# CHAPTER 3

# FINETUNING PRETRAINED DL MODELS FOR DR STAGES CLASSIFICATION

TL has gained much interest in the last ten years as a tool to boost the performance of ML and DL models. This process involves reusing a pretrained model from a specific task and fine-tuning it for a related but different application. TL provides multiple benefits compared to traditional ML and DL approaches, especially when labeled data is scarce or computational resources are limited [58-60]. The main advantages of TL are listed below.

## 3.1 ADVANTAGES OF TRANSFER LEARNING

### 3.1.1 Reducing the Need for Large Datasets

A significant challenge in ML and DL requires extensive labeled datasets for training accurate models. Traditional DL models believe in high-quality, large-scale datasets to perform effectively. However, compiling and annotating these datasets remains laborious, expensive, and often unrealistic. TL addresses this limitation by utilizing pretrained architectures trained on massive datasets (e.g., ImageNet for image classification). By adapting pre-trained models to smaller, domain-relevant datasets, TL achieves strong performance with minimal labeled data. This makes it an ideal approach for medical applications, where obtaining large annotated datasets is challenging. [61].

### 3.1.2 Faster Convergence and Lower Computational Costs

Training DL models from scratch requires significant processing power, with training duration ranging from days to weeks depending on the model's complexity. TL however, provides a shortcut by utilizing knowledge from a pretrained model. This significantly accelerates model training and convergence, as the model has already captured relevant patterns from a similar domain. Fine-tuning a pretrained model requires fewer training epochs and significantly less computational power, making it a cost-effective and efficient alternative to train a model from scratch [62].

### 3.1.3 Improved Generalization

TL improves generalization performance on novel data instances. In traditional ML and DL, a model might overfit to the key aspects of the training dataset, restricting its

ability to perform well on unfamiliar data. TL overcomes this limitation by using a pre-trained model containing learned features from a broad data set. These features are adapted to the new task, leading to better generalization. For example, in image classification, pre-trained CNN architectures such as VGG, ResNet and EfficientNet have already learned to detect edges, textures, and common patterns across many image datasets. This allows them to generalize well to new tasks, even those with different domains [63].

### 3.1.4 Reusability of Knowledge

TL makes it possible to reuse knowledge across various tasks and domains. Pre-trained models, specifically models that learn from broad and varied datasets, capture practical knowledge transferred to different domains with minimal modification. This contrasts with traditional ML and DL models, where each model is typically trained from scratch for a specific task. To illustrate, an architecture trained for image classification is adapted for medical image diagnosis, enabling the re-use of learned features. It allows efficient use of time and resources while stimulating innovation by applying established models to emerging tasks.

### 3.1.5 Performance Boost in Specific Domains

TL provides a substantial performance boost in areas like medical imaging, for which there are generally few labeled samples available and expensive to obtain. Models trained on broad and varied datasets are adapted for specialized tasks within a given domain, like DR and retinal disease detection.

The transfer of knowledge from general-purpose models enables them to adapt to the specific features of medical images, leading to high accuracy even with smaller, less labeled datasets. In comparison, traditional ML and DL models, which do not benefit from this prior knowledge, often require wider data collection and may underperform relative to alternative approaches.

### 3.1.6 Reducing the Risk of Overfitting

In ML and DL, overfitting regularly impacts model generalization, especially when models are trained on small datasets. Overfitting occurs when the model captures irrelevant variations instead of meaningful patterns or patterns in the training data that do not generalize to new inputs. By initializing with a pre-trained model that encodes generic features, TL reduces the possibility of model overfitting. Fine-tuning such a model on a

smaller dataset allows it to adapt while preserving the generalization capabilities learnt from the original, larger dataset. As a result, the overfitting decreases, and the network performs well on unfamiliar inputs [64].

This capability confirms that TL bridges the gap between data limitations and the need for high-performance solutions, making it a viable and efficient strategy for promoting research and applications in resource-limited domains. Based on that, the fundamental aim of this chapter is to finetune the pre-trained models such as VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2L for the diagnosis of DR stages and compare the performances.

## 3.2 SIGNIFICANCE OF THE STUDY

Choosing an appropriate pretrained model is critical in DR stage classification as it influences the model's performance, accuracy and operation efficiency. Pretrained models have different architectural strengths, and analysing the performance on DR classification is useful for model selection in clinical practice. This paper systematically compares pretrained DL models, such as VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2L, to provide insight into which network is most appropriate for detecting DR stages and is computationally feasible.

This study is important as it has identified a pre-trained model that is most suitable for detecting DR using pre-trained models, balancing the accuracy of the supervised modeling techniques with their generalization and associated computational cost, while increasing the scalability of the proposed method. Through evaluating the models with different complexities and feature extractors, the work thoroughly compares each model's advantages and disadvantages, which helps practitioners choose their model and setting suitable for their practical needs in medical image classification tasks.

## 3.3 OVERVIEW OF PRETRAINED MODELS' ARCHITECTURES

### 3.3.1 Finetuned VGG16 Model

The pretrained models are selected for this study due to their proven effectiveness and unique architectural features, which make them highly suitable for image classification tasks, including DR detection. Each model brings distinct advantages relevant for evaluating the trade-offs between accuracy, computational efficiency. One of such models

is VGG16, a deep CNN (DCNN) architecture by the Visual Geometry Group (VGG) from the University of Oxford, is well-suited for image classification [65] and it is appreciated for its compact size and uniformity, based on small 3x3 filters in all convolutional layers. This architecture enables the model to learn higher-level features from images, an important aspect for complex tasks like DR classification. Progressive learning detail, abstract patterns and fine-grained details from retinal images are learned based on the depth of the model. Its DR detection results provide a baseline for comparing the influence of deeper architectures on classification accuracy.

In the first block, the architecture comprises 16 layers-13 for convolution and 3 Fully Connected (FC) Layers and operates on 224×224 RGB images. It is organised into 5 main blocks, each incorporating convolutional layers and subsequent max-pooling layers. The model is optimal since it applies 3 × 3 filters with a stride of 1 to capture detailed spatial features without being computationally expensive. The first block consists of 2 convolutional layers with 64 filters and the second block consists of 2 convolution layers with 128 filters. As the architecture moves on, the filter count is raised to 256 in the third and and 512 in the fourth, and fifth blocks, respectively. After each convolutional block, a max-pooling layer uses a 2×2 filter with a stride of 2 to reduce the spatial size of feature maps. This pooling operation assists in lowering the computational burden and makes the model resilient to minor deformations and input image variations.

The feature maps then pass through the convolutional and pooling layers before being flattened to one-dimensional vectors and extracting their features using three FC layers. The first two FC hidden layers include 4096 neurons each, and the final output layer consists of 1000 neurons, each related to a class in ImageNet. These FC layers combine input features produced by the convolutional layers to produce the ultimate classification output. Hence, ReLU activation functions are used in the hidden layers to incorporate non-linearity, which allows the network to learn complex patterns. Finally, the output layer is passing through the Softmax activation to produce probabilities across the classes.

Although VGG16 is widely recognised for its straightforward design and high effectiveness, its architecture is computationally intensive owing to the large number of parameters, approximately 138 million. This makes it less efficient for some applications with less computational power. However, VGG16 has played a critical role in developing

DL models, and its simplicity of design and performance on image classification tasks have made it a typical and influential architecture in the image processing community. Figure 3.1 is the diagram of VGG16 and its fine-tuned architecture.

Fine-tuning offers significant advantages in DL, particularly when utilising pre-trained models for new tasks. This method employs a model trained on large-scale data, such as ImageNet, with a new dataset, which may be of a smaller scale. Fine-tuning also dramatically reduces the amount and the computation resources used in training, resulting from pre-training with known weights. Building on pre-trained weights allows the model to learn task-specific patterns from the beginning, speeding up convergence and reducing the training time.

Another important advantage of fine-tuning is that it increases the model's capability to perform well on something it is not optimised for (where data is scarce). Pre-trained models already encapsulate learned generalizable feature representations such as edges, textures, and shapes that are helpful to a wide range of tasks. Once their inductive biases are adjusted to specific tasks, such fine-tuned models perform better in generalising and predicting. This is particularly useful when acquiring a large annotated corpus is difficult or impossible.

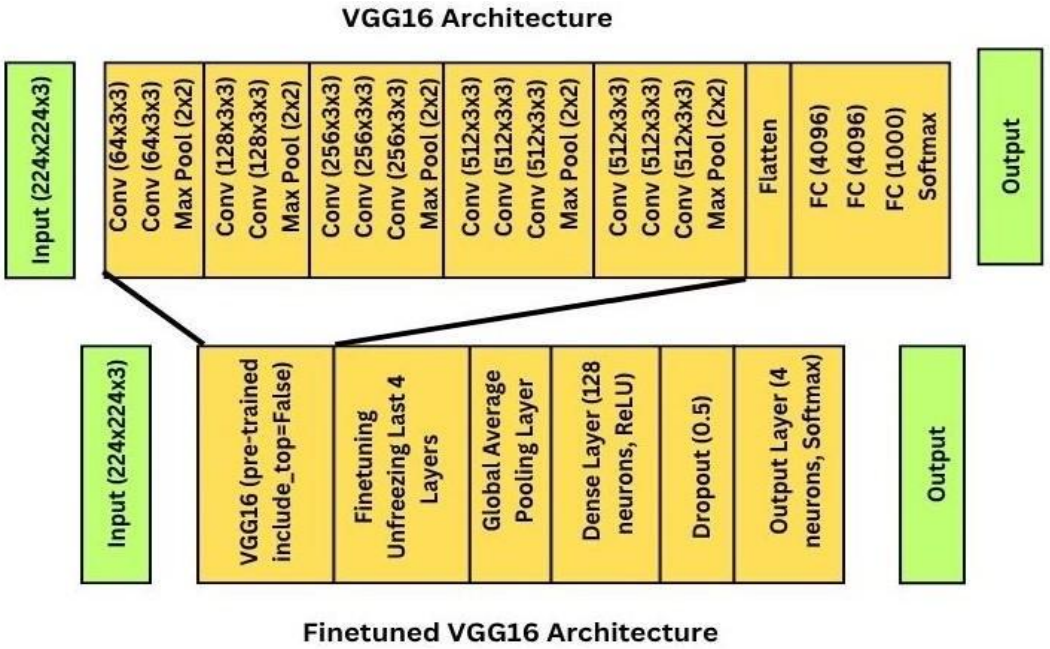


**Figure 3.1 VGG16 and Finetuned VGG16 Architecture**

Moreover, fine-tuning prevents overfitting: it locks the pretrained model's early layers in place (since these layers work as good feature extractors and adding randomness to their weights would cause the adaptive feature pooling loss to be closer to 0) and only changes the final layers.

Therefore, the fine-tuning of the VGG16 model for the four DR type classification is given in Figure 3.1. For fine-tuning, the model sets up a data pipeline for training, validating, and testing an image classification model using TensorFlow and Keras. It employs the ImageDataGenerator with VGG16's preprocess input function for standardising input images. The dataset is structured into three directories: train, validation, and test, with images resized to 224x224 pixels and a batch size of 8. The class_mode='sparse' assures integer label encoding, and shuffling is applied only to the training set. This preprocessing pipeline is important for feeding properly formatted data into a DL model. Then, the model loads the pre-trained VGG16 model, excluding the FC layers (include_top=False), which are typically used for ImageNet classification.

The model begins with ImageNet pre-trained weights (weights='imagenet'), with the input dimensions (224, 224, 3) to match the image format. The model retains its convolutional base by excluding the FC layers, which have learned valuable low- and mid-level features from the large ImageNet dataset. When the pre-trained VGG16 base model is loaded, the convolutional layers are frozen by setting the trainable parameter to False. Freezing the base model is a standard TL strategy that preserves previously learned representations while training the newly added layers. This validates that the convolutional layers that have already learned good features, such as edges and textures, are kept, which are reused for the new task. The newly added layers are responsible for learning more specific patterns regarding the new dataset and the classification task. Initially, all convolutional layers stay frozen to preserve their pretrained weights and maintain the learned feature representations. Later, the last four layers are unfrozen, enabling the model to refine its weights specifically for DR dataset while still utilising the robust feature extraction capabilities of the pretrained network.

The model is then integrated with layers: a Global Average Pooling (GAP) 2D layer, a dense FC layer with 128 neurons, a dropout layer, and a final dense output layer with four neurons corresponding to the four classes. The GAP layer computes the mean of each feature map and thus compresses the spatial dimensions, resulting in a more compact

and reduced possibility of overfitting. The dense layer with ReLU activation is tasked with adding non-linearity in the model to make it capable of learning intricate and abstract patterns of input data. Deployment of dropout at a rate of 0.5 to improve generalisation and reduce overfitting through the random disabling of 50% of the neurons in each training step. The last output layer utilizes the softmax activation, transforming the network outputs into a probability distribution over the target classes of four distinct types, thus allowing proper classification.

### 3.3.2 Finetuned ResNet-50 Model

The DCNN architecture called ResNet-50 is one of the ResNet (Residual Networks) family proposed by Microsoft Research [66]. It introduces residual connections and is a structurally deeper architecture, enabling saddle point minimization by jumping over sub-optimal local minima because of summing the input with output and promoting enhanced gradient flow for much deeper network learning. This method solves the vanishing gradient problem, which is in favour of DNN's efficient training. It is much deeper than VGG dense in the traditional/recognition model and performs well on image classification/recognition.

ResNet-50 architecture begins with an initial convolutional layer and is followed by a chain of Residual blocks. Every block has greater than one convolutional layer and the block output taken by the sum of the input and the block output of the preceding convolutional computation. This is so because of the residual connection that ensures that the gradients can propagate more effectively during backpropagation, which is difficult when deeper networks are used. In ResNet-50, the convolutional layers in bottleneck blocks first use a 1×1 convolution for dimension reduction of the input, a 3×3 convolution for feature extraction, and a 1×1 convolution to return the input dimension. This network structure decreases computation overhead and retains the model's capability to acquire complex patterns.

After residual blocks, ResNet-50 uses the spatial size of the feature maps and multiplies it with GAP to obtain a dense representation. This is then linked to an FC layer, which generates the ultimate result. The output layer is equipped with softmax as an activation function for classification, which returns a probability distribution over the target classes. BN is applied after each convolution to facilitate the training, speed up the convergence, and enhance generalisation performance.

ResNet-50 has been an influential model in DL due to its ability to render it possible to train deep networks without suffering from degradation. The architecture of ResNet-50 and fine-tuned ResNet-50 is shown in Figure 3.2

For the fine-tuning process, the dataset is loaded using the ImageDataGenerator class from TensorFlow, with preprocessing tailored for ResNet-50. The preprocessing function preprocess_input confirms that images are appropriately scaled and normalised. The dataset is divided into training, validation, and test sets, each resized to (224, 224) pixels and loaded in batches of 8 images. The train generator applies shuffling to enhance model generalisation, while val_generator and test generator do not shuffle to maintain evaluation consistency.

The dataset is classified using a sparse categorical format, appropriate for tasks involving multiple distinct classes, and then utilises the ResNet-50 model, pre-trained on the ImageNet dataset. The ResNet-50 model begins with pre-trained weights and is configured using the include_top=False parameter to exclude the original FC layers designed for ImageNet classification.

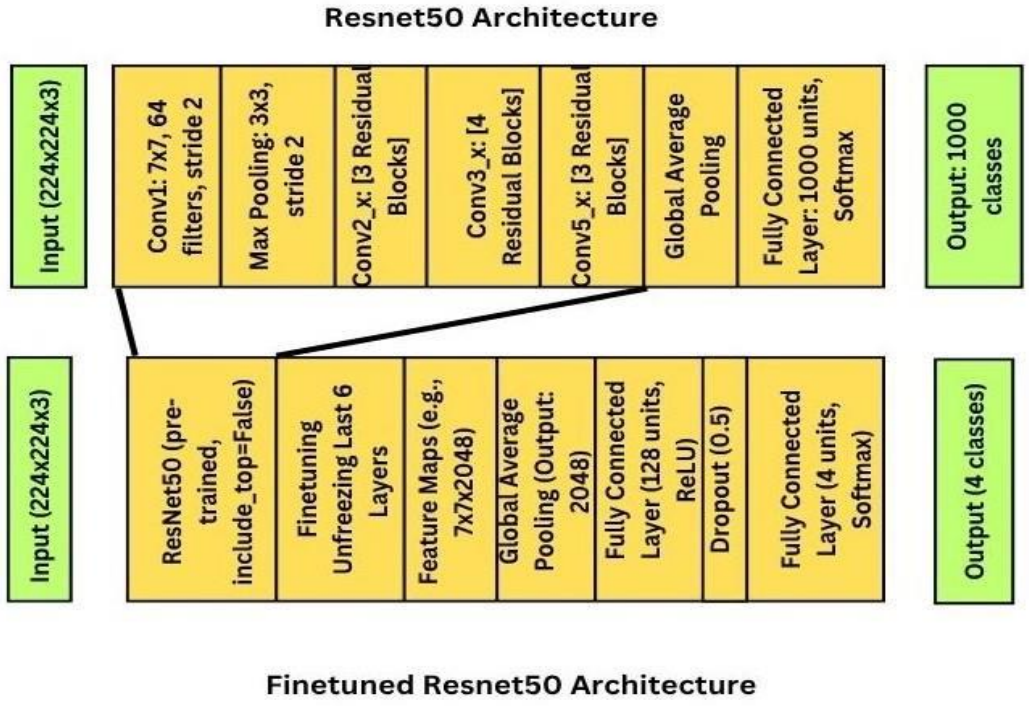


**Figure 3.2 ResNet-50 and Finetuned ResNet-50 Architecture**

This design retains only the convolutional layers, which are essential in capturing features in the input images, and it allows the model to reuse previously learned representations of visual features.

The input shape is (flower_width, flower_height, 3), like the format of retinal fundus images for DR detection. During this fine-tuning, the ResNet-50 base model is set to non-trainable (base_model. trainable = False). This prevents updating the weights during training of the convolutional layer. This approach allows the model to keep the features it learned from the ImageNet dataset, which include low-level visual patterns (edges and textures) and high-level features.

Initially, all the convolution layers are frozen to keep the pretrained weights and the learned feature representation. Then, the last six layers are unfrozen, allowing the model to update its weights specifically for DR data, while benefiting from the strong feature extraction power of the trained network.

After the base model, a GAP layer is used to average a pool of feature maps into a single average value to shrink the spatial dimension of the feature maps. This results in a compact 2D feature representation, enhancing computational efficiency and lowering the risk of overfitting. This pooled output is then fed into a dense layer of 128 neurons and ReLU activation such that the model learns complex feature interactions. A dropout layer with a rate of 0.5 is used to avoid overfitting, where half of the neurons are randomly disabled during training.

At last, the model concludes with an FC output layer comprising four neurons, each representing a DR stage: normal, mild, moderate, or severe. Softmax activation function is used to produce a probability distribution between these classes in a way that the model can both make class predictions and classify four classes. This is suitable configuration for multi-class scenarios such as DR classification, where distinguishing between four categories is essential.

### 3.3.3 Finetuned InceptionV3 Model

InceptionV3 is a DCNN model that has been introduced by Google to provide more accuracy to image classification outcomes at lesser computational costs [67]. The model is built on the existing print of the Inception architecture and has many critical new

contributions. The Inception module is a key building block of InceptionV3, and it consists of three parallel convolutional layers (filters of sizes 1 × 1, 3 × 3 and 5 × 5) that compute the features on multiple spatial scales. Such a multi-branch structure allows the network to learn different features more effectively, improving its ability to describe complicated patterns, and to remain computationally efficient.

The architecture contains 48 layers of convolutional, pooling, and FC layers. One of the key design patterns is down-sampling with 1×1 convolutions and then the larger convolutions to capture the higher-level features. This approach significantly reduces the parameters and improves the computational efficiency without losing precision. Additionally, BN is used after each convolutional layer to ensure the learning process remains stable, leads to fast convergence, and results in high performance of the overall model.

The model contains auxiliary classifiers between blocks, acting as a regulariser to encourage training with gradient flow. These classifiers help address the vanishing gradient problem, particularly in deeper networks, to allow stable and efficient convergence. The architecture of InceptionV3 and finetuned InceptionV3 is given in Figure 3.3.

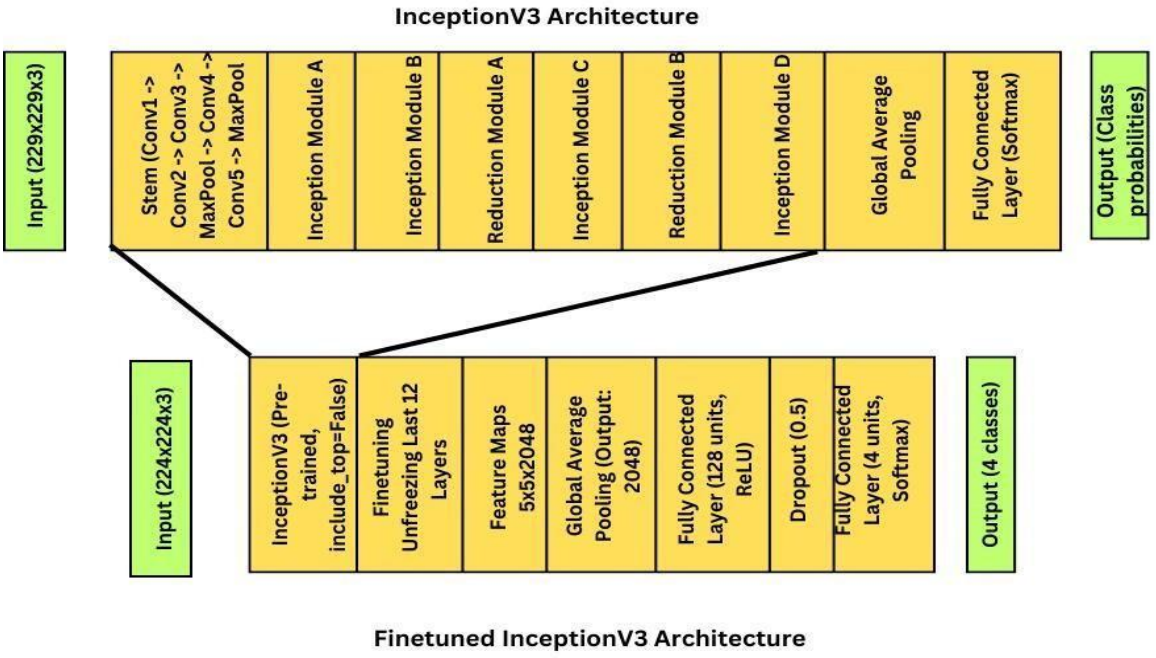


**Figure 3.3 InceptionV3 and Finetuned InceptionV3 Architecture**

The last layers of InceptionV3 consist of a GAP layer, which decreases the dimensionality of the feature maps, and an FC layer to obtain the final probability for each class. A Softmax activation function is an output stage that provides a probability distribution over target classes in the classification task.

InceptionV3 is based on previous iterations of the Inception architecture and is designed for improved computational efficiency and superior accuracy. In addition, inception modules in the fine-tuned InceptionV3 have adopted 1x1 convolutions to reduce dimensions and larger convolutions, reducing the number of parameters and thus enhancing computational efficiency.

BN is utilised following each convolutional layer, aiding in training stability and convergence acceleration by normalising activations and gradients. The second interesting aspect of InceptionV3 is the addition of auxiliary classifiers in intermediate layers to regularise the network and facilitate the gradient propagation during training, which is a way to tackle the vanishing gradient problem in deep networks. Additionally, the model replaces traditional FC layers with GAP, which decrease the number of parameters and mitigating the risk of overfitting. These enhancements make InceptionV3 more efficient and effective than its predecessors, enabling it to achieve higher accuracy while requiring fewer resources for training and inference.

Figure 3.3 illustrates the application of TL for DR classification using the InceptionV3 architecture. The dataset is prepared using ImageDataGenerator with preprocess_input from InceptionV3 to ensure proper normalization. The training, validation, and test images are resized to (224, 224) pixels and processed in batches of 8. The train_generator applies shuffling to improve model generalization, whereas val_generator and test_generator keep a fixed order for evaluation consistency.

The dataset is labelled in a sparse categorical format, generalizing it to multi-class classification tasks. The model employs InceptionV3, pretrained on the ImageNet dataset, as the base model, allowing the network to utilize previously learned feature representations such as edges, textures, and object components. The base model is set to be non-trainable by freezing its layers, ensuring that the pre-trained network weights are not updated during training. This allows the model to concentrate on learning task-specific features through the newly added layers designed for DR classification while preserving the knowledge from

the ImageNet dataset. Initially, all convolutional layers stay frozen to preserve their pretrained weights and maintain the learned feature representations. Later, the last twelve layers are unfrozen, enabling the model to refine its weights specifically for the dataset while still utilizing the robust feature extraction capabilities of the pretrained network.

The input to the model is an image of size 224x224 with three colour channels (RGB), a standard format for image classification tasks. The feature maps of the InceptionV3 model are sent through a GAP layer, which transforms the 4D feature maps into a 2D tensor by averaging the spatial dimensions. The pooling step help preserve important features and reduce the complexity amount and implement a dense layer with 128 units to pool these high-level features.

A dropout layer with a rate of 0.5 is employed to avoid the model from over-fitting; 50% of neurons are randomly turned off during the training. This regularization method enhances the model's generalization ability by avoiding the overuse of neurons. The last layer for classification has 4 units activated by the softmax, outputting a probability distribution for the 4 categories (normal, mild, moderate, and severe). The softmax function is employed to scale the output scores into the range [0, 1], which measures the probabilities of the input image belonging to each of the four classes.

TL helps the model to use pre-learned features from the large and diverse dataset to accurately categorize the DR stages even when trained on relatively small and domain-specific datasets. This method of combining a finetuned pretrained model like InceptionV3 with custom layers for classification is particularly well-suited for tasks such as DR detection, where labelled data is limited. The approach enables the efficient training of a DL model that generalizes well to unseen data, improving the reliability of DR classification in clinical applications.

### 3.3.4 Finetuned Densenet121 Model

DenseNet-121 is a family of DenseNet, a DCNN, developed to improve feature propagation and minimize the number of the network parameters without compromising on high performance in image-based applications [68]. The key feature of a DenseNet is employing dense connections so that every layer can access the previous layers' feature maps for all activations, as opposed to the conventional convolutional models in which one layer is linked to the subsequent layer of the network. This topology, namely densely

connected architecture, enhances the use of features and solve the vanishing gradient problem, which enables the network to capture more complicated patterns using fewer parameters. Thus, DenseNet-121 is an ideal trade-off between computational efficiency and classification performance.

In DenseNet-121, the architecture contains four dense blocks, and each dense block is composed of multiple convolutional layers, where each layer concatenates the output of all previous layers in the same block as input. These dense blocks are succeeded by transition layers commonly containing BN, 1×1 convolution, and a 2×2 average pooling to down-sample spatial size and adjust the network complexity.

This kind of high connectivity allows the deep network to use features in the previous layers at all network levels, improve the backprop flow, alleviate the vanishing problem for gradients, and encourage feature reuse.

After every dense block, the transition layers are employed to execute some operations like down-sampling and mitigating the dimensionality of the feature maps, to process feature maps efficiently with minimal computational cost. The architecture of DenseNet-121 and finetuned DenseNet-121 is shown in Figure 3.4.

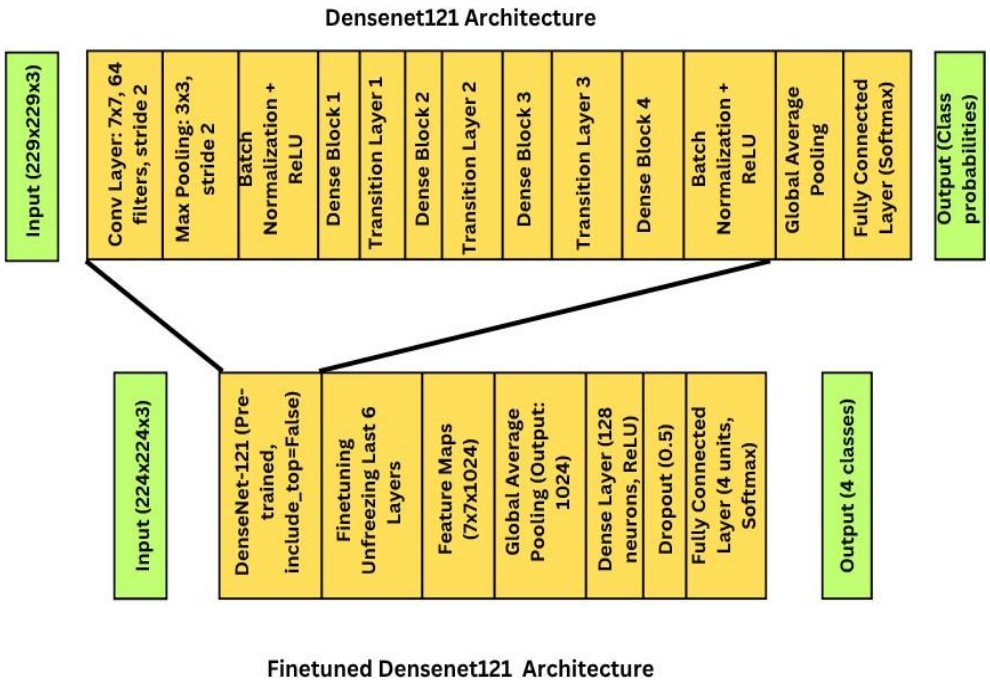


**Figure 3.4 DenseNet121 and Finetuned DenseNet121 Architecture**

A significant advantage of DenseNet-121 is its smaller number of parameters than other deep networks of similar depth. The dense connections minimise the feature redundancy between layers so that DenseNet-121 achieve equivalent or better accuracy with fewer parameters. This results in a computationally more efficient model, which is of great value when resources are scarce or when handling so-called big data.

DenseNet-121 works effectively on most image classification problems, such as medical image analysis, that requires the capture of delicate structures or details. Because of the capacity of the architecture to retain spatial information and still perform effective feature extraction, the architecture is used with high frequency for tasks as disease detection, segmentation, and classification. In short, DenseNet-121 is an efficient and practical approach to DL problems by meeting their two core requirements: high accuracy while low computational intensity.

DenseNet-121 is a specific version of the DenseNet architecture, distinguished by its relatively smaller depth than other versions like DenseNet-169, DenseNet-201, or DenseNet-264. The number "121" indicates the overall number of layers in the network, and it offers several advantages over its deeper counterparts, making it suitable for different use cases.

The dataset is preprocessed using preprocess_input from DenseNet, ensuring appropriate normalization for the model. Images are resized to (224, 224) and loaded in batches of 8 using ImageDataGenerator. The training set undergoes shuffling to enhance model generalization, while the validation and test sets hold a fixed order for uniform evaluation. The dataset is structured in a sparse categorical format, making it compatible with multi-class classification tasks. The model loads DenseNet121, pre-trained on the ImageNet dataset, for feature extraction, utilizing its ability to reuse features across layers through dense connections.

The pre-trained network weights remain fixed throughout training by freezing the base model (setting its layers to non-trainable). This allows the model to concentrate on learning features specific to the DR classification while benefiting from the generalized features learned on the large ImageNet dataset. Initially, all convolutional layers stay frozen to preserve their pretrained weights and maintain the learned feature representations. Later, the last sixteen layers are unfrozen, enabling the model to refine its weights

specifically for our dataset while still utilizing the robust feature extraction capabilities of the pretrained network.

The model receives an input of dimensions 224x224 pixels and three colour channels (RGB). After processing through the DenseNet121 base, the output feature maps are passed through a GAP layer. This layer averages the feature maps to decrease their spatial dimensions, producing a 2D tensor that preserves critical features while lowering computational complexity. Lastly by adding one dense FC layer of 128 units with ReLU activation to bring non-linearity in such a way that the model learns complex patterns corresponding to DR stages.

Following the dense layer is a 0.5 dropout layer to prevent overfitting and enhance generalization. It is a method of turning off 50% of the neurons at random during training, thus forcing the network to learn more generalized and stable feature representations. The final output layer has four units, which are DR stages: normal, mild, moderate, and severe. At this point, softmax activation is used to normalize raw output to probability distributions such that each class is calculated by the model and a final classification is performed. This TL implementation using DenseNet-121 for DR classification is an effective solution, as it leverages a pretrained model and applies on top specialised layers for its task. Adopting TL helps the model to take advantage of the abundant feature representations learned from a large dataset, and fine-tuned layers can accurately classify different DR stages with limited data. Therefore, the model has the capability of classifying DR stages with high accuracy which implies its applicability in various real-world scenarios such as medical imaging and healthcare.

### 3.3.5 Finetuned EfficientnetV2L Model

EfficientNetV2-L is based on the original EfficientNet architecture, which applies compound scaling to network depth, width, and resolution dimensions [69]. In this case, instead of scaling these dimensions independently, EfficientNetV2-L introduces a more retrofit scaling method to utilise the computational resource better. The proposed model incorporates advanced operations, such as the lightweight convolutions (i.e., depthwise separable convolutions), which decrease computational complexity but have the same high-performance level. Moreover, EfficientNetV2-L further enhances the network design with efficient swish activation functions, enhanced training procedures, and stronger regularisation methods.

A key innovation of EfficientNetV2 is the use of progressive learning. The strategy integrates network pruning and iterative image resolutions to achieve a trade-off

between accuracy and training speed. Thus, employing features for identification which further enhances this performance through enabling the network to converge and minimize the time taken for training.

This makes EfficientNetV2-L an attractive option for large-scale image classification CNNs, as it performs similarly to the other large networks with less computation cost. EfficientNetV2-L obtains the superior performance on benchmark evaluations across various image classification datasets. This has made it particularly appropriate for practical applications considering precision and efficiency, such as medical image diagnosis, self-driving, and large–scale image recognition.

EfficientNetV2-L is superior to previous models, such as EfficientNetV1 and the smaller versions of EfficientNetV2, in terms of training cost and the "L" in EfficientNetV2-L is the larger model designed to tackle more complex tasks on even larger data with higher capacity. Although the fundamental ideas of EfficientNet, like compound scaling that scales the depth/width/resolution at identical ratios, are preserved, EfficientNetV2-L exhibits significant improvements from other versions. One key advance of EfficientNetV2-L is the fine-tuning of the scaling method. A key difference between EfficientNetV2-L and the original EfficientNet is that, other than the compound scaling to adjust depth, width, and resolution simultaneously and it introduces a more advanced scaling technique. The present approach is expected to bring a balance between the computational cost and the prediction accuracy towards a more optimal level, and it achieves a moderate level of monitoring precision with little computational cost.

Trainability is also a primary advantage of EfficientNetV2-L, as it allows faster training than the proposed architecture/training time. It utilises techniques such as progressive image resolution and Neural Architecture Search (NAS) that aid it in learning faster. By training with increasingly higher image resolutions while prioritising optimisation of specific components of the model architecture, EfficientNetV2-L trains much more efficiently than previous models. It is also superior to its predecessors and other DL architectures (e.g., ResNet and DenseNet) concerning performance on multiple image classification tasks. It has better accuracy on standard datasets and less computation and

memory consumption. The architecture of EfficientNetV2-L and finetuned EfficientNetV2-L is shown in Figure 3.5.

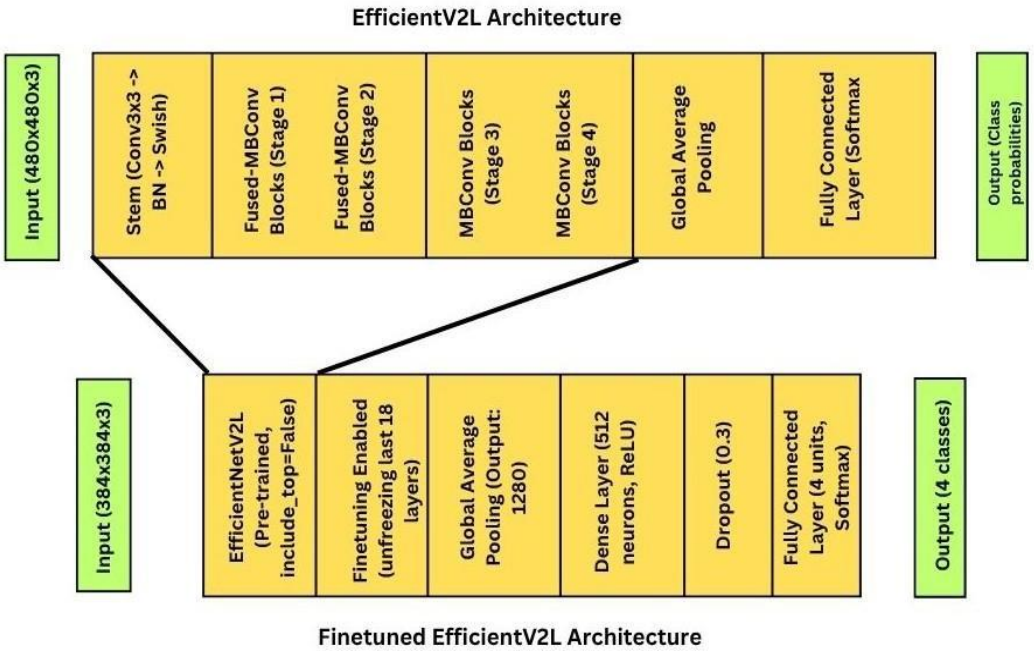


**Figure 3.5 EfficientNetV2-L and Finetuned EfficientNetV2-L Architecture**

In the first architecture, EfficientNetV2-L, pre-trained on the ImageNet dataset which is used for feature extraction and in the second fine-tuned architecture, where the model loads and preprocesses image data for training, validation, and testing using TensorFlow's ImageDataGenerator. It normalizes pixel values by rescaling them from [0,255] to [0,1] and loads images from specified directories, resizing them to 384×384 pixels. Training data is fetched from the "train" directory with shuffle=True to confirm randomness, while validation and test data are loaded from their respective directories with shuffle=False to maintain order.

The class_mode='sparse' setting is used since labels are stored as integer values rather than one-hot encoded vectors. This setup guarantees efficient and structured data feeding into the DL model for classification tasks. Setting include_top=False removes the final classification layers from the pre-trained model, enabling the DR classification task customization. Freezing the layers of EfficientNetV2-L confirms that the weights learned from ImageNet are preserved and prevents modification during training. Initially, all convolutional layers stay frozen to preserve their pre-trained weights and maintain the

learned feature representations. Then, the upper eighteen layers are unfrozen, and the model is further adapted to exact weights for the dataset, still taking advantage of the network's learned features.

The model accepts input images of size 384×384 pixels with 3 RGB channels, thus ensuring the image's input condition for classification. The feature maps are extracted from the EfficientNetV2-L base and then passed through a GAP layer that averages each map, lowering spatial dimensions. This leads to a more compact 2D representation, preserving key information properties and reducing computational complexity.

Afterwards, an FC dense layer that contains 256 units is added to learn high-level feature representations. A dropout layer is added to alleviate overfitting and improve generalization. Next, a four- unit (one unit for each of the four stages of DR classification, that is, normal, mild, moderate, and severe) FC layer is appended. The softmax activation ensures that the model outputs a probability distribution over these four potential classes and lets us classify on the maximum predicted class. This style of DR classification utilizes EfficientNetV2-L's impressive feature-extracting capabilities and pairs them with a simple yet effective classifier. The model utilize pre-trained representations with a negligible computational overhead by freezing the pre-trained layers but selectively fine-tuning the last dense layers.

### 3.3.6 Ablation Study on Fine-Tuned Pretrained models

The Ablation study examines how different fine-tuning choices, augmentation strategies, classifier-head designs, loss functions, and input resolutions affect the performance of five pretrained CNN models (VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2L) for four class DR classification. Each model is evaluated under controlled settings so the influence of each design choice becomes clear across accuracy, class-wise behaviour, calibration, and efficiency.

- VGG16 - The study uses a TL VGG16 model which receives 224×224 inputs and uses the standard VGG16 preprocess_input normalization. The pretrained backbone is frozen initially, and the final four convolutional blocks are unfrozen to examine the effect of partial fine-tuning. A lightweight head with Global Average Pooling, a 128-unit ReLU layer, and dropout is added to analyze the impact of classifier-head simplicity.

- ResNet-50 - The ResNet-50 model is adapted using 224×224 images normalized through the ResNet preprocessing pipeline. The backbone remains frozen except for the final six layers, allowing the study to evaluate how limited residual-block fine-tuning affects feature adaptation. A Global Average Pooling layer followed by a 128-unit ReLU dense layer and dropout forms the classification head, enabling comparison with the simpler VGG16 head under identical training conditions. The architecture helps assess how residual connections respond to partial fine-tuning in the DR domain.

- InceptionV3 - InceptionV3 is fine-tuned using 224×224 inputs and standard Inception normalization. The pretrained layers remain frozen initially, and the last twelve layers are unfrozen to evaluate the effect of fine-tuning deeper inception modules. A custom head with Global Average Pooling, a 128-unit ReLU layer, dropout, and softmax output is added to examine classification-head consistency across models. This setup allows analysis of how multi-branch inception features adapt when only higher-level layers are trainable.

- DenseNet121 - DenseNet121 uses 224×224 normalized inputs and a pretrained ImageNet backbone. The model keeps most layers frozen, selectively unfreezing the final sixteen layers to observe how dense connectivity behaves under controlled fine-tuning. A Global Average Pooling layer, a 128-unit ReLU dense layer, dropout, and a 4-class softmax head are used, matching the classifier designs of the other models. This configuration helps evaluate how DenseNet's feature-reuse structure influences convergence and class-wise performance during partial unfreezing.

- EfficientNetV2L - EfficientNetV2L is fine-tuned using 384×384 inputs to examine the effect of higher input resolution relative to the other models. The pretrained backbone is frozen initially, and the final eighteen layers are unfrozen to measure the impact of deeper fine-tuning in a larger-capacity architecture. A Global Average Pooling layer, a 256-unit ReLU dense layer, dropout, and a 4-class softmax head are added to test whether a slightly larger classifier head improves representation for severe DR classes. This setup allows analysis of how scaling depth and resolution influence model behaviour during fine-tuning.

Across all five models-VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2L-the evaluation uses accuracy, weighted precision, recall, F1-score, and a

confusion matrix to assess performance on the test set and to observe class-wise behavior for Mild, Moderate, Normal, and Severe DR. Each model trains for 10 epochs using the Adam optimizer with a learning rate of 1e-4 and sparse categorical cross-entropy loss, which keeps the training process stable and consistent. The results show that every model benefits from selective fine-tuning, controlled learning rates, and proper preprocessing, which help the pretrained ImageNet features adapt to the retinal domain. Overall, the ablation study shows that the choice of architecture, the number of layers unfrozen, and the design of the classifier head all play an important role in improving predictive performance while keeping the models computationally efficient.

## 3.4 DATASET

The dataset is a widely accepted standard benchmark DR classification database from the APTOS in the Kaggle [70]. This dataset provides high-precision imaging of fundus cameras for the patient's retinas, and each of them are finely marked with the severity level of DR based on the predefined standards of clinical grading. They are classified into four grades of DR as follows:

- **No DR:** This group is the no-DR group, and the subjects in this population do not have visible diabetic retinal damage.
- **Mild DR:** In mild DR, there is usually MA in the retina and small HEM spots, but there is no significant loss of retinal function at this stage.
- **Moderate DR:** At this stage, the retinas exhibit more pronounced signs of damage, including increased MAs, HEMs, and EXs. Retinal blood vessels may begin to show signs of ischemia.
- **Severe DR:** For the appearance of more retinal alterations, such as larger HEMs, EX, and severe ischemia. NV of the retinal vasculature may also occur, resulting in complications such as vitreous HEM or retinal detachment.

This baseline dataset is employed for training and benchmarking DL systems in DR screening. It contains extensive ML annotation with high-quality fundus images and is a valuable tool for training and testing autonomous systems for the on-time diagnosis of DR in telemedicine.

### 3.4.1 Dataset Partitioning

In DL, the dataset is often divided into 70% for training, 20% for validation, and 10% for testing. This split provides a good balance between model development, tuning, and evaluation. The large training portion helps the model learn effectively and reduces the risk of underfitting, improving its ability to generalize to new data [71].

The validation set (about 20%) is used to fine-tune hyperparameters such as learning rate and network architecture. Since it contains data not seen during training, it helps reduce overfitting and provides a more reliable way to choose the best model [72].

Finally, the test set (around 10%) is reserved for evaluating the final model. It offers a realistic estimate of how well the model performs on completely new data. Overall, the 70-20-10 split is considered an effective balance for building, validating, and testing DL models [73].

The APTOS dataset comprises 3367 high-resolution fundus images, partitioned into three subsets for training, validation and testing of DL models used in this research for DR screening. The data set is split as follows:

Training Set (70% - 2,356 images): This set is used to train the model, which learns to recognize patterns, features, and differences between the different stages of DR. When confronted with a broad spectrum of well-labelled images, the training set is instrumental in improving the model's capability to infer results on new data points.

Validation Set (20% - 672 images): It is used during training for model selection and hyperparameter tuning. It holds significant importance in fine-tuning the model to avoid overfitting, ensuring that it performs consistently on both training and test set. This structured partitioning approach confirms that the model's training, evaluation, and fine-tuning are conducted using a balanced dataset to ensure reliable performance. It supports designing a robust and reliable model for detecting the different stages of DR, paving the way for effective and automated screening systems in clinical settings.

Testing set (10% - 339 images): The testing set is employed to validate the model's performance following the training. It is an independent data set that assesses the model's ability to generalize to new visual inputs, providing information about its accuracy, precision, and stage-wise DR classification performance.

### 3.4.2 Hyperparameters for finetuned pretrained models

A loss function is a measure that quantifies the difference between the predicted and actual outputs of a model during training. In this study, two widely used loss functions for multiclass classification - Categorical Crossentropy and Sparse Categorical Crossentropy are utilized to train the EfficientNetV2L model. The model's performance and generalization ability are then evaluated using both loss functions, as summarized in Table 3.1.

**Table 3.1 Loss function evaluation**

| S.No. | Loss Function | Training Accuracy (%) | Validation Accuracy (%) | Training Loss (%) | Validation Loss (%) |
|---|---|---|---|---|---|
| 1 | Categorical Crossentropy | 86 | 87 | 18 | 40 |
| 2 | Sparse Categorical Crossentropy | 89 | 91 | 16 | 39 |

From the Table 3.1 it indicates that Sparse Categorical Crossentropy provided better convergence and generalization, making it more suitable for this multiclass DR classification task, particularly when the target labels are integer encoded.

Learning rate is one of the most essential hyperparameters that define model optimization, and based on this, the model optimizes its parameters at a certain rate. In this case, three different values of learning rate, that is, 0.001, 0.0001, and 0.00001, are considered for experimentation based on 1 to 35 epochs as shown in Table 3.2.

All these cases are considered based on sparse categorical cross-entropy loss as the loss function, and this helps present a consistent framework for experimentation.

At a learning rate of 0.001, the model achieved high training accuracy (up to 98%) but showed lower validation accuracy (85–88%) and increased loss, indicating overfitting. Thereby reducing the learning rate to 0.0001 produced more balanced results, with training accuracy around 85–89%, validation accuracy peaking at 91%, and lower validation loss (39%), reflecting better generalization.

**Table 3.2 Learning Rate evaluation**

| S.No. | Learning Rate | Epoch | Training Accuracy (%) | Validation Accuracy (%) | Training Loss (%) | Validation Loss (%) |
|---|---|---|---|---|---|---|
| 1 | 0.001 | 10 | 96 | 86 | 8 | 62 |
| 2 | | 15 | 98 | 88 | 5 | 67 |
| 3 | | 25 | 97 | 86 | 6 | 87 |
| 4 | | 35 | 98 | 85 | 5 | 88 |
| 5 | **0.0001** | **10** | **89** | **91** | **16** | **39** |
| 6 | | 15 | 85 | 85 | 11 | 46 |
| 7 | | 25 | 86 | 88 | 8 | 54 |
| 8 | | 35 | 87 | 86 | 7 | 59 |
| 9 | 0.00001 | 10 | 85 | 86 | 40 | 37 |
| 10 | | 15 | 88 | 87 | 33 | 35 |
| 11 | | 25 | 92 | 87 | 21 | 33 |
| 12 | | 35 | 95 | 88 | 13 | 37 |

At 0.00001, training progressed slowly but steadily, with accuracy improving to 95% and stable validation performance around 86–88%. Overall, a learning rate of 0.0001 with 10–15 epochs offered the best balance between training efficiency, accuracy, and generalization, while higher rates caused overfitting and lower rates slowed convergence. These parameters correspond to a well-balanced trade-off between convergence speed and model generalization. Therefore, eventual differences are primarily due to the architecture rather than to training hyperparameters.

## 3.5 SYSTEM CONFIGURATION

This research work's hardware and software setup has been carefully selected to fulfil the computational requirements for training DL models for DR Stage classification.

### 3.5.1 Hardware

The model training and evaluation are conducted on a high-performance computing system with an Intel Core i7-9700K processor, featuring 8 cores and 8 threads. The system is further enhanced with an NVIDIA RTX 2080 Ti GPU and 64 GB of memory, ensuring efficient handling of DL computations. Running on a 64-bit Windows 10 operating system, the setup provides a stable and robust environment for experimentation.

### 3.5.2 Software

Framework: Google's TensorFlow provides a comprehensive and scalable open-source platform for developing and implementing ML models [74]. It offers a wide-ranging ecosystem of libraries and tools that facilitate the entire model lifecycle, from data preprocessing covering the pipeline, including training, deployment, and optimisation. Within TensorFlow, Keras is employed as the high-level Application Programming Interface (API), streamlining the construction and training of DL models. Keras is known for its simplicity and user-friendliness, allowing developers to rapidly prototype models with minimal code. Its intuitive interface abstracts complex backend operations, making it accessible to beginners and field experts. Together, TensorFlow and Keras offer a robust integration for developing scalable, efficient, and cutting-edge DL architectures, ensuring flexibility and ease of use throughout the modelling process.

### 3.5.3 Integrated Development Environment (IDE)

Jupyter Notebook is a widely used development environment within the data science community, offering an interactive platform that supports coding, data visualisation, and model experimentation [75]. It provides an ideal setting for building and developing DL models based on its ability to facilitate real-time code execution, debugging, and instant feedback. Documentation with the code is facilitated by markdown cell integration enabling easier presentation of work flows, exchange of results, and collaborative working. Moreover, Jupyter support for inline visualization libraries such as matplotlib and seaborn, help in visualizing data distributions, loss curves, and other performance metrics. This adaptive and interactive characteristic renders Jupyter Notebook a first choice option for activities such as data exploration, model construction, and iterative testing in DL.

## 3.6 PERFORMANCE METRICS

Accuracy, precision, recall, and F1-score are the basic measuring terms of the classification model's performance. Accuracy is the percentage of total observations classified correctly, giving a general idea of the model's performance and is computed in Equation 3.1. Precision calculates the ratio of actual positive instances over the true negative and false positive instances as given in Equation 3.2, which means the model's accuracy in identifying the positive cases. Equation 3.3 is used to calculate sensitivity

(recall) which is the percentage of the actual positive cases the model claims. The F1-score measures model performance as given in Equation 3.4, that balances precision and recall, thus being particularly relevant when the class distribution is unbalanced [76 - 78].

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

$$\text{Sensitivity} = \text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

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

where,

TP = True Positives

TN = True Negatives

FP = False Positives

FN = False Negatives

## 3.7 RESULTS

Training loss, validation loss, training accuracy, and validation accuracy are important performance metrics for assessing the training performance of a DL model, especially in TL tasks like DR classification.

The training loss is the difference between predictions made by the model and true labels of the dataset it is trained on, at the end of every block of iterations or an epoch. It represents the model's learning effectiveness in generalizing the training data. A low training loss recommend that the model is able to learn efficiently, adjusting its weights to minimise errors. The loss function used during classification is called categorical cross-entropy, which computes the difference between the predicted probabilities and labels. As the training continues, a reduction in training loss signifies better model performance for the training set.

Figure 3.6 shows that the training loss rates for different DL architectures which gives a reasonable insight into each model's learning efficiency during training. Finetuned

VGG16's training loss steadily decreased from 1.0529 to 0.3673 in the last epoch. This trend demonstrates that Finetuned VGG16 is learning something, and the training loss is slowly decreasing. Finetuned ResNet-50 demonstrates a significant decrease in training loss from 0.6963 to 0.2566, showing effective learning and optimization. The learning curve exhibits a decreasing trend in loss as the training proceeds and the model attains a smaller final loss.

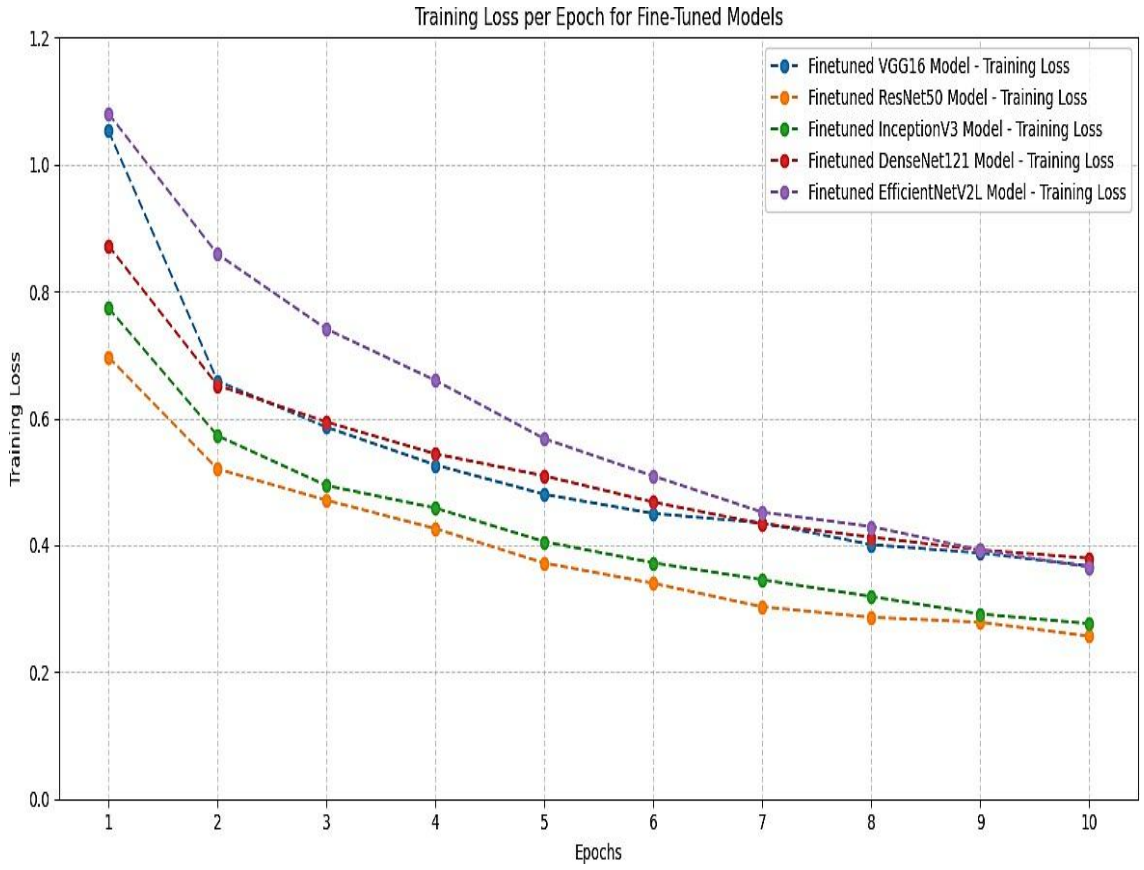


**Figure 3.6 Training Loss for Finetuned Pretrained Models**

Finetuned InceptionV3 (with a higher initial training loss of 0.7737 experiences an even bigger reduction, to a loss of 0.2764. That means the network structure of Finetuned InceptionV3 itself is conducive in decreasing training loss. The ability to suppress significantly the training loss, the clear evidence of the model's learning and adaptation to this classification task which again underlines the efficiency in training of the model.

Finetuned DenseNet121 begins with the least value of initial training loss, 0.8718, and attains the minimum final value of loss, 0.3794. This dramatic decrease in loss indicates

that Finetuned DenseNet121 is quite efficient at learning. Its design, which allows better feature re-use, helps to converge the model quickly and effectively, resulting in higher performance in training loss minimisation compared with other models on this dataset.

Finetuned EfficientNetV2-L begins with a higher training loss of 1.0800 but fluctuates throughout the epochs, ultimately achieving a final loss of 0.3654. The oscillations indicate that the model needs more fine-tuning, but the ultimate training loss shows it is learning effectively. Although it is relatively unstable initially, EfficientNetV2-L yielded a relatively robust performance, showing that the model is still effective after the initial battle.

On the other hand, validation loss tests the model error on the validation set, a different portion of data left aside for training as shown in Figure 3.7. This metric is utilised to estimate the model's generalisation on out-of-sample data. The validation loss is an important overfitting indicator: it should ideally decrease in tandem with the training loss. It may be overfitting if the validation loss is rising while the training loss is decreasing. This results in the use of early stopping, dropout, or model regularisation to enhance the generalisation of the model.

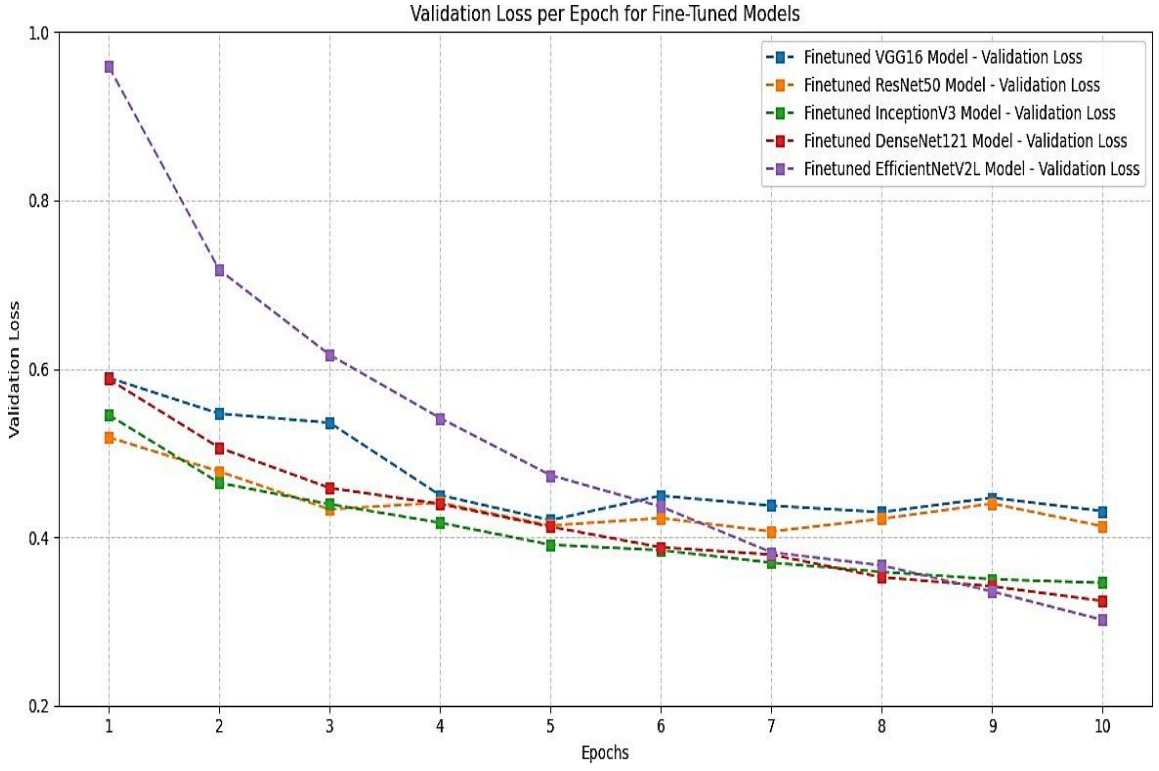


**Figure 3.7 Validation Loss for Finetuned Pretrained Models**

The validation loss of the finetuned VGG16 model steadily drops from 0.5892 to 0.4201 in the initial five epochs, demonstrating effective learning of features. The drop reflects enhanced generalization towards unseen data. From the later epochs, mild fluctuations are seen with values between 0.4296 and 0.4492. The fluctuations reflect that the model stabilizes after initial advancement.

Decrease in validation for the fine-tuned ResNet-50 model shows a gradual downward trend in the initial epochs, dropping from 0.5185 to 0.4138. This is a sign of effective transferring of pretrained weights to the goal task. For the later epochs, there are negligible changes, which represent convergence in the learning process. The loss remains within a narrow range, showing that the performance is consistent.

Finetuned InceptionV3 shows a smooth improvement in validation loss from 0.5451 to 0.3457. This is to indicate the model captures from data effectively and exhibits excellent generalisation performance on the validation set during training.

Finetuned DenseNet121 is good at generalising to unseen data and has validation loss from 0.5875 down to 0.3242. The continual decrease of the validation loss throughout the training epochs suggests Finetuned DenseNet121's promising performance in fighting against overfitting and in capturing the generalisation features through training. Its potential to bring justified validation loss to such a low value makes it one of the most performing models for this data. It guarantees a robust performance on novel data, thus supporting its very high suitability to generalisation tasks.

The validation loss of the fine-tuned EfficientNetV2L model demonstrates a strong and consistent decline across epochs, starting at 0.9588 and reaching 0.3017 by the tenth epoch. This substantial reduction reflects efficient learning and robust feature extraction. The steady improvement indicates that the model continuously adapts to the data without plateauing. Unlike other models, no major fluctuations are observed throughout training. This trend confirms the model's superior generalization capability and stability.

Training accuracy graph show the model performance on the training data and how many training set predictions it gets right. A high training accuracy means the model memorises the training patterns. However, if the model has high training and low validation accuracy, it may overfit the patterns in the training data instead of generalising.

The training accuracies from Figure 3.8 indicate how well each model can identify the training set. Training accuracy of the fine-tuned VGG16 model increases consistently from 0.7175 to 0.8462 over ten epochs. While the growth rate remains consistent, the improvement rate slows down in subsequent epochs. Low oscillation between epochs indicates that the model is adapting to sophisticated features. Overall increasing trend indicates that the model keeps on learning efficiently. The trend indicates smooth convergence during training.

Finetuned ResNet-50 shows much better training accuracy, which starts from 0.7506 and goes all the way up to 0.8957. This steep rise demonstrates the model's good learning capacity and the quality of the training, making it one of the best-performing models interms of accuracy. The significant increase in performance over time indicates that Finetuned ResNet-50 is well-tuned to the dataset and has a strong learning process.

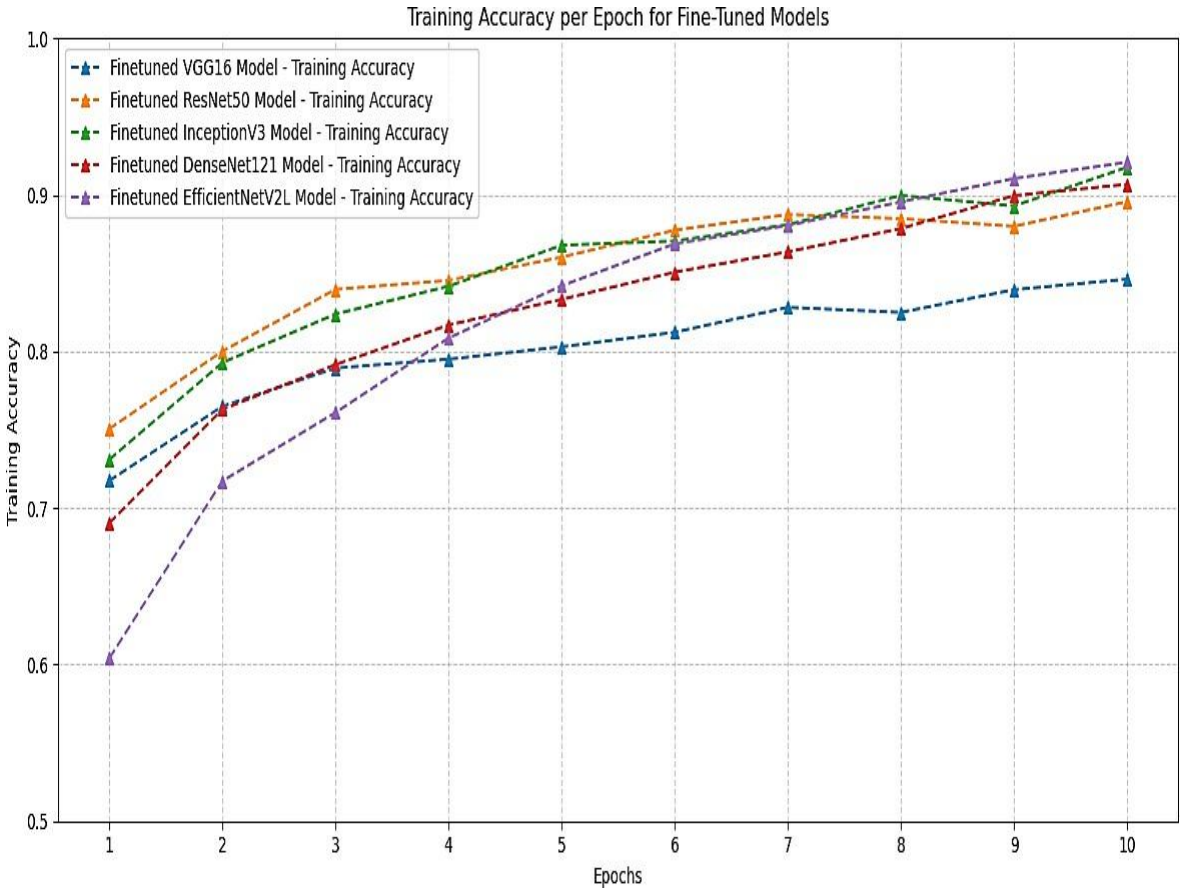


**Figure 3.8 Training Accuracy for Finetuned Pretrained Models**

When initialised, the accuracy of a finetuned InceptionV3 model is 0.7310 and increases during training, reaching an accuracy of 0.9177 at the final epoch. This model's

success in achieving a reasonable level of accuracy could be interpreted as rich evidence that it could learn effectively from the training set and has the architectural capacity for further training.

Finetuned DenseNet121 achieves the highest training accuracy among all models 0.6904 at the 1st epoch to 0.9068 at the final epoch. That implies that it picks up a lot more from the data and continues to learn at every step of the training. With the best optimisation and generalisation ability, it stands out in this comparison.

Finetuned EfficientNetV2- L indicates that this approach performs well with a beginning accuracy of 0.6040 and an ending accuracy of 0.9209. This gradual improvement proves the model's learning is stable, or that the model is good enough to train on, yielding good outcomes. Finetuned EfficientNetV2-L results in high accuracy, similar to Finetuned DenseNet121.

Validation accuracy is the percentage of predictions made right on the validation set and is the primary metric for testing the model on unseen data. A good generalisation model is expected to exhibit increases in validation accuracy as it learns, just as the training accuracy increases. However, if the validation accuracy plateaus or decreases while the training accuracy is still growing, it could mean overfitting. To achieve this goal of generalisation while developing a model, strategies like lowering the model complexity and adding regularisation techniques should be considered. By monitoring training loss, validation loss, training accuracy, and validation accuracy, significant insights are obtained into the learning of the model. Such measures help guarantee that the model learns well from the training data and generalises correctly from unseen samples, an important aspect for tasks such as DR classification. Routine monitoring of model performance enables early detection of overfitting and underfitting, and it helps the model to converge to the best performance.

Figure 3.9 shows that the validation accuracy values offer insights into the models' ability to generalize to novel data, with higher accuracy indicating better generalization. The validation accuracy of the finetuned VGG16 model increases step by step, from 0.7665 to 0.8105, over ten epochs. An increase is prominently seen up to epoch 4, capturing initial generalization. Epochs following this have minimal variability with scores

fluctuating in a small range. This consistency captures the model's consistent performance on unseen data.

Validation accuracy of the fine-tuned ResNet-50 is increasing steadily, from 0.8289 to 0.8616 in ten epochs. Early epochs show strong improvements, and then good high-performance retention continues. Oscillations between epochs are small and indicative of natural fluctuations without performance decline. Accuracy rates remain above 0.8427 from epoch 3. This pattern is indicative of good generalization and robust model performance.

Validation accuracy of fine-tuned EfficientNetV2L model has a stable and appreciable growth, from 0.7812 to 0.8963 over ten iterations. There is improvement significantly in early epochs, which reflects early successful learning. Accuracy keeps on growing steadily with very slight undulation during training. Values are above 0.8718 from the 6th epoch onwards, reflecting high confidence and accuracy of the model.

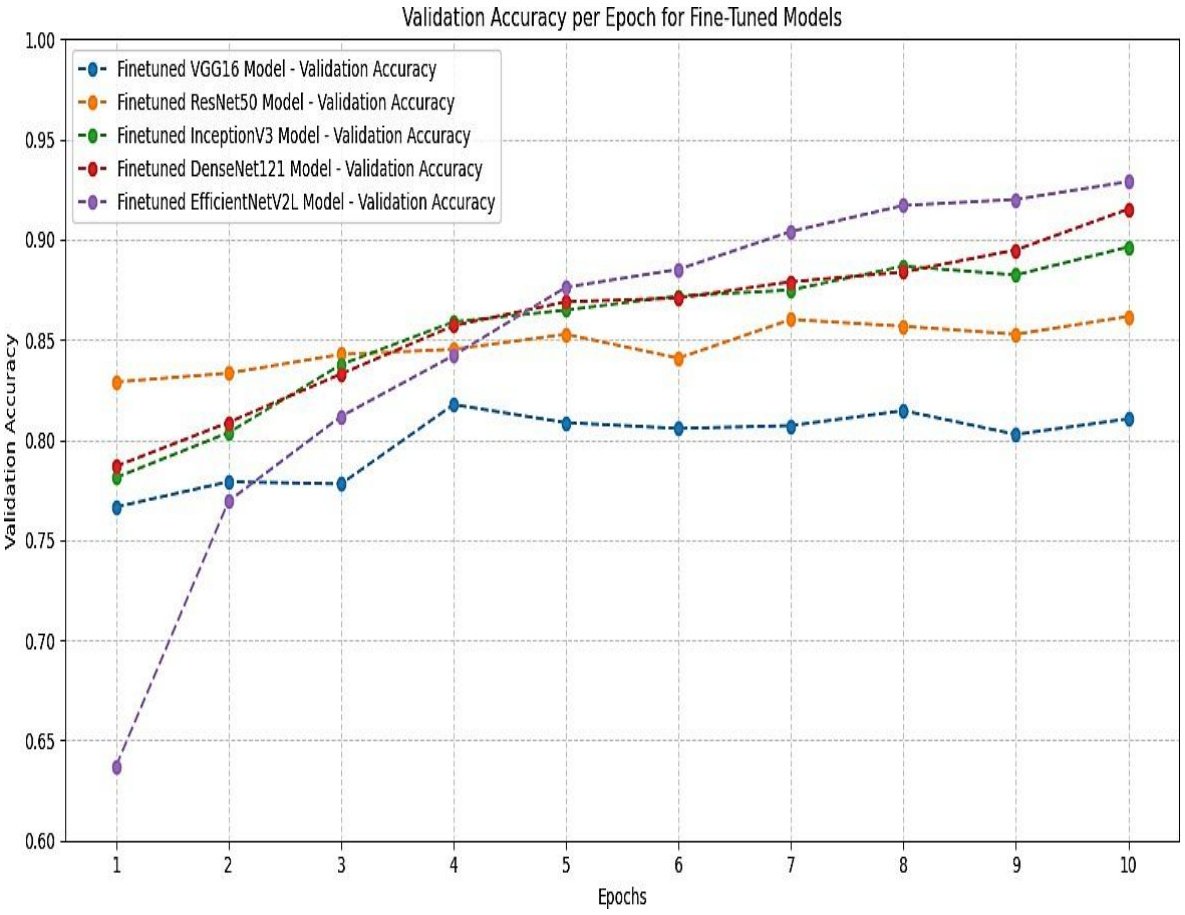


**Figure 3.9 Validation Accuracy for Finetuned Pretrained Models**

DenseNet121 shows validation accuracy ranging from 0.7868 to 0.9152, achieving higher accuracy in several epochs. These results reflect strong generalization capability and effective learning, making DenseNet121 a competitive model with the ability to perform well on unseen data. However, it does not consistently outperform EfficientNetV2-L, which demonstrates the strongest generalization ability.

Finetuned EfficientNetV2-L stands out as the top performer in validation accuracy, starting at 0.6372 and reaching 0.9289 by the final epoch. This consistently high performance indicates that Finetuned EfficientNetV2-L generalises well to unseen data, outperforming all other models regarding validation accuracy, suggesting that it is the better model for this task.

Table 3.3 presents the performance of five pre-trained CNN architectures VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2L evaluated before and after fine-tuning.

**Table 3.3 Performance metrics for Pretrained Models**

| **Architectures/ Performance metrics** | **Without finetuning** | | | | **With finetuning** | | | |
|---|---|---|---|---|---|---|---|---|
| | **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1score (%)** | **Accuracy (%)** | **Precision (%)** | **Recall (%)** | **F1score (%)** |
| VGG16 Model | 66 | 78 | 61 | 69 | 85 | 84 | 85 | 85 |
| ResNet-50 Model | 73 | 80 | 65 | 72 | 87 | 86 | 87 | 87 |
| InceptionV3 Model | 76 | 84 | 73 | 78 | 89 | 88 | 89 | 89 |
| Densenet121 Model | 84 | 86 | 82 | 84 | 90 | 89 | 90 | 90 |
| **EfficientnetV2L Model** | **87** | **89** | **78** | **83** | **92** | **91** | **92** | **92** |

Without fine-tuning, the models achieved moderate performance, with accuracies ranging from 66% (VGG16) to 87% (EfficientNetV2L). After fine-tuning, notable improvements were observed: VGG16 increased from 66% to 85%, ResNet-50 from 73% to 87%, InceptionV3 from 76% to 89%, DenseNet121 from 84% to 90%, and EfficientNetV2L from 87% to 92%.

The Finetuned VGG16 improves with an accuracy of 85%, precision of 84%, recall of 85%, and F1-score of 85%, showing better performance and the Finetuned ResNet-50 performs better with an accuracy of 87%, precision of 86%, recall of 87%, and F1-score of 87%, demonstrating stronger learning capability. InceptionV3 offers further improvement with an accuracy of 89%, precision of 88%, recall of 89%, and F1-score of 89%, highlighting its learning and generalisation.

While the Finetuned DenseNet121 results with an accuracy of 90%, precision of 89%, recall of 90 %, and F1-score of 90 %, showing superior performance in terms of precision, though recall is further optimised. Finally, the Finetuned EfficientNetV2L leads the comparison with an accuracy of 92%, precision of 91%, recall of 92%, and an F1-score of 92%, demonstrating exceeding performance across all metrics, particularly in recall and F1-score, making it the reliable model for the task.

These results demonstrate that fine-tuning enhances the models' ability to extract task-specific features, leading to better generalization and higher predictive accuracy. Among all tested architectures, EfficientNetV2L achieved the highest overall performance, indicating its superior feature extraction capability and adaptability for the given dataset.

## 3.8 DISCUSSION

This chapter focuses on DR classification which involves determining the severity of the disease by analyzing retinal images, where precision and high generalisation are paramount for accurate diagnosis. Hence, DL models, including VGG16, ResNet-50, InceptionV3, DenseNet121, and EfficientNetV2-L, are finetuned and evaluated based on their ability to classify DR stages using various performance metrics such as training loss, validation loss, training accuracy, validation accuracy, precision, recall, and F1 score.

Among the models tested, Finetuned EfficientNetV2-L exhibits better results in every assessed metric, particularly in validation accuracy, where it consistently outperforms other models with a range of 0.6372 to 0.9289. This indicates its exceptional ability to generalize to unseen data, an important characteristic in real-world medical applications where unseen patient data must be correctly classified. Finetuned EfficientNetV2-L also achieves a low validation loss (ranging from 0.9588 to 0.3017), suggesting strong generalization and minimal overfitting. In terms of training accuracy, it varies from 0.6040 to 0.9209, indicating robust learning throughout the training process. These results are the best among all these models, highlighting EfficientNetV2-L as this study's most reliable model for DR classification.

In comparison, Finetuned DenseNet121 also demonstrates strong performance with relatively high training accuracy and low validation loss, but its validation accuracy and F1 score are marginally below the levels observed in Finetuned EfficientNetV2-L. The Finetuned ResNet-50 and InceptionV3 show moderate training and validation accuracy results, but do not achieve the highest performance.

Hence, Finetuned EfficientNetV2-L is this study's most effective and reliable model for DR classification. It excels in all key metrics, particularly in validation accuracy and loss, demonstrating superior generalization to unseen data compared to other models. This renders EfficientNetV2-L a highly competitive candidate for real-world implementation within healthcare environments, where the capability to label new and novel retinal images is critical to quality patient care.