseen validation data.

In Figure 3.14, the loss decreases sharply during early epochs, indicating the model is learning. Training loss drops to a very low value, around 0.05, indicating that the model has effectively learned from the training set. Validation loss decreases but stabilizes around 0.15, higher than the training loss, indicating some generalization error.

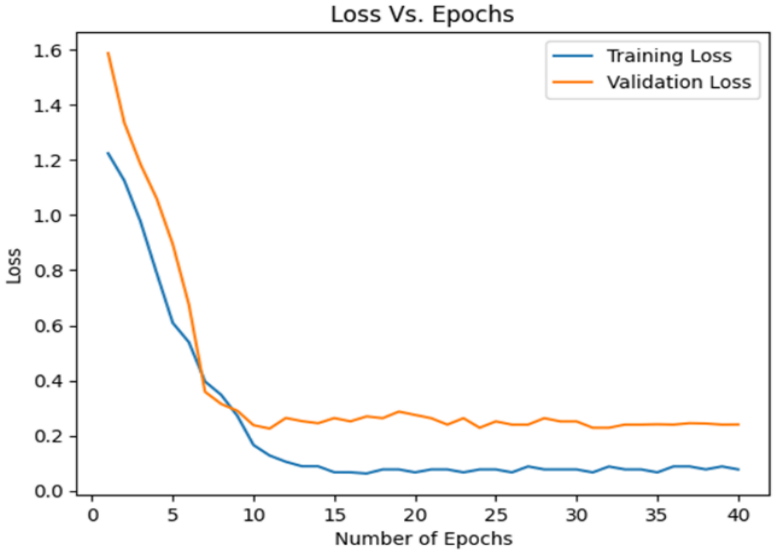


Figure 3.14 Loss Curve of Modified UNET on SDUMLA Dataset

The disparity between the low training loss and comparatively higher validation loss signifies that the model has learned the training data effectively but may be overfitting, thereby reducing its generalization capability. The increased training accuracy and reduced training loss compared to their validation counterparts show signs of overfitting of the model.

3.7.2 Performance Evaluation of VGG16

a. THUFV Dataset

Figure 3.15 depicts the progress of accuracy for VGG16 on the THUFV dataset. The accuracy graph shows an upward trend for both curves. The training accuracy stabilizes at 92.5%, and the validation accuracy is around 85%.

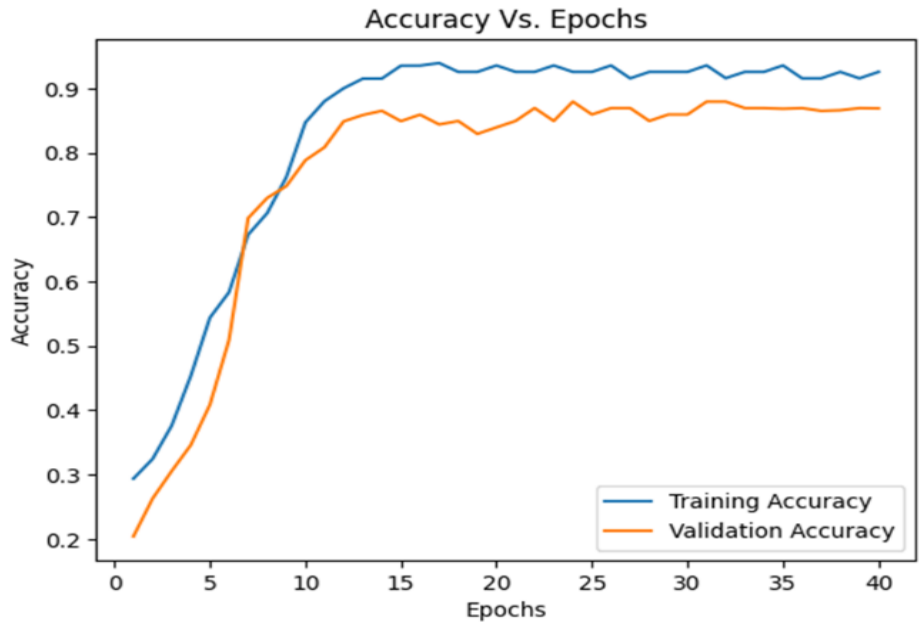


Figure 3.15 Accuracy Curve of VGG16 on THUFV Dataset

The loss graph in Figure 3.16 illustrates that both “training and validation loss” decrease during the initial epoch, with training loss reaching a lower level than validation loss. The difference observed between training and validation performance indicates that the model may not generalize as desired.

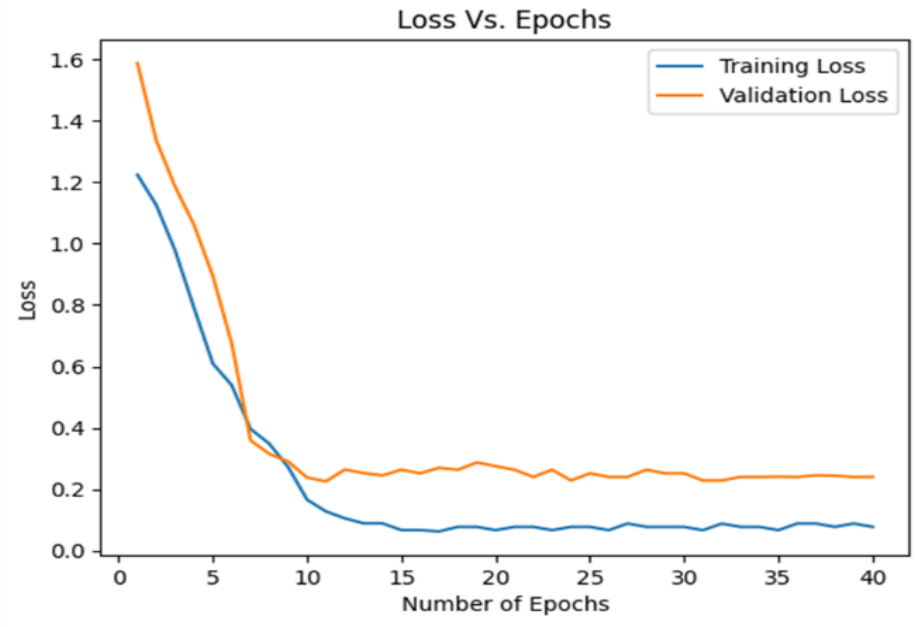


Figure 3.16 Loss curve of VGG16 on THUFV dataset

b. SDUMLA Dataset

Figure 3.17 illustrates the accuracy of the VGG16 model on the SDUMLA dataset over 25 epochs.

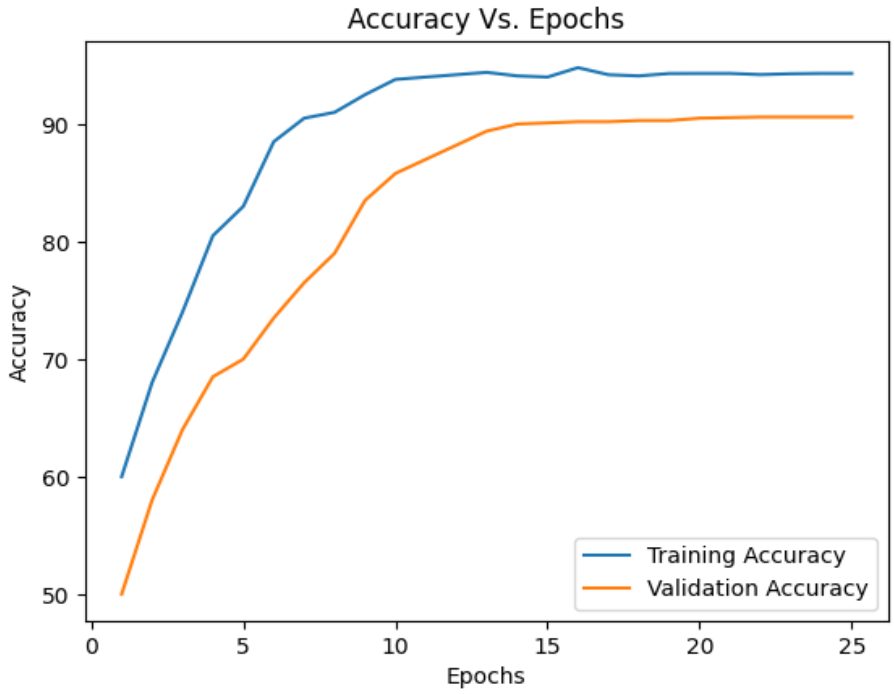


Figure 3.17 Accuracy Curve of VGG16 on SDUMLA Dataset

Both learning accuracy and validation accuracy increase significantly during the initial epochs. The learning accuracy quickly rises, eventually stabilizing around 94.31%, while the validation accuracy stabilizes at a slightly lower value, around 85%.

In Figure 3.18, both "training and validation loss" decline sharply in the initial epochs, suggesting that the model is learning effectively.

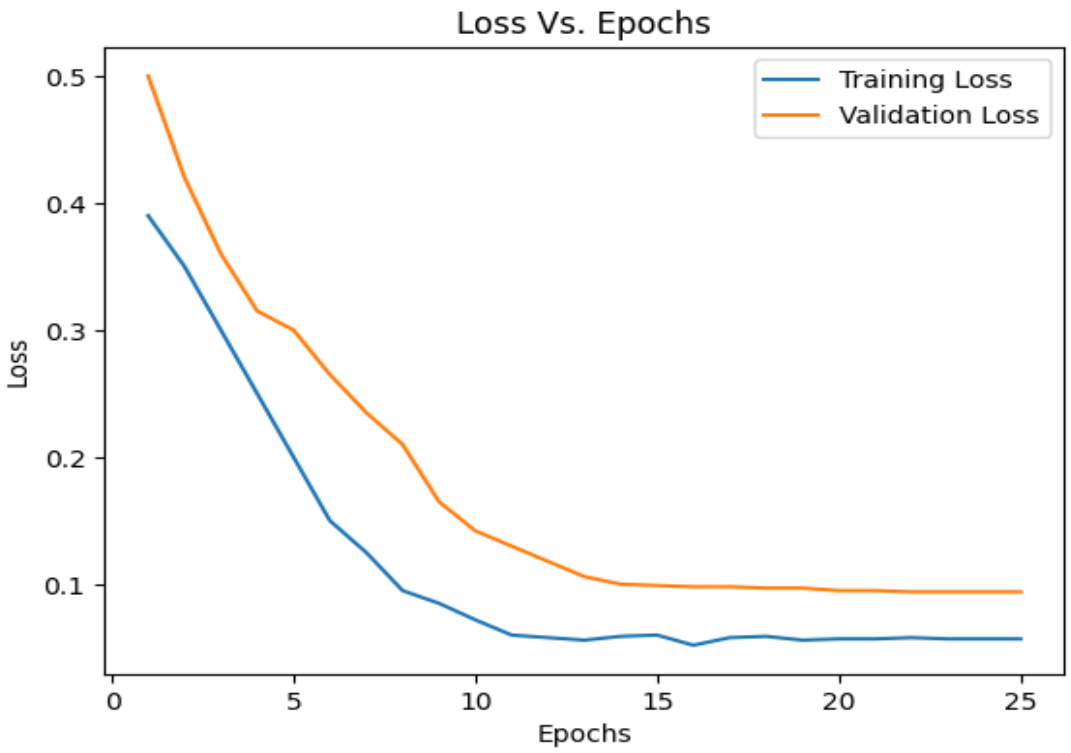


Figure 3.18 Loss Curve of VGG16 on SDUMLA Dataset

The training loss continues to decline and levels off at a very low value of approximately 0.05, indicating that the model demonstrates a strong fit to the training data. The validation loss stabilizes around 0.15, which suggests a minor generalization error.

3.7.3 Comparison of Modified UNET and VGG16

a. THUFV Dataset

Table 3.4 shows the comparison of the accuracy of the modified UNET and VGG16 with existing works on the THUFV dataset.

Table 3.4 Comparison of Accuracy of Modified UNET and VGG16 with Existing Algorithms on THUFV Dataset

Model	Patch-DNN + P-SVM	DNN + P-SVM	PCA + SVM	Modified UNET	VGG16
Accuracy (%)	79.9	81.98	88.21	**89.17**	**92.5**

Modified UNET and VGG16 achieved an accuracy of 89.17% and 92.5%, respectively, which is higher than the accuracy of the existing algorithms, such as Patch-DNN+PSVM, DNN + P-SVM, and PCA + SVM. The accuracy of VGG16 is 3.33% more than the modified UNET, indicating that VGG16 is more suited for the task. Table 3.5 presents the precision, recall and F1-score of the modified UNET and VGG16 models on the THUFV dataset.

Table 3.5 Performance Comparison of Modified UNET and VGG16 on THUFV Dataset

Model	Precision (%)	Recall (%)	F1-Score (%)
Modified UNET	93.61	94.47	94.03
VGG16	94.09	97.64	95.83

The VGG16 model demonstrates greater performance across all evaluated metrics. In terms of precision, VGG16 records a slightly higher value of 94.09% compared to the modified UNET (93.61%), reflecting a 0.48% improvement. The most significant enhancement is seen in the recall, where VGG16 attains 97.64%, a substantial increase of 3.17% over the modified UNET's 94.47%. This improvement in recall indicates that VGG16 is particularly more effective in identifying true positive cases. Consequently, the F1-Score for VGG16 is 95.83%, which is 1.80% higher than the modified UNET's 94.03%. Overall, the VGG16 model provides notable advancements in accuracy, recall, and F1-Score, highlighting its effectiveness in FVR tasks on the THUFV dataset.

b. SDUMLA Dataset

The performance metrics presented in Table 3.6 illustrate a comparison between a modified UNET model and the VGG16 model on the SDUMLA-HMT dataset. Table 3.6 shows the comparison of the accuracy of the modified UNET and VGG16 with the existing algorithms such as SRC, Alexnet, PCA+SVM, Reset on the SDUMLA dataset.

Table 3.6 Comparison of Accuracy of Modified UNET and VGG16 with Existing Algorithms on SDUMLA Dataset

Model	MSRC	Alexnet	PCA + SVM	Resnet	Modified UNET	VGG16
Accuracy (%)	68.08	70.20	86.87	90.07	**91.36**	**94.31**

The accuracy of VGG16 is 94.31%, which is 2.95% more than the modified UNET's 91.36% and outperforms other existing algorithms. Table 3.7 presents the precision, recall and F1-score of the modified UNET and VGG16 models on the SDUMLA dataset.

Table 3.7 Performance Comparison of Modified UNET and VGG16 on SDUMLA Dataset

Model	Precision (%)	Recall (%)	F1-Score (%)
Modified UNET	89.61	91.07	90.33
VGG16	93.60	92.07	92.83

The VGG16 model outperforms the modified UNET, demonstrating a clear improvement across all the metrics. In terms of precision, VGG16 records 93.60%, an increase of 3.99% over the modified UNET's 89.61%, indicating that VGG16 is more effective in correctly identifying true positive cases. The recall for VGG16 is 92.07%,

which, while only 1% higher than the modified UNET's 91.07%, still reflects an enhanced effectiveness in detecting genuine positive instances. Finally, the F1-score for VGG16 is 92.83%, 2.50% more than the modified UNET. These findings indicate that VGG16 performs better than modified UNET for FVR tasks.

3.8 SUMMARY

Experiments were conducted using modified UNET and VGG16 architectures to evaluate their performance for FVR tasks. Both models were evaluated using the SDUMLA-HMT dataset and the THUFV dataset. The results demonstrated that VGG16 significantly outperformed the Modified UNET for various performance metrics. VGG16 achieves higher accuracy than modified UNET, with 3.33% more for the THUFV dataset and 2.95% more for the SDUMLA dataset. The precision, recall, and F1 score also indicate that VGG16 is better suited for FVR than the Modified UNET architecture.