CHAPTER-4

Proposed Methodology

4.1 Introduction

In this chapter, we delve into the heart of our proposed methodology for addressing the classification problem at hand, which leverages the power of transfer learning. The introduction provides an overview of the primary architectural components we will be working with, namely the VGG16 and ResNet50 architectures, both in their original forms and as modified models tailored to our specific requirements.

The framework of the proposed model for classification of real and retouching faces is described in detailed first. The vgg16 model detailed architecture is explained and the modified architecture is introduced later. The second widely used CNN model, namely ResNet50 and its acrhitecture is explained, followed by modified version of the CNN which is used for this research task.

4.2 Proposed Methodology using Transfer Learning

Transfer Learning can be used in variety of fields like medical, weather reporting, forecasting, road map detection, image retouching to classify the deceases, or cancer or tumors, sky conditions, map detection and to detect forgery on images, etc. As compared to ML & DL approach, TL(transfer learning) approach are faster and trained more accurately than other traditional methods like manual grading and other machine vision techniques or other classifiers.[19] there are several challenges ,when using DL(Deep Learning) model to detect retouching on facial images. 1. Need of large facial dataset containing real and retouched face images 2. Proper labelled data 2.large amount of images for training the mdoel for accurately detecting retouched images this is difficult to obtain such a large facial dataset. the DL models are prone to overfitting too which leads to give biased output. Using, TL, all these challenges are overcome and optimal detection accuracy can be achieved. TL offers following advantages in ML and DL tasks[20],

1. Reduced Training Time: Transfer learning allows you to leverage pre-trained models that have been trained on large datasets. By using a pre-trained model as a starting point, you can save a significant amount of time and computational resources that would otherwise be required to train a model from scratch.
2. Lower Data Requirements: Training deep learning models often requires large amounts of labeled data. Transfer learning enables you to overcome this challenge by using the knowledge gained from a source task (where data may be abundant) to improve the performance on a target task with limited data. This is particularly beneficial in scenarios where collecting large amounts of labeled data is difficult or expensive.
3. Improved Generalization: Transfer learning helps improve the generalization capabilities of a model. Pre-trained models have learned useful features from a diverse range of data, which can be transferred to a new task. This transfer of knowledge allows the model to extract relevant features from the data more effectively, even when the target task has different characteristics or a smaller dataset.
4. Avoiding the "Cold Start" Problem: When starting a new machine learning project, especially with limited data, it can be challenging to train an accurate model from scratch. Transfer learning helps overcome the "cold start" problem by providing a well-initialized model that has already learned useful representations from a different but related task.
5. Effective in Domain Adaptation: Transfer learning is highly useful in domain adaptation scenarios, where the distribution of the source data differs from the target data. By using a pre-trained model on a source domain and fine-tuning it on the

target domain, transfer learning enables the model to adapt and perform well in the target domain.

6. Useful for Small-Scale Deployment: In resource-constrained environments, such as edge devices or mobile applications, transfer learning allows you to deploy efficient models that consume less computational power and memory. By leveraging pre-trained models and fine-tuning them on specific tasks, you can achieve good performance with fewer resources.

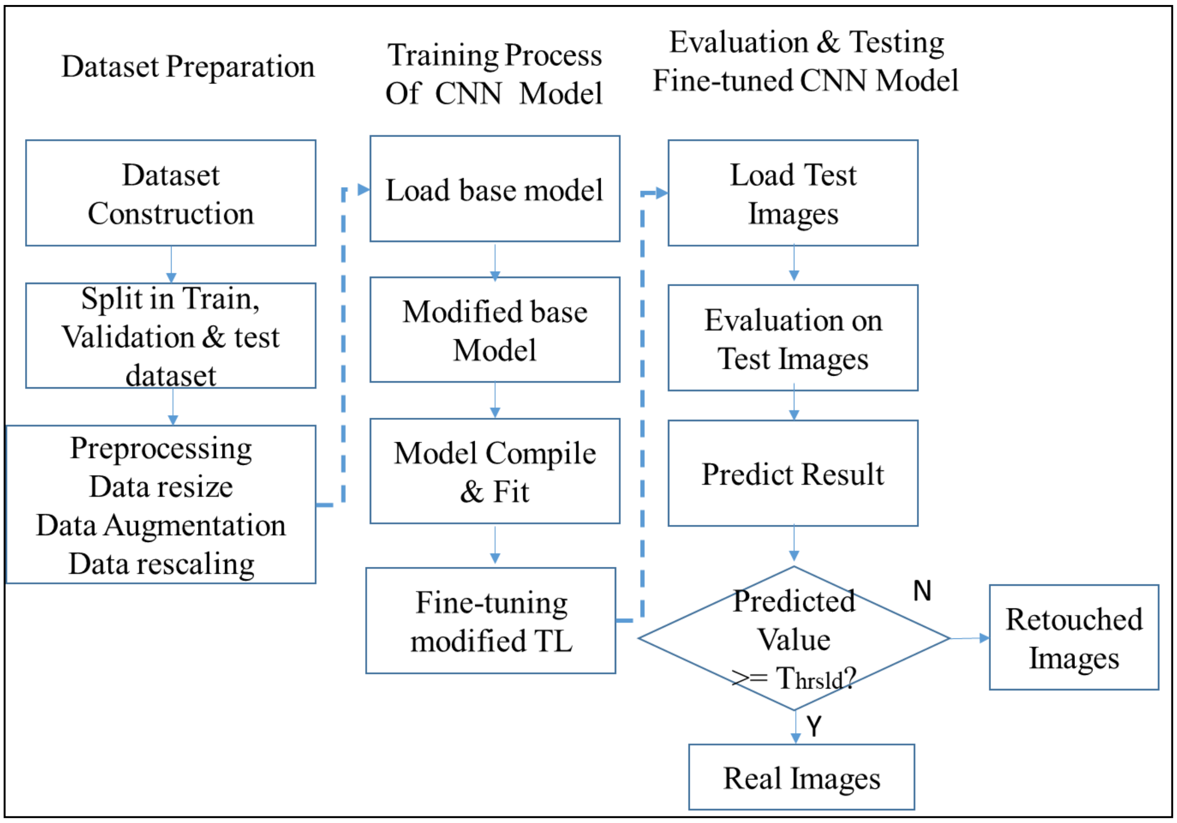
Dataset Preparation
Training Process Of CNN Model
Evaluation & Testing Fine-tuned CNN Model
Dataset Construction
Split in Train, Validation & test dataset
Preprocessing Data resize Data Augmentation Data rescaling
Load base model
Modified base Model
Model Compile & Fit
Fine-tuning modified TL
Load Test Images
Evaluation on Test Images
Predict Result
Predicted Value >= Thrsld?
N
Y
Retouched Images
Real Images

FIGURE 4.1: Flow diagram for detection and classification of retouched images using Fine-tuned pre-trained CNNs

The steps for proposed methodology is shown in fig. 4.1. This approach enables the utilization of knowledge acquired by the pre-trained model on a sizable dataset and its adaptation to the particular task at hand. the steps involved in this process are as follows:

1. **Dataset Preparation:** The standard dataset ND-IIITD retouched dataset is divided into training, validation, and test subsets. To mitigate the risk of overfitting, data augmentation techniques are applied, including horizontal flipping and a 0.20-degree rotation, generating artificially enhanced yet authentic images during training

phase.

2. **Pre-processing on images**: VGG16 and ResNet50 accepts images of size 224x224. Hence, data transformation is done over the training and validation dataset to rescale and fit the images as per the pre-trained models. The images are then divided into batch of 32.
3. **Modify the Base model**: VGG16/ResNet50 pre-trained model is loaded using a deep learning framework such as Keras or TensorFlow. The top layers of the pre-trained models are removed and one FC layer is build up by adding Global average pooling, a dropout and a dense layer.
4. **Compile & Fit**: The new model is trained using training set and evaluate its performance on the validation set. The convolution layers of pre-trained model are freeze during first training and only custom added FC layer will learn the feature maps from the given dataset. The Adam optimizer is used during this training phase rather than the SGD algorithm.
5. **Fine-tune the new model**: Since the VGG16/ResNet50 model was pre-trained on a large dataset, it already has learned many features that can be used for image classification. However, we fine-tune the model by unfreezing some top convolution layers and re-train the new model. Hence, the weight of Custom FC layers and unfreeze layers are updated during fine- tuning, which improves the performance of the model. During this training, either Adam or RMSprop optimizer is used and the performance is compared based on the selected optimizer.
6. **Test the new model**: The accuracy of the model is tested on the test dataset. the class of the images of test dataset are predicted and normalized the values of prediction near to one of the values 0 or 1 using sigmoid function. As it is binary classification, threshold is defined to determine the predicted values as either 0 or 1.

4.3 VGG16 model architecture

VGG-16 is a network of 16 levels that was proposed by the Visual Geometric Group, an organisation based at Oxford University. These 16 Covolution layers do contain the trainable parameters.There are further layers, such as the Max pool layer, but they do not contain any trainable parameters. This architecture, created by Simonyan and Zisserman, came in first place in the 2014 Visual Recognition Challenge, also known as ILSVRC-2014.

The top-1 error and top-5 error achieved by the VGG16 model when trained on an ImagNet-large scale dataset, is 28.1% and 9.3% respectively[21].

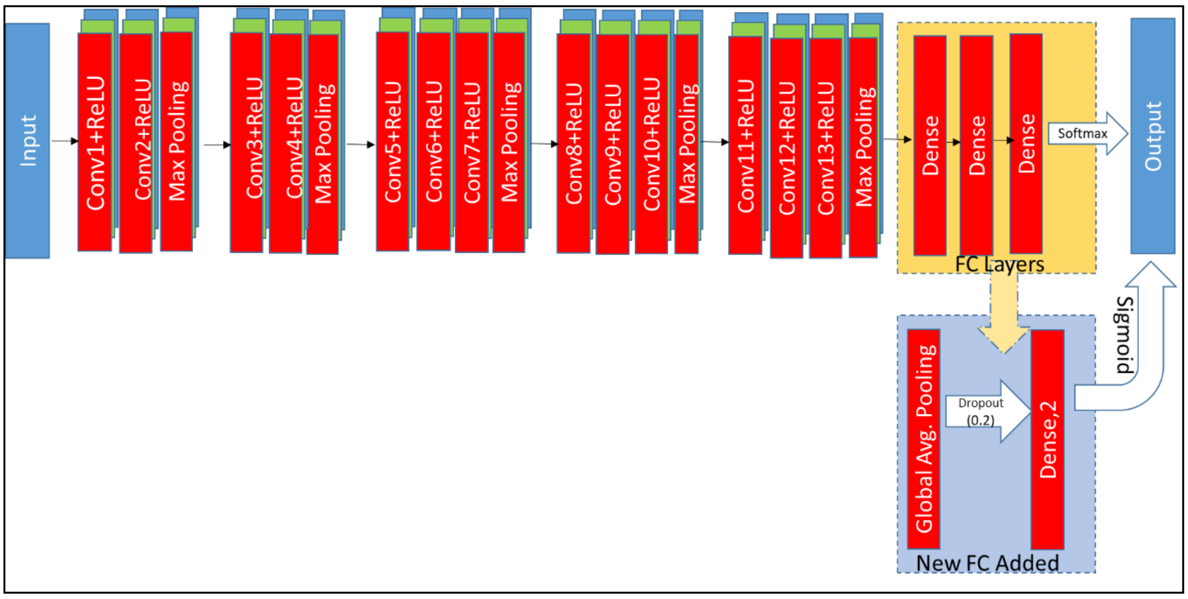

FIGURE 4.2: Architecture of VGG16 ,a CNN model. The FC layers of base model (highlighted by yellow rectangle). Modified VGG16 model (highlighted by blue rectangle)

The key characteristic of VGG16 is its use of a series of small 3x3 convolutional filters stacked on top of each other, which allows the network to learn hierarchical features at different levels of abstraction. The architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The hierarchical structure of VGG16 model is shown in below table 3. The layers are organized into blocks, and the network architecture can be summarized as follows:

- **Input Layer:** The network takes an input image with a fixed size (e.g., 224x224 pixels).
- **Convolutional Blocks:** The network consists of five sets of convolutional layers, each followed by a max pooling layer. Each convolutional layer applies a 3x3 filter and stride of 1 to the input feature maps, followed by a ReLU activation function to introduce non-linearity. The number of filters in each convolutional layer increases as we go deeper into the network.
- **Max Pooling Layers:** After each set of convolutional layers, a max pooling layer with a 2x2 window and a stride of 2 is applied to reduce the spatial dimensions of the feature maps and help in capturing more robust features.
- **Fully Connected (FC) Layers:** After the last max pooling layer, the network has

three fully connected layers. The fully connected layers act as a classifier, taking the high-level features learned by the convolutional layers and transforming them into class probabilities. The last fully connected layer has units equal to the number of classes which are 1000 in ImageNet dataset and uses the softmax activation function to produce class probabilities.

4.3.1 Modified VGG16 Architecture

In the modification of the VGG16 model, several key changes were made to adapt it for a specific classification task, as shown in fig 4.2. Firstly, the top layers of the model, which typically consist of fully connected layers, were removed. This step is often taken to prepare the model for custom classification. Next, a single fully connected (FC) layer was introduced to the architecture. This added layer helps in learning task-specific features from the extracted representations. In addition to the FC layer, global average pooling was incorporated into the network. Global average pooling reduces the spatial dimensions of the feature maps, which can enhance model generalization and reduce overfitting. To further prevent overfitting, a dropout rate of 0.2 was applied after the global average pooling layer. Dropout randomly deactivates a portion of neurons during training, promoting better generalization. Finally, the newly introduced FC layer was configured with 2 output neurons, aligning it with the binary classification nature of the task, indicating two distinct classes or categories. This modification ensures that the model's final output is compatible with the specific classification problem at hand.

4.4 ResNet50 Model Architecture

ResNet-50 is a deep convolutional neural network architecture that belongs to the family of Residual Networks (ResNets) [35]known for its exceptional performance in image classification and various computer vision tasks, primarily due to its ability to train very deep neural networks effectively. The ResNet-50 model architecture involves following layers.

1. Input Layer: The model takes an input image of size typically 224x224 pixels with three color channels (RGB).

2. Convolutional Layers: The initial convolutional layer performs a 7x7 convolution with 64 filters, followed by a 3x3 max-pooling layer. This stage helps extract basic features from the input image.
3. Residual Blocks: The core innovation of ResNet is the residual block. Residual blocks enable training of extremely deep networks without the vanishing gradient problem. Each residual block consists of several convolutional layers, with shortcut connections that skip one or more layers. The residual blocks come in different variations, but in ResNet-50, there are four types:
4. A bottleneck layer with 1x1, 3x3, and 1x1 convolutional layers, which reduces computational complexity.
5. The number of filters in each block gradually increases, creating a pyramid-like structure. The convolution layer in each bottleneck layer with filter size is shown in table 3.
6. Global Average Pooling (GAP): After the convolutional layers, ResNet-50 uses global average pooling to reduce the spatial dimensions of the feature maps to a 1x1 size. This operation computes the average value of each feature map, producing a fixed-size vector regardless of the input size.
7. Fully Connected Layer: A final fully connected layer with 1000 neurons (for the original ImageNet challenge with 1000 classes) is added for classification. The softmax activation function is typically applied to generate class probabilities.
8. Output Layer: The output layer produces the final classification predictions. Which is either 0 (fake/retouched) or 1(real) for facial retouching task.

4.4.1 Modified ResNet50

In the modification made to the ResNet50 model, the top layers, including the fully connected layers, have been removed. Instead, a single new fully connected (FC) layer has been introduced, serving as a bridge between the existing architecture and the final classification layer. This new FC layer is followed by global average pooling, which helps in reducing spatial dimensions while preserving essential feature information. Additionally, a dropout layer with a dropout rate of 0.2 has been incorporated before the newly added FC layer to prevent overfitting and enhance generalization.

TABLE 4.1: Layer wise architecture of ResNet50 base model

Layer Name	Output Size	Kernel Size	Filters	Repetition(i)
Conv1	112x112	7x7	64	x 0
	3x3 Max pool, stride 2			
Conv2.i	56x56	1x1 3x3 1x1	64 64 256	x 3
Conv3.i	28x28	1x1 3x3 1x1	128 128 512	x 3
Conv4.i	14x14	1x1 3x3 1x1	256 256 1024	x 4
Conv5.i	7x7	1x1 3x3 1x1	512 512 2048	x 6
	Avg. Pooling 1x1x2048			
Dense	1x1x1000			

4.5 Fine-tuned proposed CNN models

Fine-tuning in transfer learning is a technique commonly used in machine learning and deep learning, particularly in the context of neural networks and natural language processing. Fine-tuning is widely used in various applications, including image classification, object detection, text classification, sentiment analysis, machine translation etc. Although wonderful, pre-trained language models are not by nature task-specific. These general-purpose models are modified through fine-tuning to carry out specific tasks more precisely and successfully[36]. The pre-trained model is needed to fine-tuned to comprehend the intricacies of that unique job and domain when we come across a particular classification task, such as retouching detection. It involves modifying the pre-trained model by updating its weights through additional training on your target task. This process helps the model to learn task-specific patterns and information. During fine-tuning, a few layers are replaced or added at the end of the pre-trained model to match the number of classes or the specific requirements of the task. Fine-tuning offers several advantages, such as faster convergence, lower data requirements, and better performance. Since the model has already learned general features and representations from the pre-training, it often requires fewer training examples to adapt to the new task. This is especially valuable when working with limited data or computational resources[37].

VGG16 and ResNet50 models are pre-trained on different large scale datasets. Hence, based on the training over specific dataset, the models will learn the new features for classification task. Out of all weights (i.e. the large publically available datasets), ImageNet weight of the models is selected in this research for retouching classification.

.

Before fine-tuning, the CNN model has a fixed set of parameters. These parameters are learned from ImageNet data and are generalized to capture various low-level and high-level features in images. During initial training of the new CNN model, all convolution layers except FC layers are freeze. Hence, the weight of only newly added FC layer is updated. During fine tuning, few convolution layers of the new CNN model are made unfreeze and the weight of FC layers including unfreeze layers is updated. This process will allow more parameters to be trained for and learned new features for classification task. A summary of trainable and non-trainable parameters for modified VGG16 and ResNet50 model is given in Table 4.2. The table reflects out of total parameters of the respective model, only few convolution layers are involved in weight updating task. And these are the parameters which are highly involved for reducing the training loss of the model.

TABLE 4.2: Summary of parameters which are being updated during training process of the models

Model	**Trainable parameters**		**Non-trainable parameters**		**Total Parameters**
	Before fine-tune	**After fine-tune**	**Before fine-tune**	**After fine-tune**	
Vgg16	513	7079937	14714688	7635264	14715201
ResNet50	2049	19454977	2049	19454977	23589761

4.6 Experimental Setup

The proposed methods were implemented using the Python programming language within the Google Colab environment, leveraging GPU runtime capabilities. These implementations were applied to two datasets, namely the ND-IIITD and MDRF retouched faces datasets (Dataset 1 & Dataset2 respectively), utilizing the TensorFlow-Keras framework and other freely available libraries like pandas, matplotlib, cv2, numpy, seaborn, OpenCV etc. A total of 16 experiments were conducted for each dataset, and the ideal model was selected based on its performance with the most suitable split ratio for the retouching classification task. The hyper parameters are set as per Table 4.3.

For the initial training, the Adam optimizer is employed. This optimizer is chosen for its efficiency in optimizing the model's weights during the initial phase. The Hyperparameters like batch size, learning rate and epoch are set to 32, 0.001 and 10 respectively. During fine-tuning, two different optimizers, Adam and RMSprop, are used one by one for each model along with 32 image size, learning rate set to be 0.0001 and epoch 20(for VGG16) and 10 (foe ResNet50). The Adam optimizer is applied to fine-tune the model's weights. It is known for its adaptive learning rates and is effective in addressing the nuances of the training data, potentially leading to better convergence during fine-tuning[38]. The RMSprop optimizer is also employed during fine-tuning as an alternative to Adam. It is adaptive like Adam but employs a slightly different update rule for the learning rates. This choice allows for a comparison of the optimizers' performance during fine-tuning.

The effect of using different optimizers during fine-tuning is analyzed. This analysis helps us understand how the choice of optimizer impacts the model's performance during this phase. Specifically, it evaluates how the loss and training accuracy of the model change when using Adam or RMSprop for fine-tuning. This comparison is essential to identify which optimizer is more effective in updating the weights of both the unfrozen convolutional layers and the newly added fully connected layers during fine-tuning.

TABLE 4.3: Summary of Hyper Parameters which are set during learning phase of the proposed TL models

TL Model Training Mode	**VGG16/ResNet50 Parameters**	**Value**
Initial Training	Batch Size	32
	Image size	224x224
	LR(Learning Rate)	0.001
	Loss	Cross Entropy
	Epoch	10
	Optimizer	Adam
	β_1	0.9
	β_2	0.999
Fine-Tuning	Batch Size	32
	Image size	224x224
	LR(Learning Rate)	0.0001
	Loss	Cross Entropy
	Epoch	10(VGG16) / 10(ResNet50)
	Optimizer	Adam / RMSprop
	Momentum	0

4.7 Evaluation Matric

A confusion matrix is a fundamental tool in image classification and machine learning in general. It provides a clear and concise summary of how well a classification model is performing. In image classification, it helps us understand how many images were correctly classified into their respective categories and how many were misclassified. The confusion matrix is typically organized into a grid, where the rows represent the true classes, and the columns represent the predicted classes[39]. The confusion metrics shows the value of TP (true Positive), True Negative(TN), FP(False Positive), FN(False Negative), as shown in fig 4.3. These values allow us to calculate various performance metrics, such as accuracy, precision, recall, and F1-score, which are essential for assessing the effectiveness of an image classification model[40]. By examining the elements of the confusion matrix, we can gain insights into the model's strengths and weaknesses, making it a critical tool for model evaluation and improvement.

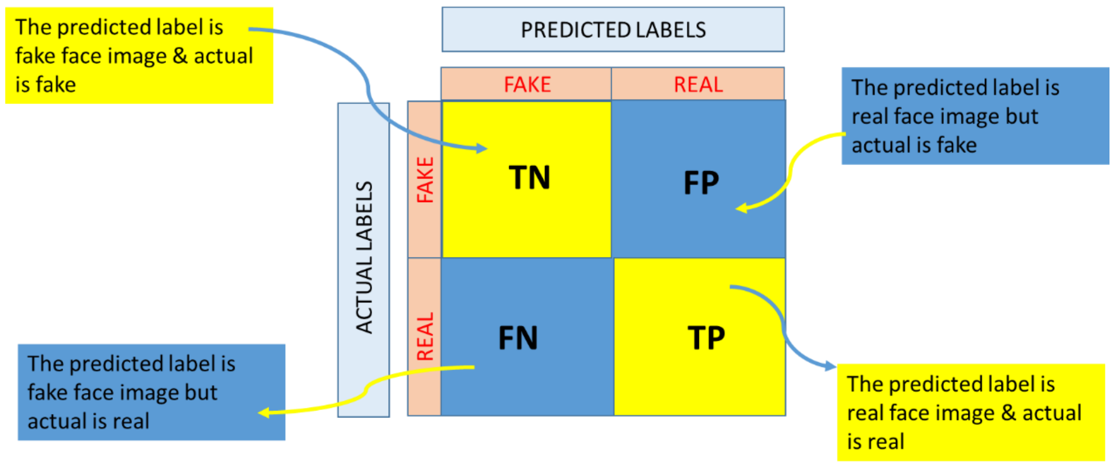
The predicted label is fake face image & actual is fake
PREDICTED LABELS
FAKE
REAL
The predicted label is real face image but actual is fake
ACTUAL LABELS
FAKE
REAL
TN
FP
FN
TP
The predicted label is fake face image but actual is real
The predicted label is real face image & actual is real

FIGURE 4.3: Understanding of Confusion Matric

Precision, is number of correctly identified real images from all images identified as real.

$$Precision = \frac{x}{x+y} \tag{4.1}$$

Recall, is number of correctly identified real images from all actual real images.

$$Recall = \frac{x}{x+z} \tag{4.2}$$

Here, x= $true\ negative$, y = $true\ negative$, z = $false\ positive$

F1-score, is average value of precision and recall.

$$F1\_score = \frac{2 * (P * R)}{P + R} \tag{4.3}$$

Acc(Accuracy), is the ratio of correctly identified samples to the total predicted samples.

$$Acc = \frac{Correctly\ Identified\ images}{Total\ Predicted\ images} \tag{4.4}$$

Receiver Operating Characteristic Curve (ROC), is a graph that displays how well a classification model performs across both classes. It gives the relation between TPR (True Positive Rate) and FPR (False Positive Rate).

$$\text{TPR} = \frac{x}{x+z} \tag{4.5}$$

$$\text{FPR} = \frac{z}{\text{y}+z} \tag{4.6}$$

4.8 **Summary**

This chapter provides an in-depth understanding of the VGG16 and ResNet50 architectures, their modifications for specific tasks, and the powerful technique of fine-tuning for transfer learning. These concepts are essential in deep learning and computer vision, enabling us to harness the strengths of pre-trained models while tailoring them to specific applications. The hyper parameters set considered during training process are briefly explained based on the our classification task and literature reviews. Hence, we can make informed decisions about which optimizer to use during the fine-tuning phase, considering its impact on loss reduction and training accuracy, and choose the most effective approach for adapting the pre-trained models (VGG16 and ResNet50) to the target task.