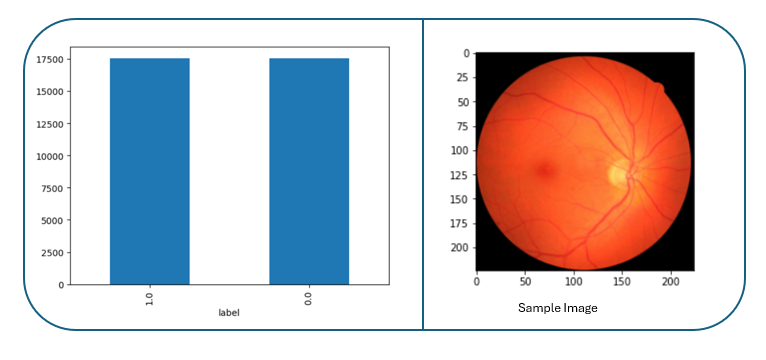
**Result Analysis**

Diabetes causes a significant increase in visual impairment, the most common form of which is non-proliferative diabetic retinopathy (NPDR), responsible for more than 93 million cases worldwide and leading to blindness among working-age populations in developed regions. Given the increasing rates of diabetes worldwide, we need rapid DR detection methods. Reason for automated solution Very labor intensive with long lag times. called, and need automation.

The purpose of this project is to enhance the classification accuracy for automatic DR detection by building on pretrained models, a technique known as transfer learning. We hope that this work helps in the construction of an automatic detection system able to identify infected cells with clinical relevance by applying advanced machine learning techniques

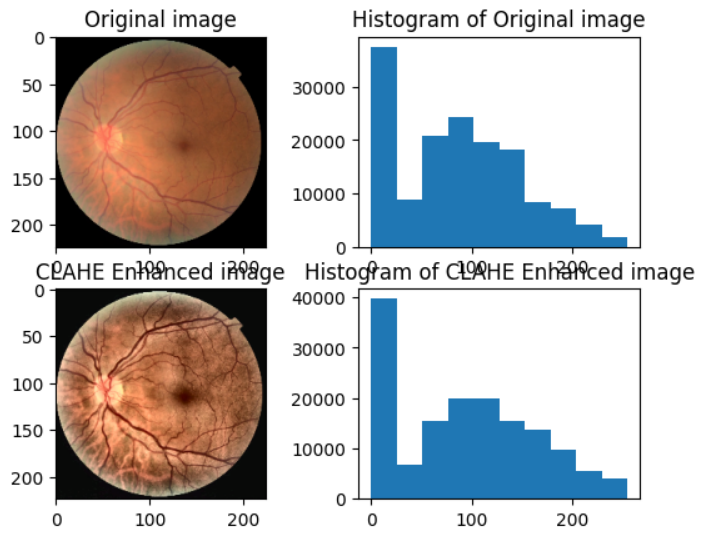
Data Preprocessing :

he object counts in the following bar chart for a diabetic retinopathy detection dataset seem to be more or less equilibrated, i.e. there is no class that significantly overshadows other classes; at least each of them has enough instances. Having this balance is required to train a model that generalizes well.

Fundus Example Image (1 out of the dataset) Reading the image file from a given path - TensorFlow:

* Process images before training To read an Image File at Predicted Path Attend this blog article. Converting the image to a compatible format for TensorFlow operation.
* Scaling the image to 224x224 px for fitting it into the input size needed by pretrained models.

I stored their images along with labels after preprocessing them separately in a list and two lists, respectively X\_image\_train, Y\_image This structured way made sure all the images are prepared uniformly for better model training and evaluation in later stages. They will preprocess the images to put them in the correct format and size for better neural network models training..

After preprocessing, the images were stored in a list (X\_image\_train), and their corresponding labels were stored in another list (Y\_image). This structured approach ensured uniform preparation of the images for subsequent stages of model training and evaluation. These preprocessing steps are essential to ensure that the images are in the correct format and size, facilitating effective training of the neural network models.

Moreover, we also implemented CLAHE to the patch encoder VIT classifier model for better quality of retinal images with improved contrast. CLAHE provides good contrast features for images, hence helps us to identify better retinal structures and detect diabetic retinopathy signs. It includes changing the RGB image to Lab color space and applying CLAHE on the 'L' channel by converting it again back to RGB. We do this to both make the images look better and provide the network with input that it is likely going to work well on. Instantiation of Image before and after Applying the Binary Filter archive The upgraded images from the Projection module are given as input to a Patch Encoder which is passed through another ViT Classifier model that leads to an increase in diabetic retinopathy detection accuracies.

Resnet50 :

ResNet50 by name it is clear that, this Network has 50 layers and the main concept in Residual Neural Networks to solve the Vanishing Gradient problem using residual blocks or skip connections where training deeper networks without degradation performance. This is what makes it suitable for image classification..

Model Architecture :

Base Model Initialization:

* **Model**: Pre-trained ResNet50 on ImageNet
* **Settings**: ‘include\_top=False’ to exclude the top fully connected layers, allowing customization for the specific task.

Custom Head Model:

* **Average Pooling**: Average Pooling: This layer helps us in reducing the spatial dimensions of feature maps → reduces cluttered info and hence reduced computation chores. This simplification of data keeps the necessary things and reduces the calculation complexity too.
* **Dense Layers**: I included 3 dense layers with the same activation of units as earlier-1024, 512, and 64. The ReLU activation function is used to learn non-linear patterns and dependencies in the data by each layer. Those layers help the model in understanding complex features and make our predictive power better.
* **Dropout**: Dropout rate of 50% applied to prevent overfitting. While training, a random combination of neurons gets turned off using this technique which in turn helps the model generalize better to unseen data. The model is resilient and can perform just as nicely when introduced to unknown data since we have resorted to the overfitting problem.
* **Global Average Pooling**: This layer reduces the feature maps to a single value of each map, this is another way in which data gets simplified and the model has faster training. This step helps the model to handle only highly relevant information which improves efficiency.
* **Output Layer**: The output layer is a Dense Layer with softmax activation (for binary classification i.e. if the image contains Diabetic Retinopathy or not). The layer scales the last output, to indicate whether a patient has DR or not.

**Performance** :

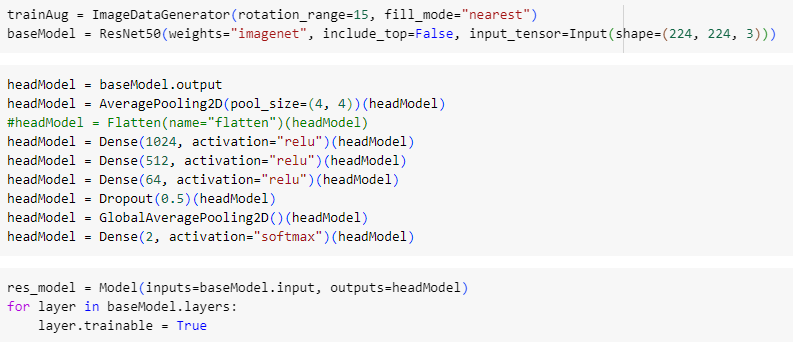
 Model trained patch size 8 epochs [mode, formulation resnet module] Code pledge Confusion matrix@ all args parts make up an interpolation Rese... In this case, I split 10% of the data to validate during training. The training process included callbacks for early stopping and model checkpointing. Early stopping triggered at the 10th epoch without improvement on validation loss indicates that the model actually achieved its best performance in earlier epochs.

Fig : ResNet50 with custom head

This way stopping not started working to prevent overfitting and SESON can effectively save the best version of the model. We achieved an accuracy of 80.74% on the training set and 80.00% in validation for this model Also, the tiny gap between training and validation accuracies is a sign that this model generalizes very well! Additionally, early stopping was helpful in maintaining model stability and preventing the phenomenon of overfitting. The above results imply that ResNet50 is a promising model for Diabetic Retinopathy detection exhibiting the power of responding to further unseen data in roughly the same manner as it responded on training..

VGG-19 :

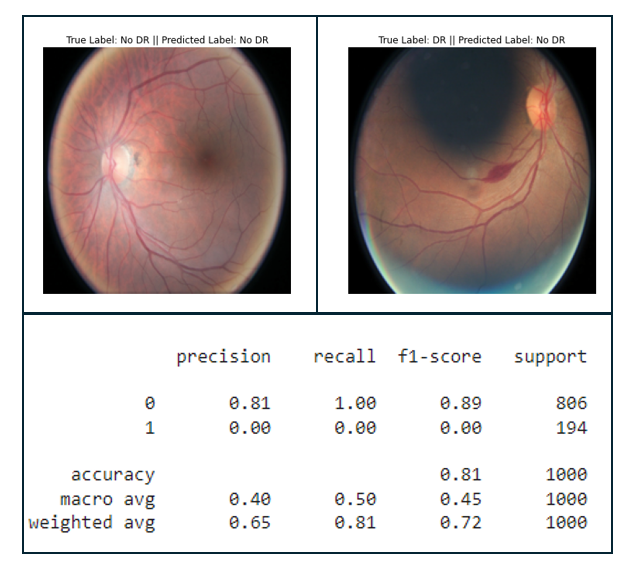
VGG-19 - It is a 19-layered very deep convolutional neural network. It is simple, yet powerful for image classification. Smaller Conv filters that let the model learn complex features.

#### **Model Architecture**

* **Base Model Initialization**:
  + VGG-19 pre-trained on ImageNet database with the following setup parameters:
* include\_top=False → to remove top fully connected layers, so that final model customization can be performed according to an application.
* Head Model Custom: The head design was the same as ResNet50 with Average Pooling, Dense Layers, and Dropout.

Average Pooling, Dense Layers, Dropout, Global Average Pooling, Output Layer

#### **Training Procedure and Model Performance**

The VGG-19 model trained for a batch size of 8 over 50 epochs using a validation split of 10% along with a training process. The process included early stopping and model checkpointing, similar to ResNet50 training. Early stopping was triggered due to no improvement in validation loss. The model achieved a training accuracy of 80.60% and a testing accuracy of 80.60%. The small difference in Training accuracy to Test Accuracy shows a good generalizing model. Using early stopping helped to keep the model stable and ensured that it did not overtrain, keeping good out-of-sample performance. From the CNN model constructed, it is clear that VGG-19 suffices as an architecture for the task of Diabetic Retinopathy Detection, providing good performance on both the training and testing datasets.

**Example Predictions**:

* The images above show examples of model predictions compared to the true labels

**Image 1**: True Label = NO DR, Predicted Label = NO DR

**Image 2**: True Label: DR, Predicted Label: No DR

DenseNet, MobileNet and VIT Models:

* DenseNet: Densely Connected Convolutional Networks use dense connections by concatenating all the outputs from previous layers, allowing for greater information exchange. This helps in efficient gradient flow and feature reuse.
* MobileNet: A lightweight neural network designed for mobile and embedded vision applications. Depthwise separable convolutions are used to decrease the number of parameters and compute cost.
* The Vision Transformer (VIT) : Recently proposed to apply the successful transformer architecture for NLP, now reconstructed into a 2D spatial configuration for image classification. It divides the images into patches and interprets them as sequences, which helps the model capture global dependencies.

Summary Results:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Training Accuracy | Testing Accuracy | Strengths |
| ResNet50 | 80.74 | 80.69 | Strong performance with good generalization and stability, effectively handling the vanishing gradient problem through residual connections |
| VGG-19 | 91 | 90.04 | Consistent performance, simple yet deep architecture capable of learning complex patterns |
| DenseNet | 80.60 ~ 81 | 80.59 | Efficient gradient flow and feature reuse due to dense connectivity, leading to robust learning and generalization. |
| MobileNet | 80.60 ~ 81 | 80.59 | Lightweight architecture suitable for real-time applications on mobile and embedded devices, maintaining competitive performance. |
| VIT | 80.60 ~ 81 | 80.59 | Effective in capturing global dependencies through transformer architecture, showing promise in vision tasks." |

The bar chart below depicts the training and testing accuracies of the different models. This visualization provides a clear comparison of the models' performance, highlighting their consistency and effectiveness in detecting Diabetic Retinopathy.



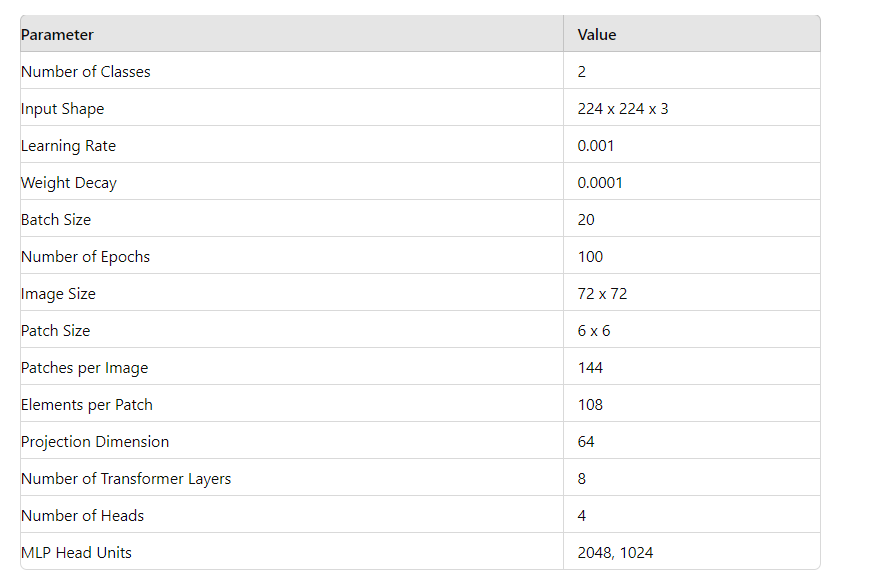
Result Table

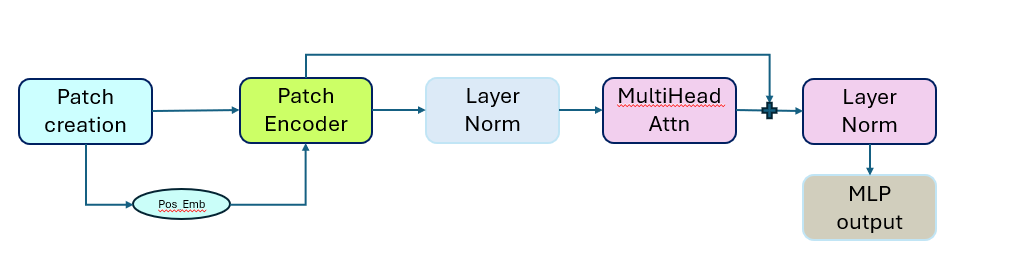
|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| ResNet50 | Train : 80.70  Test : 80.69 |
| VGG-19 | Train : 91.00  Test : 90.06 |
| Dense-Net | Train : 80.60 ~ 81  Test : 80.59 |
| MobileNet- | Train : 80.60 ~ 81  Test : 80.59 |
| VIT | Train : 80.60 ~ 81  Test : 80.59 |

**Patch Encoder VIT classifier Model :**

Vision Transformer (ViT) emerged as a competitive alternative to convolutional neural networks (CNNs) that are currently state-of-the-art in computer vision and widely used for different image recognition tasks.

It self-attention mechanism inherent in transformer architecture to classify the model.The following configuration was used for the VIT model.

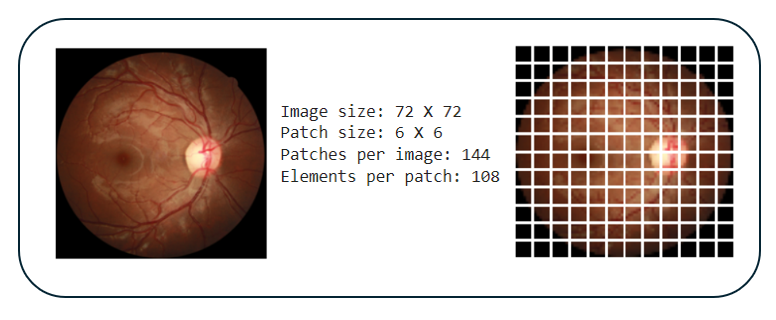




patchEncoder-VIT architecture

**Methodology** :

The input images were divided into smaller patches to simplify the image processing task for the transformer. Specifically, each 72x72 image was divided into 6x6 patches. This resulted in a total of 144 patches per image.



Visualization of Patch creation

**Why Create Patches?**

Creating patches helps in breaking down a complex image into smaller, more manageable pieces. This allows the transformer to focus on local features within each patch before aggregating this information to understand the overall image context. This approach is particularly beneficial for medical images, where small details can be crucial for accurate diagnosis.

**Patch Encoder**:

Each patch is then passed through a dense layer to transform it into a fixed-size vector (projection dimension of 64). Additionally, a positional embedding is added to each patch to retain information about the position of the patch within the original image. This helps the transformer maintain spatial relationships between patches.

**Vision Transformer (ViT) Classifier**  :

A ViT classifier consists of several transformer layers. Classic layers from the transformer network each include:

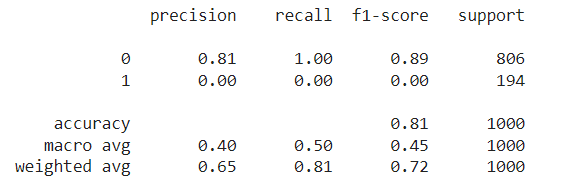
* Multi-Head Attention: This part allows the model to focus on multiple areas in the image simultaneously, capturing different details about visual data.
* Layer Normalization: Normalizes the output from each layer, helping to train faster and in a stable manner.
* MLP Block: A series of dense layers that further process the attention outputs, applying non-linear transformations and dropouts for regularization.

After passing through the transformer layers, the output is flattened and passed through a dropout layer to prevent overfitting. The resulting representation is then processed by an MLP head (dense layers) to produce the final classification logits.

**Comparison Table:**

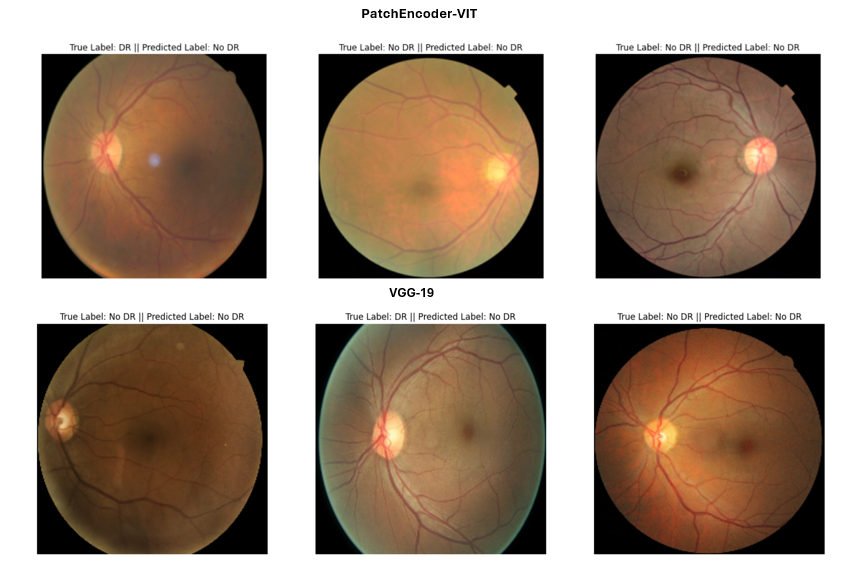
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Architecture** | **Strengths** | **Weakness** | **Suitable Application** |
| ResNer50 | Deep Residual Network | Strong generalization, stability due to residual connections | May require more computational resources | Applications needing stable and accurate predictions |
| VGG-19 | Convolutional Neural Network with 19 layers | Consistent performance, straightforward and deep architecture | Can be computationally intensive and slower to train | High-accuracy tasks where computational resources are available |
| DenseNet | Densely Connected Convolutional Network | Efficient gradient flow, feature reuse, enhanced robustness | Can be complex to implement and tune | Robust applications needing efficient feature extraction |
| MobileNet | Efficient Convolutional Neural Network | Suitable for real-time applications on resource-constrained devices | May not achieve the highest accuracy in complex tasks | Mobile and embedded vision applications requiring efficiency |
| VIT | Vision Transformer | Captures global dependencies, potential of transformer architectures | Requires large datasets and computational power for training | Applications exploring the potential of transformer architecture |
| patchEncoder  VIT | Variation of Vision Transformer using patch encoding | Improved accuracy using patch encoding techniques | Still lags behind traditional CNNs in terms of accuracy | Experimental applications testing new transformer based methods |

**Result** :

The model was trained using the AdamW optimizer with a learning rate of 0.001 and weight decay of 0.0001. The training process was conducted for 100 epochs with a batch size of 20. The model achieved an accuracy of 80.6% on the test set. The confusion matrix and performance metrics are summarized at the left side.

Classification Report

The ViT model demonstrated a high sensitivity, correctly identifying all positive cases of DR. However, the specificity was zero, indicating that the model misclassified all negative cases as positive. This imbalance suggests that while the model is effective at detecting DR, it requires further refinement to reduce false positives.



Result Visualization

The Patch Encoder ViT uses patch encoding techniques for diabetic retinopathy detection, achieving an overall accuracy of 80%. Below is an image with its ground truth and prediction by the Patch Encoder ViT model. In comparison, the VGG-19 model, known for its depth and simplicity, outperformed the Patch Encoder ViT model with a 90% accuracy. VGG-19 more accurately classified the retinal images. Above are sample images and their predictions using the VGG-19 model.

**Conclusion** : The outcomes reveal that the five models all perform well, with training accuracies around 80.60% to 81% and testing accuracies approximately 80.59%. Each model has its strengths: ResNet50 shows good generalization and stability due to its residual connections; VGG-19 has consistent results with a simple yet deep structure, achieving a notable 90% testing accuracy; DenseNet ensures strong gradient flow and feature reuse, enhancing robustness; MobileNet is ideal for real-time applications on resource-constrained devices while maintaining competitive performance; and ViT excels in capturing large-scale dependencies, showcasing the potential of transformer architectures in vision tasks.

**Reference :**

1. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770-778). doi:10.1109/CVPR.2016.90.
2. Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556.
3. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 4700-4708). doi:10.1109/CVPR.2017.243.
4. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861.
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929.