Report on

Thumbs-Up Vs Peace Sign Binary Classification Using Transfer Learning

Submitted by - Group 6

Joy Karmoker

Student ID: 1810176127

Masum Billah

Student ID: 1910576118

Jannatun Ferdous

Student ID: 1912076147

Shahin Alam

Student ID: 1810176115

Contents

1 Introduction

In computer vision and artificial intelligence, the ability to interpret and comprehend gestures holds immense significance. Sign language, as a form of non-verbal communication, is crucial in facilitating communication for individuals with hearing impairments. Recognizing and classifying common sign language symbols using advanced technologies can pave the way for more inclusive and accessible human-computer interactions. This project focuses on the binary classification of two iconic sign language gestures: the thumbs-up and the peace sign. These gestures, transcending cultural and linguistic boundaries, convey positive sentiments and are integral to everyday communication. Leveraging the power of deep learning, we employed two distinct neural network architectures, namely a modified VGG16 and a customized ResNet50, to perform binary classification tasks. Utilizing transfer learning, we harnessed the capabilities of pre-trained models, fine-tuning them with our dataset to enhance their adaptability to the specific nuances of sign language gestures. In this report, we provide a comprehensive overview of the project, encompassing the data used, and the architectural modifications implemented for VGG16 and ResNet50, as well as an in-depth exploration of the results and discussions arising from our experiments. The report unfolds in a structured manner, beginning with a detailed description of the dataset. The subsequent section delves into the architectural modifications made to VGG16 and ResNet50. Moving forward, Result and Discussion presents the outcomes of our experiments and engages in discussions regarding the performance of the two models. Finally, we draw conclusions.

Through this endeavour, our primary objectives include leveraging the power of transfer learning with VGG16 and ResNet50 architectures for the binary classification of thumbs-up and peace signs. We aim to assess the performance of these pre-trained models after fine-tuning with our dataset, comparing their classification accuracies, and elucidating the strengths and limitations of employing transfer learning in this context. The outcomes of this study contribute valuable insights into the effectiveness of transfer learning with well-established architectures for sign language recognition, with implications for fostering inclusivity in technological interfaces.“‘

2 Dataset Description

2.1 Source of data

There are two types of captured data:

i) Thumbs-up sign: This gesture, formed by raising the thumb, symbolizes a positive affirmation or agreement.

ii) Peace sign: Formed by raising the index and middle fingers together, this gesture signifies the number two or conveys a message of peace.

some examples of source data:

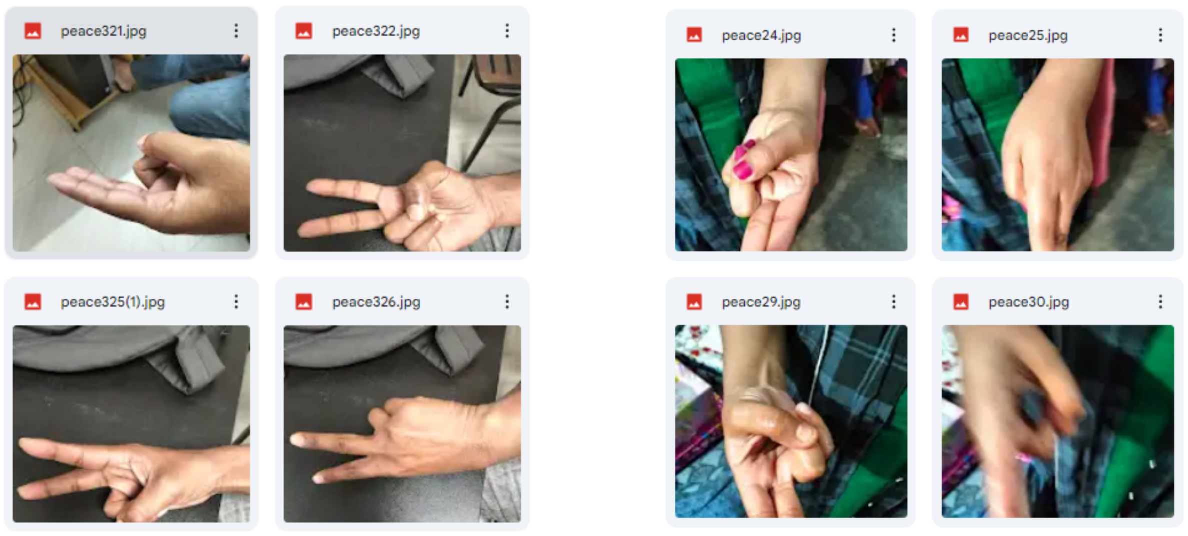


Figure 1: Peace sign test and train dataset

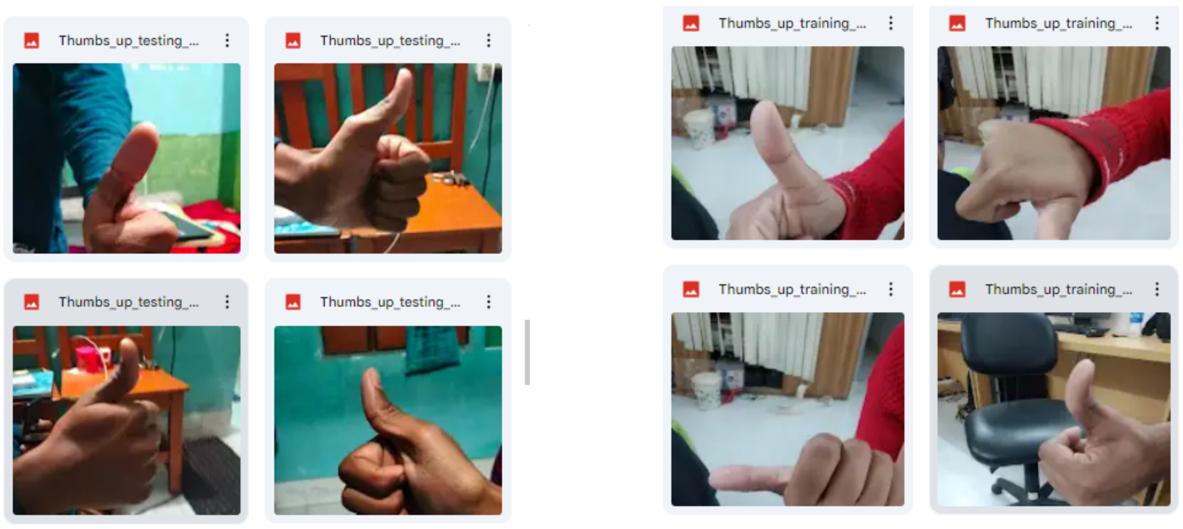


Figure 2: Thumbs-up sign test and train dataset

We have captured 800 (400 images for each classification) RGB images with the help of conventional mobile camera and collected the dataset from 100 person by capturing 8 images

from each person in different angle. All the images are taken with permission from individual. All the images are kept into the google drive after collecting from our group members.
Drive link: `https://drive.google.com/drive/u/3/folders/1DSKOBP34OtMqk2Vxmd2pE`

2.2 Categorical classification

Our binary classification dataset is divided into four categories:

- Peace sign for train:This section contains images of the peace sign used for training.
- Peace sign for test:Here, you'll find images of the peace sign reserved for testing the model.
- Thumbs-up sign for train: Images of the thumbs-up sign are included here for model training.
- Thumbs-up sign for test:This section holds images of the thumbs-up sign for evaluating the trained model.

2.3 Splitting the data

To manage the training and testing data splits for the "Peace" and "Thumbs-up" , we can follow these steps:

Data Preparation: To manage the training and testing data splits for the "Peace" and "Thumbs-up" categories, we can follow these steps. Firstly, in the data preparation phase, we divide the dataset into training and testing sets. Specifically, 80% of the data is reserved for training purposes, while the remaining 20% is allocated for testing.

Training Set Splitting: within the training data, 90% is utilized for the actual training process. The remaining 10% of the training data is set aside for validation purposes. This ensures that the model is trained on a significant portion of the data while still having a subset reserved for validating its performance during the training process.

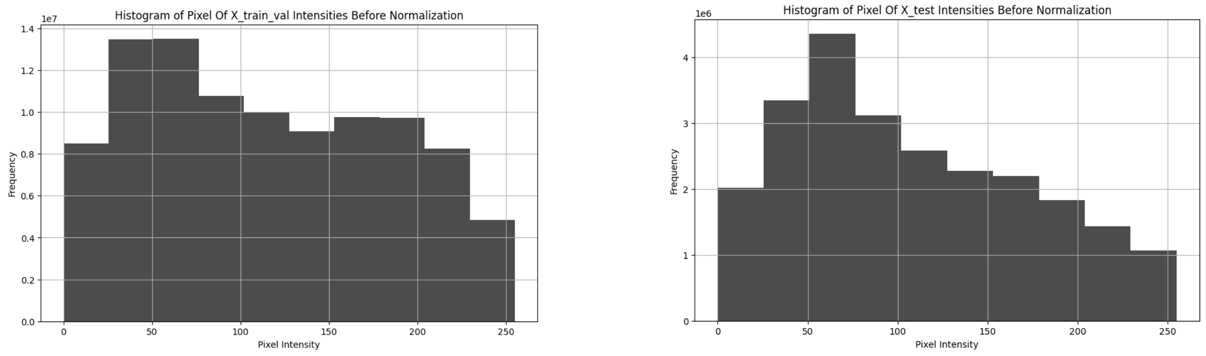


Figure 3: fig:X_train_val and X_test

Testing Set Usage:

- Use the entire testing set (20% of the original data) for evaluating the models trained on the respective training datasets.

2.4 Data Normalization

X_train_val and X_test data are normalized. The data is split into training and validation sets, with 90% used for model training and 10% reserved for validation.

2.4.1 Pixel Intensity Distribution (Before Normalization)

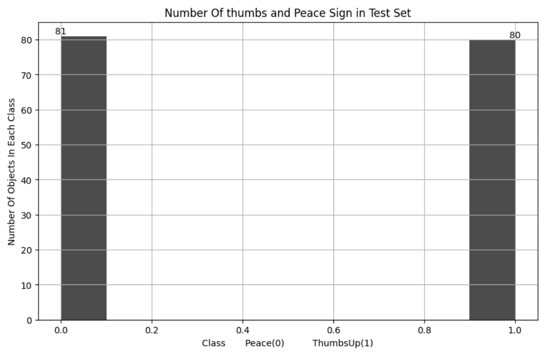
Histograms are plotted for X_train_val and X_test before normalization to visualize pixel intensity distribution. The highest and lowest intensity of pixels are printed.

2.4.2 Pixel Intensity Distribution (After Normalization)

Histograms are plotted for X_train_val and X_test after normalization to visualize pixel intensity distributions. Additionally, the highest and lowest intensity of pixels are printed.

2.5 Class Distribution in Training and Validation Set

A histogram is displayed showing the distribution of classes (Peace and ThumbsUp) in both the training set and validation set.We plot the Train Set,Test Set and Validation Set.



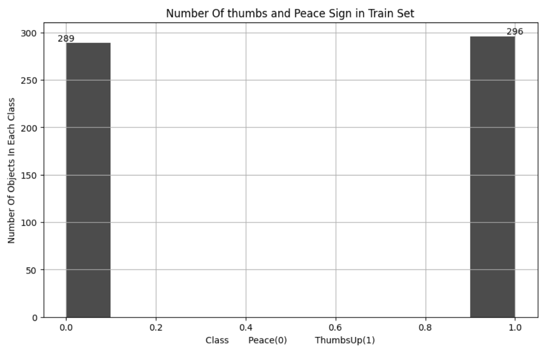


Figure 4: Test Set and Train Set

Figure 5: Caption

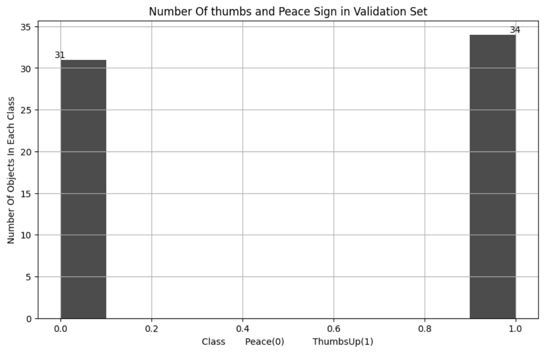


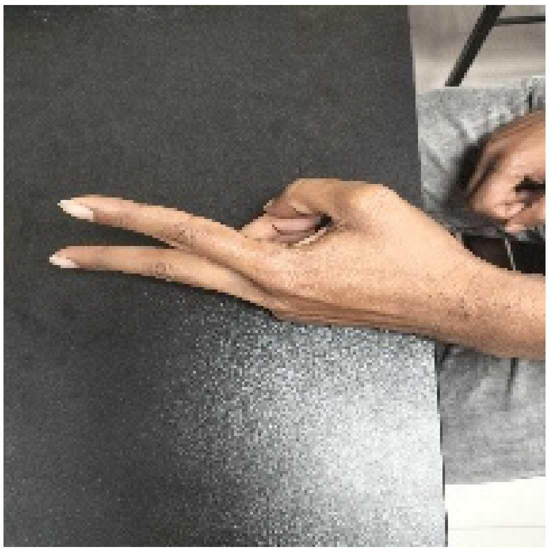
Figure 6: validation Set

2.6 Labeling the Data

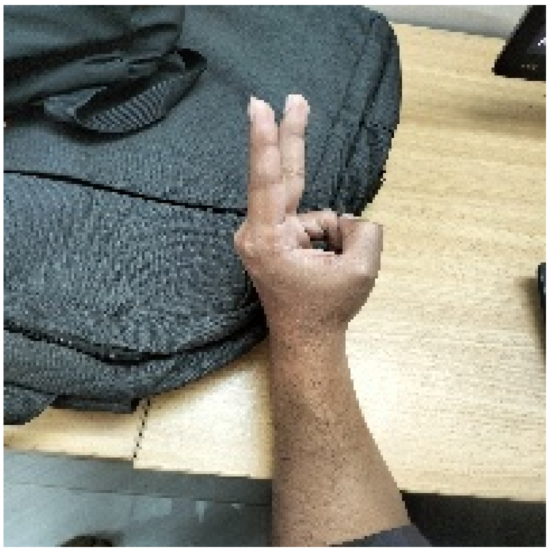
- We have 4 categories of data set which is defined previously. First two dataset is for peace sign which is labeled as 0 and the last two dataset is for Thumbs-up which is labeled as

1.

- resized images: all the images are resized into 224x224 pixels.

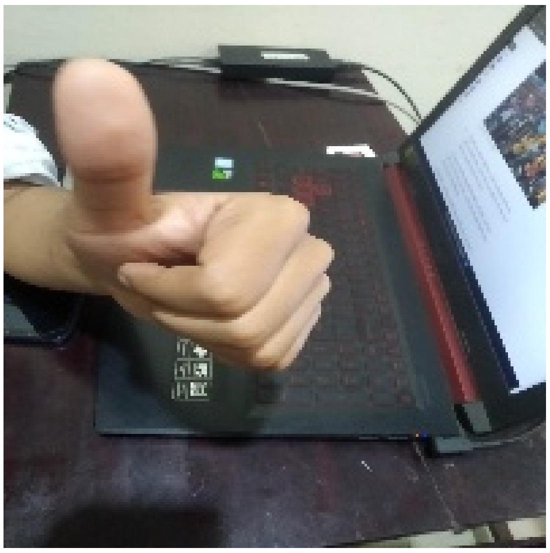


Label: 0

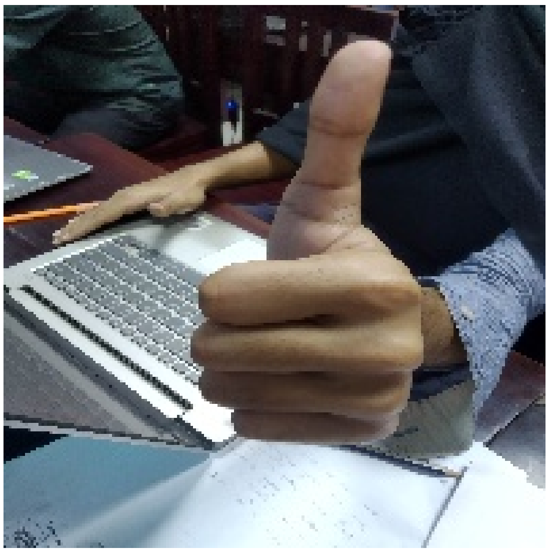


Label: 0

Label: 0



Label: 1



Label: 1

Label: 1

Images labeled with Peace sign (Label: 0) and Thumbs-up sign (Label: 1)

2.7 Shapes of Different data

The shape of the training data (x) is (585, 224, 224, 3), indicating that there are 585 samples with an image size of 224x224 pixels and 3 color channels (RGB). Similarly, the validation data (x) has a shape of (65, 224, 224, 3), containing 65 samples with the same image dimensions.

The test data (x) has a shape of (161, 224, 224, 3), consisting of 161 samples with the same image size and color channels as the training and validation sets.

Regarding the labels (y), the shape of the training data is (585,), indicating there are 585 corresponding labels. The validation data's label shape is (65,), containing 65 labels, while the test data's label shape is (161,), indicating 161 labels corresponding to the test samples.

3 Architecture

In this section, we delve into the architectural details of our binary classification project, focusing on transfer learning with two prominent neural network architectures: VGG16 and ResNet50.

3.1 VGG16 Transfer Learning

For VGG16, we leveraged the pre-trained weights from the ImageNet dataset using the `tf.keras.applications.VGG16` module. The model was configured with an input shape of (224, 224, 3) to accommodate our image data. The sequential model includes the VGG16 convolutional base, followed by a flattening layer to transform the output into a one-dimensional array. To prevent overfitting, we introduced a dropout layer with a dropout rate of 0.5. The final layer is a dense layer with two units and a sigmoid activation function, suitable for binary classification. Importantly, we set the VGG16 layers to be non-trainable, only training the additional layers introduced for our specific task.

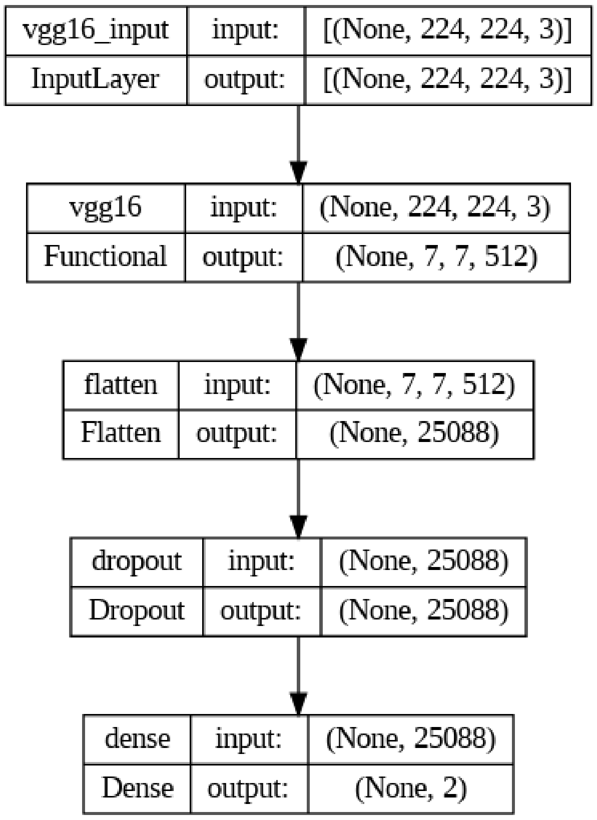


Figure 14: Architecture of VGG16 Transfer Learning Model

The resulting architecture, as illustrated in Figure 14, provides a compact representation of the model, with a total of 14,714,688 parameters.

3.2 ResNet50 Transfer Learning

For ResNet50, we employed the `tf.keras.applications.ResNet50` module with the same philosophy of utilizing pre-trained ImageNet weights. The model architecture consists of a ResNet50 convolutional base followed by a global max-pooling layer. A dense layer with two units and a softmax activation function serves as the final layer for binary classification. Similar to VGG16, we set the ResNet50 layers to be non-trainable.

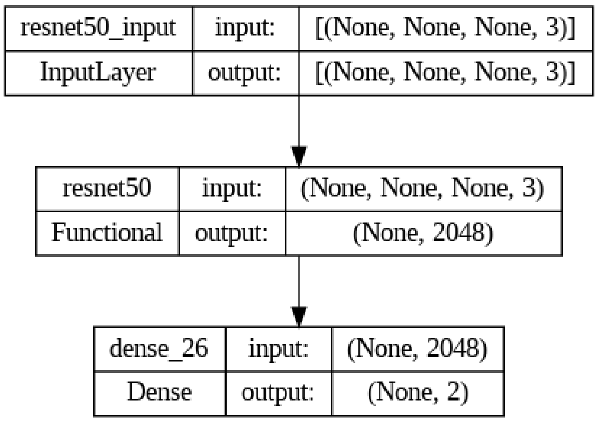


Figure 15: Architecture of ResNet50 Transfer Learning Model

The architecture overview is presented in Figure 15, illustrating the simplicity of the model with a total of 23,591,810 parameters.

The dropout layer in the VGG16 model aids in preventing overfitting by randomly deactivating a fraction of neurons during training. This regularization technique enhances the model's generalization performance on unseen data.

In both architectures, we employed the Adam optimizer with a learning rate of 0.001, and the model performance was evaluated using relevant metrics such as accuracy and loss during training.

The subsequent sections of this report delve into the experimental results, discussions, and conclusions derived from our exploration of these transfer learning models.

4 Results

4.1 Overview of the used model

In our transfer learning approach, we employed two distinct models: VGG16 and ResNet. These models serve as powerful architectures pre-trained on the ImageNet dataset, providing a robust foundation for feature extraction.

VGG16 model: we utilized its deep convolutional layers to capture hierarchical features from input images. By initializing the model with pre-trained weights and customizing the top layers for our binary classification task, we effectively leveraged its learned representations while adapting it to our specific problem. During training, only the top layers were trainable, ensuring that the model focuses on learning task-specific features without disturbing the already learned representations in the lower layers.

ResNet modell: We benefited from its unique architecture incorporating residual connections, which facilitated the training of very deep neural networks. By initializing ResNet with pre-trained weights and modifying the top layers for our binary classification task, we harnessed its capacity to extract rich features from input images. Like VGG16, during training, we kept the lower layers frozen while fine-tuning only the top layers to prevent overfitting and preserve the learned representations.

In summary, both VGG16 and ResNet served as effective frameworks for transfer learning, enabling us to adapt pre-trained models to our specific classification task with limited data while maximizing computational efficiency and performance.

4.2 Learning curve for train loss and accuracy

VGG16 model:The learning curve for the VGG16 model shows how the model learns over time.

In the first plot, we see how the training loss (in purple) decreases as the model learns from the data. The validation loss (in orange) also decreases initially but may start to level off or increase later, indicating potential overfitting.

In the second plot, we observe the training accuracy (in purple) and validation accuracy (in orange) increasing over epochs. However, the gap between them may widen, suggesting the model is becoming too specialized in the training data and may struggle to generalize to new data.

ResNet modell: The two plots show how our model learns and performs during training.

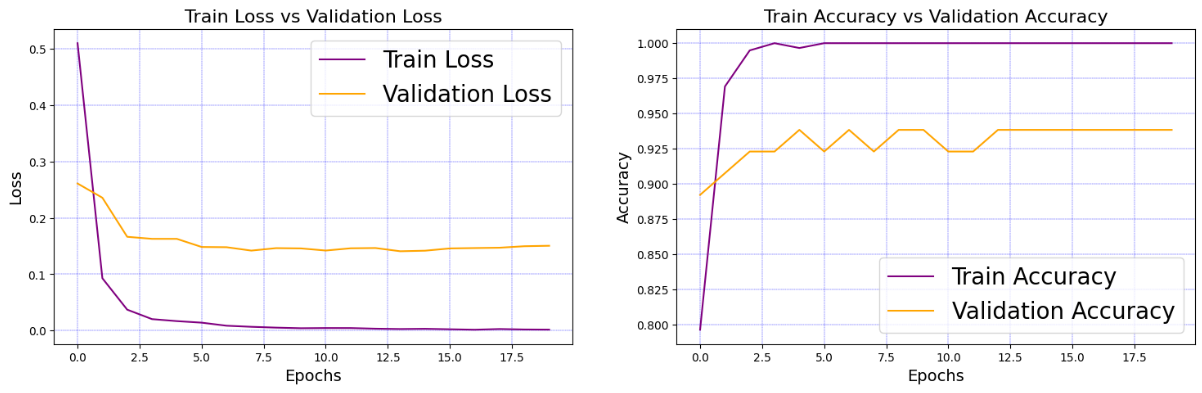


Figure 16: Train vs Validation test for vgg16(transfer learning)

The first plot displays the training loss (in purple) and validation loss (in orange) over different epochs. We want both losses to decrease, but if the validation loss starts to rise while the training loss continues to fall, it suggests our model might be overfitting.

The second plot illustrates the training accuracy (in purple) and validation accuracy (in orange) across epochs. We aim for both accuracies to increase, but if the gap between them widens, it might indicate overfitting, meaning our model is getting too good at the training data but not generalizing well to new data.

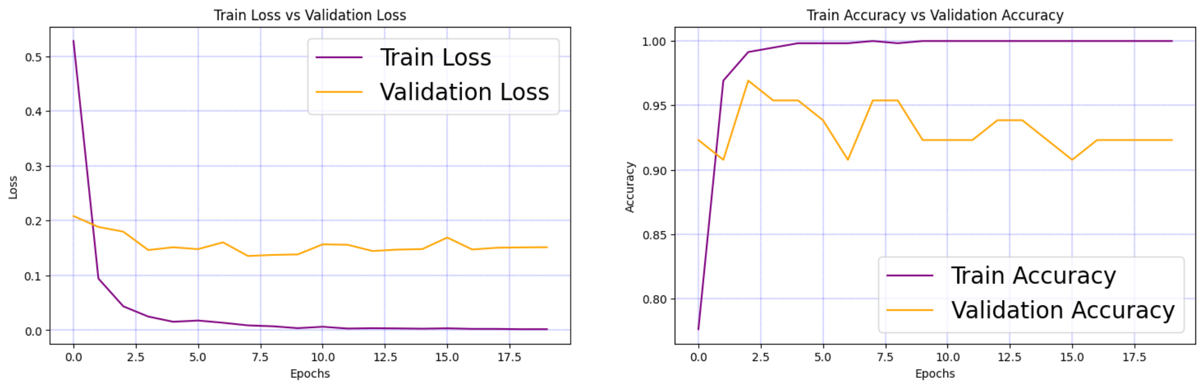


Figure 17: Train vs Validation test for Restnet (transfer learning)

4.3 The variations of results (table)

Table 1: Performance of VGG16 transfer learning on various settings

Epochs	Learning Rate	Batch Size	Test		Validation	
			Loss	Accuracy	Loss	Accuracy
5	0.001	16	0.0187	0.9983	0.0984	0.9692
10	0.001	16	0.0076	1.00	0.0796	0.9692
20	0.0001	16	0.1031	0.499	0.1085	0.4615
20	**0.001**	**16**	**0.0020**	**1.00**	**0.0774**	**0.9692**
20	0.001	32	0.0043	1.00	0.2159	0.9077
20	0.001	64	0.0074	1.00	0.1969	0.9231
30	0.001	16	0.0009	1.00	0.2109	0.9385

Table 2: Performance of Resnet50 transfer learning on various settings

Epochs	Learning Rate	Batch Size	Test		Validation	
			Loss	Accuracy	Loss	Accuracy
20	0.01	16	5.5506	0.7111	2.5666	0.7874
20	0.001	16	0.2869	0.8940	0.4084	0.8154
40	0.001	16	0.2001	0.9248	0.3395	0.8308
100	0.001	16	0.1093	0.9675	0.2979	0.8769
100	0.001	31	0.1359	0.9658	0.300	0.8923
30	0.0001	16	0.0009	1.00	0.2109	0.9385

4.4 Prediction of Test Data

The model's predictions on the test data are checked. Learning curves for FCNN, We evaluate the FCNN model with Validation Data. We get an accuracy of in validation data 74%. Here accuracy is decreased. When We evaluate the FCNN model on test Data, we get an accuracy of 86%.

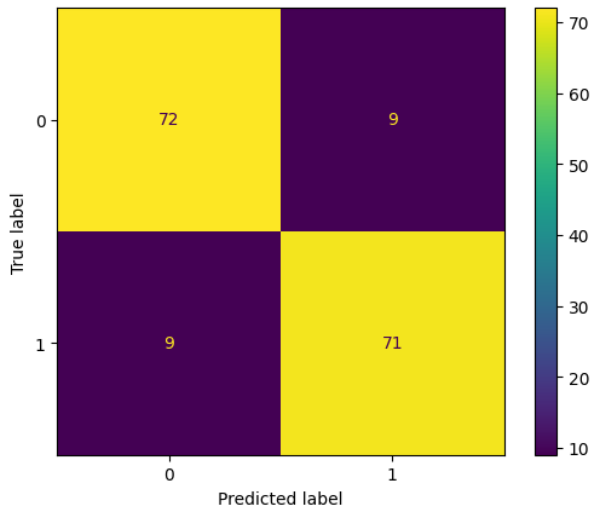


Figure 18: Confusion Matrix of CNN

The FCNN model is evaluated on both validation and test data. Evaluation metrics include loss and accuracy. Additionally, a confusion matrix is generated to visualize classification performance on the test data.

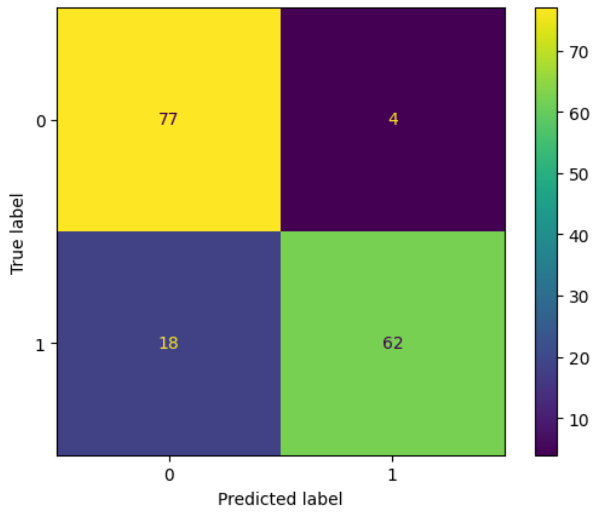


Figure 19: Confusion Matrix of FCNN