Diabetic Retinopathy Evaluation: A ViT Approach

Author: Adi Majumdar

Mentor: Abdullah P Rashed Ahmed

Abstract

Background: Diabetic Retinopathy (DR) is a complication of diabetes, which afflicts about 400 million people in today's world, that causes the blood vessels of the retina to swell and to leak fluids and blood. Early detection of DR can help manage it with lifestyle changes and medical interventions, preventing the need for costly treatments like laser or surgery, and providing a much better standard of life. This motivates the search for easier, affordable and accessible DR detection technology using the help of Artificial Intelligence given the huge number of potential patients.

Purpose: This paper investigates the application of a Vision Transformer (ViT) model for classifying diabetic retinopathy severity levels from fundus images, aiming to provide accurate predictions essential for early diagnosis and treatment.

Methods: To optimize model performance, images were preprocessed to appropriate resolutions, and the dataset was divided into training, validation, and test sets. Data augmentation techniques, including geometric transformations and color adjustments, were employed to enhance data diversity and prevent overfitting. The ViT model was modified by freezing upper layers and adjusting the final fully connected layer to classify images across five severity levels, leveraging transfer learning for efficient fine-tuning.

Results: With a learning rate of 0.0002 and a batch size of 52, the model was trained with early stopping, achieving a validation accuracy of 75.96%, a loss of 0.58, a recall of 0.72, a precision of 0.700, and an F1-score of 0.675.These results demonstrate the model's balanced performance, capturing most positive cases while maintaining reasonable precision. Yet, the results fell short of 90%+ accuracy obtained in [8] throughout our testing period and more analysis is necessary to assess. Future work will explore dynamic learning rate schedules and alternative batch sizes to further enhance model accuracy to reach the higher goal.

Conclusion: This study highlights the Vision Transformer model's potential for nuanced medical image classification tasks, contributing to advancements in automated diagnostic tools.

1. Introduction

Diabetes is a disease, caused by the lack of insulin, which is reflected by an increased amount of glucose in the blood. It affects more than 425 million people and is expected to rise to 552 million by 2030[1]. Diabetic Retinopathy (DR) is a complication of diabetes that causes the blood vessels of the retina to swell and to leak fluids and blood. DR damages retinal blood vessels, causing vision problems that can lead to blindness, significantly impacting independence, work ability, and quality of life [2]. As global diabetes rates rise, diabetic retinopathy (DR) also increases, affecting nearly one-third of people with diabetes and leading to progressive vision loss if untreated. Worldwide, DR causes 2.6% of blindness.

Early DR detection can be managed with lifestyle changes and medical interventions, but advanced stages require costly treatments like laser therapy or surgery. Research to better understand and treat DR could enable early intervention, reducing healthcare costs, preventing blindness, and offering insights into other diabetes-related vascular diseases.

DR is detected by the appearance of different types of lesions on a retina image. These lesions are microaneurysms (MA), hemorrhages (HM), soft and hard exudates (EX) [9]. Microaneurysms are the earliest sign of DR that appears as small red round dots on the retina due to the weakness of the vessel's walls. Hemorrhages (HM) appear as larger spots on the retina, where its size is greater than 125um with an irregular margin. There can be two types of HM - superficial HM called flame and deeper HM called blot.
Hard Exudates appear as bright-yellow spots on the retina caused by leakage of plasma, usually found on the outer margins of the retina. Soft exudates, also called cotton wool, appear as white spots on the retina caused by the swelling of the nerve fiber, usually oval or round in shape. The five stages of DR depending on the presence of these lesions are, namely, no DR, mild DR, moderate DR, severe DR and proliferative DR.

In the case of image classification for DR severity, this paper applies the ViT architecture for DR grade recognition, classifying fundus images into five grades, as introduced before: Grade 0 - no DR, Grade 1 - mild non proliferative diabetic retinopathy (NPDR), Grade 2 - moderate NPDR, Grade 3 - severe NPDR and Grade 4 - proliferative diabetic retinopathy (PDR). To clarify, a fundus image is a photograph of the interior surface of the eye, specifically the retina, optic disc, macula, and surrounding blood vessels. These images are captured using a specialized camera called a fundus camera during an eye examination.

To perform ViT for DR grade classification, we start with a pre-trained Pytorch based Vision Transformer, developed by Luke Melas. We subdivide the fundus images into non-overlapping patches and convert them into sequences to be used with the Transformer using flattening and embedding. The position embedding is added to preserve the position information of the patches. These generated patch sequences are then fed into several multi-head attention layers to generate the final representation. The resultant token sequence is input to the SoftMax classification layer to produce the recognition output.

The remainder of the paper is organized as follows: Section 2 provides a background to the Vision Transformer based approach. Section 3 introduces the relevant datasets considered. Section 4 introduces the model. Section 5 provides the overall methodology used. Section 6 analyzes and discusses the results. Finally, Section 7 concludes the work.

2. Background

Early detection and treatment of DR is crucial to managing DR with lifestyle changes and early medical intervention. This is essential to improve life for the patients afflicted as well as reduce their healthcare costs, especially given that most of the people afflicted live in low- and middle- income countries. Before deep learning, computer vision relied on manual feature engineering, where experts designed specific features (like edges, shapes, or textures) to represent images. This approach was labor-intensive and often limited by the features' complexity.

With deep learning, convolutional neural networks (CNNs) automatically learn hierarchical feature representations, making models more powerful and adaptive to complex visual data. This shift has greatly simplified the workflow, as models learn directly from raw data without hand-engineered features. Deep learning has enabled dramatic improvements in image classification accuracy achieving superhuman performance on large-scale classification tasks. [4][5]

Alternatively, to CNN-based approaches, a deep learning method entitled Transformer has also been proposed achieving success on vision tasks [10]. It relies on a powerful attention mechanism, which focuses on certain parts of the input image to obtain more efficient results and is today the norm for Natural Language Processing (NLP) and utilized in machine translation, language modeling and speech recognition. Thus, researchers were looking to extend Transformer to image processing tasks, despite a basic challenge. Compared to language sequences, images are matrices with spatial structures containing many pixels, forcing the attention mechanism to estimate each pixel-pair interplay and imposing a high computational cost. This has been resolved by focusing on critical parts of the image and then processing them sequentially.

Transfer learning, especially using pretrained models, has accelerated the development of vision applications. Pretrained models on large datasets like ImageNet can be fine-tuned on smaller datasets, making it feasible to achieve high accuracy without massive, labeled data. This approach has opened deep learning to domains where labeled data is scarce or costly, such as in medical or scientific imaging. This was demonstrated by Dosovitsky et al [5] by subdividing an image into non-overlapping patches and inputting these as a sequence of patch embeddings into a Transformer, which achieved great success by exploiting its multi-head self-attentive mechanism, creating a pre-trained Vision Transformer (ViT). In a recent study used for the diagnosis of pneumonia through transfer learning, the ViT showed considerable competitive advantage compared to 7 other CNN models [6].

3. The Diabetic Retinopathy Dataset

There were 3 different datasets that were considered before embarking on the project. All the datasets are from publicly available sources so that anyone who wanted to follow our work and recreate it - the basic requirement of good research - would be able to do so without having any issues in accessing the dataset to create comparative results. The three datasets considered were as

We can see that the First Dataset had some better features to be used. It had a separation between train and test datasets. On looking at the fundus images in the dataset, we believed it could help to find affected arteries, white clumps and other elements that could provide a deeper indication of the severity of diabetic retinopathy in a person’s eyes. The model being used could likely get trained on these elements and would be able to provide a better evaluation of the severity.

Dataset preprocessing: We did have to perform some preprocessing on the dataset as some of the pre-trained Vision Transformers being used required us to provide images of two resolutions: (224x224) as well as (384x384). We developed a Python script to perform these transformations on the data and create two separate databases of images with different resolutions.

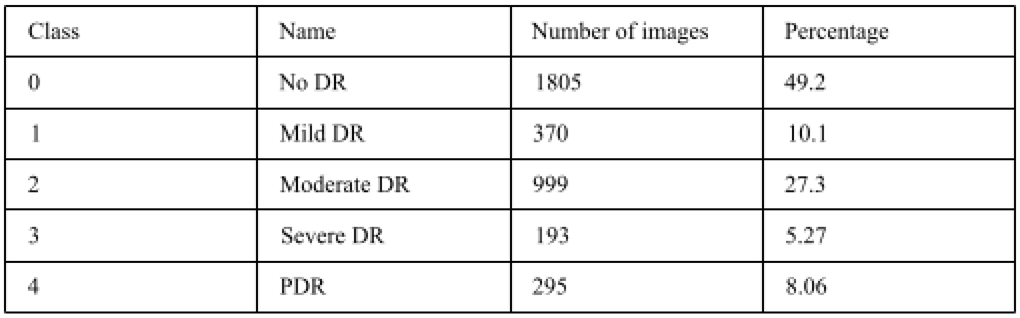
Both the databases of images were then divided into a training set, test set, and a validation set according to an 8:1:1 ratio with another Python script to perform the separation into folders. The sets were also required to be separated according to the pre-labelled classification labels that were already available with the data.

The primary dataset used in our project is the **Diabetic Retinopathy (resized)** dataset, sourced from Kaggle. The dataset can be accessed via the following link: Kaggle - Diabetic Retinopathy (resized). It primarily consists of fundus images of the retina, formatted specifically for classification tasks.

The dataset includes two main components. First, the **trainLabels.csv** file contains two columns: the 'image' column lists the file names of the images, and the 'level' column provides the corresponding labels indicating the severity level of diabetic retinopathy. Second, the dataset provides a folder named **Resized_train**, which contains the resized images. These images were resized to 1024x1024 pixels if their original dimensions exceeded this size, while smaller images remained unaltered.

In terms of data characteristics, the images are prepared in a structured manner to facilitate classification techniques. The fundus images are specifically formatted to display a complete circular view of the eye, ensuring consistency and completeness in the input data. The dataset is organized into training data and resized test data, allowing for effective training of the AI model as well as evaluation of the model's performance on unseen data.

The overall data distribution of the dataset per class is given in the Table below.



Class	Name	Number of images	Percentage
0	No DR	1805	49.2
1	Mild DR	370	10.1
2	Moderate DR	999	27.3
3	Severe DR	193	5.27
4	PDR	295	8.06

Figure 1 : Dataset Management : Imbalanced Ratio of Classes with According Percentages

The data above demonstrates that there is an imbalance in the dataset likely causing an overfitting for the larger classes, class 0 and class 2, and the underfitting for the remaining classes. To mitigate this issue, I augmented the data by performing image transforms of randomly rotating, scaling and cropping the images.

4. The Vision Transformer

In computer vision, Convolutional Neural Networks are the de-facto standard in image classification problems. Transformer architectures have become a similar standard for Natural Language Processing (NLP) tasks, yet the application to computer vision remains limited. It was shown that a pure transformer based model applied directly to sequences of image patches can also perform as

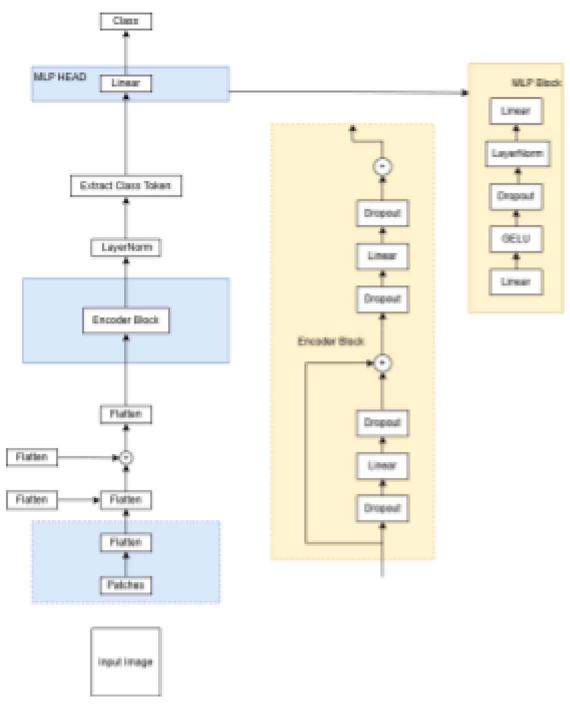


Figure 2 : ViT Architecture with added Sequential MLP Block

well or better on image classification tasks [5]. When pre-trained on large amounts of data from image recognition benchmarks like ImageNet, the Vision Transformer (ViT) model was found to be superior in quality compared to the state-of-the-art convolutional networks while being more parallelizable and requiring significantly less time to train [7].

The Vision Transformer model works in the following steps. First, it splits an image into fixed-size patches, linearly embedding each one of them, adding position embeddings, and feeding the resulting sequence of vectors to a standard Transformer encoder. The standard Transformer encoder maps an input sequence of symbol representations into a sequence of continuous representations using a stack of N=6 identical layers. Each of these layers have two sub-layers - first being a multi-head self-attention mechanism and the second being a simple, position-wise fully connected feed forward network. The output of each sub-layer is Layer Normal (x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub layer itself. Note that to facilitate the residual connections as described, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension 512 [7] In order to perform classification, the model uses the standard approach of adding an extra learnable "classification token" to the sequence. The main training steps are presented in Algorithm 1 and to see the pytorch code used to define this model, see supplementary material 3A.

4.1 Hyperparameters

The following hyperparameters were chosen for the ViT model we use.

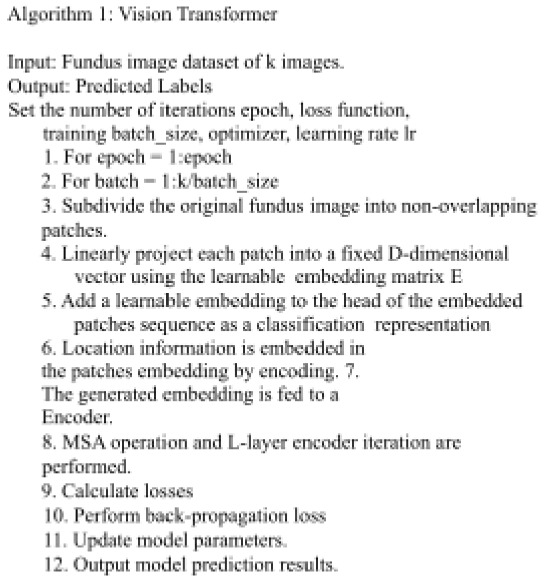
In our model configuration, we kept the **dropout rate** consistent with the values used in the pre-trained transformer, without any modifications. Specifically, dropout is defined by Dropout(p=0.1, inplace=False) in both the attention and feedforward layers within each block, with a value of 0.1 (or 10%). This helps regularize the model and prevent overfitting. Although we did not alter it, adjusting the dropout rate (e.g., increasing to 0.2) could influence the model's generalization capabilities, particularly on smaller datasets.

The **Layer Normalization epsilon value** is set to 1e-06, defined as LayerNorm((768,), eps=1e-06). This small constant ensures numerical stability by preventing division by zero during normalization. While it is rarely changed, fine-tuning this value may improve model stability if issues arise during training.

The **final classification layer dimension** is defined as Linear(in_features=768, out_features=5, bias=True). The output dimension is set to 5, corresponding to the number of classes in our classification task (such as diabetic retinopathy severity levels 0-4). Adjusting this parameter would only be necessary if the classification task involved a different number of output classes.

We configured the **learning rate** to 0.0002. The learning rate is a critical hyperparameter that controls the step size for updating model weights. Setting it too high can lead to convergence problems, while too low a value may slow down the training process. Techniques like learning rate schedules (e.g., cosine annealing, step decay) or warm-up strategies can help optimize the learning rate over the course of training.

Finally, the **batch size** is set to 160. The batch size determines how many samples are processed before the model's weights are updated. Larger batch sizes provide more stable gradients, reducing noise, whereas smaller batch sizes introduce variability in updates. Due to GPU memory constraints, we employed gradient accumulation to simulate a larger effective batch size by accumulating gradients across multiple mini-batches before performing weight updates.



```
Algorithm 1: Vision Transformer

Input: Fundus image dataset of k images.
Output: Predicted Labels
Set the number of iterations epoch, loss function,
    training batch_size, optimizer, learning rate lr
    1. For epoch = 1:epoch
    2. For batch = 1:k/batch_size
    3. Subdivide the original fundus image into non-overlapping
    patches.
    4. Linearly project each patch into a fixed D-dimensional
        vector using the learnable  embedding matrix E
    5. Add a learnable embedding to the head of the embedded
        patches sequence as a classification  representation
    6. Location information is embedded in
    the patches embedding by encoding. 7.
    The generated embedding is fed to a
    Encoder.
    8. MSA operation and L-layer encoder iteration are
    performed.
    9. Calculate losses
    10. Perform back-propagation loss
    11. Update model parameters.
    12. Output model prediction results.
```

Figure 3 : Algorithm for Vision Transformer

5. Methodology

5.1 Preprocessing and Methods

The methodology for this experiment involved several structured steps to prepare, train, and evaluate the Vision Transformer (ViT) model for classifying diabetic retinopathy severity levels. One of the main aims was to be able to first replicate the results obtained by Jianfang Wu et al in [8] and then perform experiments to improve on them.

Replication of results, especially in healthcare research, is essential to ensure that findings are accurate, reliable, and broadly applicable. It validates results, reduces biases, and enhances the methodological rigor of studies, which is critical when patient outcomes depend on these findings. Replication also aids in generalizing results across different populations, forming a solid foundation for clinical guidelines, and improving patient safety by preventing unverified treatments from reaching clinical practice. By addressing the reproducibility crisis, replication strengthens the credibility of healthcare research, fosters scientific progress, and builds public trust in medical advancements.

All experiments are implemented on an AMD Ryzen 9 5900 12-core processor @ 4.46Ghz with 32GB RAM, utilizing an NVIDIA GeForce RTX 3080 Ti GPU having 16GB of GPU memory. The code is implemented in Pytorch 2.4.0, an open source deep neural network library written in Python and we use Python version 3.12.3.

The steps followed here ensured that the data was preprocessed, the model was appropriately adapted, and the results were effectively monitored, making for a robust and reproducible workflow. 1. **Preprocessing the Input Dataset**: The initial step involved preprocessing the input dataset, resizing each image to the resolution required by the ViT model. Properly sizing the images was crucial for compatibility with the model's architecture and ensuring that each input token represented the intended patch size.

2. **Data Splitting**: We separated the image database into training, validation, and test sets using Python scripts. This partitioning allowed us to train the model on one subset, validate and tune it on another, and ultimately test its generalization on unseen data, ensuring the reliability of the results.
3. **Data Augmentation**: Using PyTorch's augmentation utilities, we applied a series of transformations to the training data to enhance model generalization. Data augmentation generated diverse samples by rotating, flipping, cropping, scaling, and panning images. We also adjusted brightness, contrast, and color saturation, introducing variations that would help

the model generalize better to real-world data. This step was particularly valuable as it effectively increased the training set's size and diversity, mitigating the risk of overfitting, especially with a limited dataset.

4. **Modifying the Vision Transformer Model**: To adapt the ViT model for our specific task, we updated the final classification layer to output predictions for five severity levels. We froze the upper layers of the ViT model, which retained pretrained weights and minimized the risk of losing learned representations during fine-tuning. This transfer learning approach optimized training efficiency while allowing the model to learn task-specific features in the final fully connected (FC) layer.
5. **Evaluating Model Performance**: We calculated validation accuracy throughout training to monitor the model's ability to generalize to unseen data. Tracking metrics like validation accuracy allowed us to detect overfitting early and adjust hyperparameters as needed.
6. **Result Tracking with Neptune.ai**: Finally, we integrated Neptune.ai for seamless tracking and monitoring of experiment metrics and hyperparameters across runs. This tool enabled a clear and organized record of results, fostering consistency and ease in comparing and reproducing experiments.

Each of these steps contributed to an efficient and effective experiment setup, allowing us to train a Vision Transformer model tailored to classify diabetic retinopathy severity levels with an optimal combination of accuracy and generalization.

5.2 Final Model Implementation

After much experimentation, this is the final implementation of the model. Because of too much early overfitting, there was a necessity to add a Sequential block after the Transformer which would help the model efficiently aid with an unbalanced dataset. Adding the Sequential layer allowed customization of the classification self-attention head while keeping the backbone frozen (or fine-tuned). The order of the layers inside of the Sequential block is also very important. This order of layers is effective because it follows a structured approach to refining and optimizing the features extracted from the imported model. First, the Linear(768 → 256) layer reduces dimensionality, helping the model focus on the most relevant features while improving computational efficiency. Next, LayerNorm stabilizes the activations, ensuring balanced feature scaling and preventing issues like internal covariate shift. To enhance generalization, Dropout(0.1) randomly deactivated neurons, reducing overfitting and making the model more robust. The ReLU activation introduces non-linearity, allowing the model to learn complex patterns instead of being limited to linear transformations. Finally, the Linear(256 → 5) layer maps the refined features to the desired output space, ensuring correct classification or prediction. Together, these layers create a pipeline that extracts, normalizes, regularizes, and transforms features before making the final prediction, ultimately improving model performance and generalization.

5.3 Model Random Sampler

The **WeightedRandomSampler** is a utility in PyTorch designed to address class imbalance during the training of neural networks. It works by assigning specific sampling probabilities to each class, ensuring that both majority and minority classes are represented more evenly in each training batch. This is particularly useful when dealing with datasets where certain classes are underrepresented.

Using a **WeightedRandomSampler** helps avoid several common problems associated with imbalanced datasets. Without it, the model might generalize poorly on the minority class, as it encounters far fewer examples of it during training. Additionally, models trained on imbalanced data can show deceptively high overall accuracy by primarily predicting the majority class, while neglecting the minority classes. Another key issue it mitigates is low recall for minority classes; by sampling more instances from these classes, the model learns to better recognize and predict them, leading to improved performance and fairness across all classes.

5.4 Learning Rate Scheduler

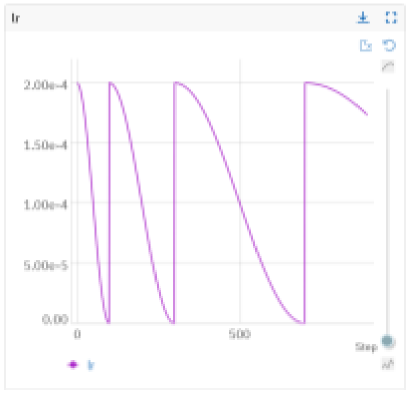


Figure 4 : Learning Rate Curve, dips and rises show scheduler achieving task

A learning rate scheduler in deep learning is a technique used to dynamically adjust the learning rate during training to optimize convergence. The learning rate is one of the most critical hyperparameters when training a neural network. If the learning rate is set too high, the model may fail to converge or oscillate without settling on an optimal solution. Conversely, if the learning rate is too low, the model may converge too slowly or get stuck in sharp local minima. By using a learning rate scheduler, the model benefits from faster and more stable convergence. Without a scheduler, the learning rate remains constant throughout training, which can lead to slow convergence or failure to find the best set of weights. A scheduler allows the model to adapt during different phases of training, improving both speed and stability. It also helps avoid local minima; initially, a high learning rate encourages exploration of various minima, while later, a lower learning rate enables the model to settle into a good solution. Additionally, starting with a high learning rate can help prevent overfitting, and gradually lowering it allows the model to fine-tune its weights effectively.

5.5 Saliency Mapping utilizing Self-Attention Mechanisms

A **saliency map** is a visualization technique used in deep learning to highlight the most important regions of an input image that the model focuses on while making predictions. It provides insights into what the model "sees" as important when making a decision.

The saliency map is computed by calculating the gradient of the output (e.g., the predicted class score) with respect to the input image pixels. The magnitude of these gradients indicates how sensitive the model's output is to changes in each pixel, thereby highlighting the regions of the image that have the most influence on the prediction.

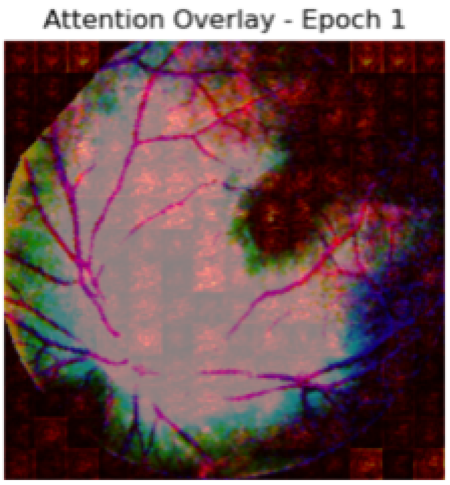


Figure 5 : Saliency Mapping Attention Overlay of Single Image

Saliency maps serve several important purposes in deep learning, particularly in enhancing model explainability. They help visualize which parts of an image contributed most to a model's prediction, providing insight into the model's decision-making process. This makes it easier to interpret how and why the model arrives at certain outputs. Saliency maps are also valuable for debugging, as they can reveal whether the model is focusing on irrelevant or incorrect regions of the image—potentially highlighting flaws in either the model architecture or the dataset itself. Additionally, they emphasize the most influential features in the input data, offering a clearer understanding of feature importance. In critical fields such as healthcare and autonomous systems, where trustworthiness is essential, saliency maps improve transparency by providing visual explanations of AI behavior.

When interpreting a saliency map image, the visualization often appears overlaid with a grid structure, especially in models like Vision Transformers where inputs are processed in patches. Bright regions on the saliency map correspond to areas of the input image that the model identified as most significant for its prediction; these regions have the largest gradients, indicating high sensitivity to the model's output. Dark regions, on the other hand, show areas that had little or no influence on the model's decision. The grid-like appearance represents the segmentation of the input image or batch, where certain images within the batch may show more pronounced highlighted regions. This can lead to higher output scores, signifying a stronger diagnosis or classification decision based on the amount of focus on these critical areas.

5.6 Model Metrics

There are many performance measurements that are applied to deep learning (DL) methods to measure their classification performance. The commonly used measurements in DL are accuracy, precision, recall and specificity. These are based on the categories in the error in classification. These can be defined as: True Positive (TP) is the number of disease images that are classified as disease. True Negative (TN) is the number of normal images that are classified as normal. False Positive (FP) is the number of normal images that are classified as disease. False Negative (FN) is the number of disease images that are classified as normal. Based on these definitions, the metrics are calculated as:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- Accuracy = (TN + TP) / (TN + TP + FN + FP)
- F1_score = 2 * precision x recall / (precision + recall)

This is a run of 1000 epochs, where our early stopping of 100 epochs triggered after 890 epochs. Now, below we look at the behavior of the model metrics measured in this run.

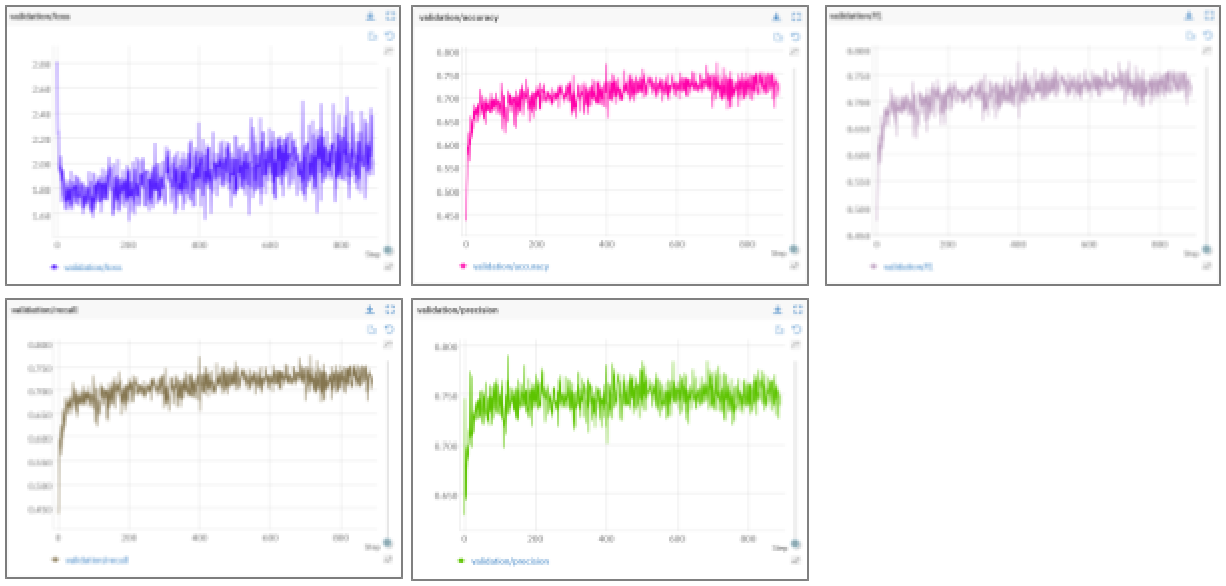


Figure 6 : Validation Accuracy, Loss, F1, Recall and Precision Scores

6. Results

This section presents an analysis of the Vision Transformer model's performance in classifying diabetic retinopathy severity given the hyperparameters selected and how they influenced the outcome. Early stopping was applied to help avoid overfitting preserving model efficiency. We first introduce all the hyperparameters and then present the model metrics and discuss the results obtained on the various metrics.

Validation loss measures how well the model's predictions align with the actual labels or target outputs for the given training dataset. In our case, the training loss starts at approximately 1.6 and sharply reduces to a steady average value of about 1.90. Since there was such a sudden change in a short amount of steps, that signifies a case of overfitting of the training data, where the model becomes too familiar with a particular set of images and isn't able to adapt to variations in unknown test data.

Validation accuracy provides an indication of the model's performance on unseen data, helping to assess its generalization ability. Since the model is getting adjusted to providing validation accuracies, the accuracy suddenly jumps from 0 to approximately 65%. Then, the model has a steady increase in the accuracy, finally staying around 75%. The highest validation accuracy that the model achieved was 77.6%. Since each epoch is different, the variation of validation accuracy values will make the values oscillate.

Recall is a metric that measures the model's ability to correctly identify all relevant instances of a particular class in the validation set. High recall indicates that the model successfully identifies a large portion of actual positives, which is critical in applications where missing a positive instance is costly (e.g., failing to detect a high-severity diabetic retinopathy case). Low recall suggests that the model may be overlooking some instances of the target class. The recall does the same action of jumping to a large value immediately and then plateauing to an average value of around 0.725, which is low to the standards of a Vision Transformer analysis.

Precision is a metric that measures the accuracy of positive predictions, showing the model's ability to correctly identify instances of a particular class without incorrectly labeling too many negatives as positives. High precision indicates that the model's positive predictions are accurate, with few false positives. Low precision suggests that the model is labeling many instances as positive incorrectly, which can lead to unwarranted interventions or actions. The jump of our precision values at the beginning again soars to around 0.700, and then plateau to around 0.750. However, the difference in the amplitude of the values around the average of 0.5 are unexpectedly high.

The **F1-score** is a metric that combines both precision and recall into a single measure, offering a balanced evaluation of a model's performance when both false positives and false negatives are critical. High F1 indicates that the model is both accurate in its positive predictions and comprehensive in capturing relevant instances.If the F1-score increases, it implies an improvement in the model's overall ability to handle both false positives and false negatives effectively. In our experiment, the F1-score also leaps up to around 0.650 in a quick manner before plateauing around a median of about 0.725 as more epochs are performed.

7. Discussion

One of the foremost goals of the effort was to try and replicate the results of the original paper [8] which provided an excellent baseline of applying vision transformer based classification techniques for diabetic retinopathy fundus images. In spite of our best efforts, our results did not completely match the results from the original paper, even though the resulting graphs of the training and validation seemed to be following the original paper. We seemed to fall short on the validation accuracy reported in the original paper, achieving about 77% accuracy instead of 92% as reported in the original paper. There were similar shortfalls in the other metrics as validation loss, precision, recall and the F1-score.

There could be several reasons for the difference in the results, which are likely due to the lack of some details on the model used in the original paper. Among some of the details of the model that was not clear were that we didn't know anything about the number of epochs that the model trained on, or that a vast amount of hyperparameters that weren't specified, or that there were more specific pre-processing performed due to the fact that the input dataset had low number of samples in the various classification categories and specifics on the overall model architecture, indicating whether just the original Vision Transformer architecture was used or whether more layers needed to be added as was true in our case. All of this information that seems pretty vital to success weren't revealed in this paper, so we tried to use the strategies discussed in this paper, but the results still fell short of the published ones in the original paper.

Some of the efforts that we performed specifically to overcome the shortcomings were as follows.

7.1 Differently Layered Models:

Before the finalized model, we had been using the original architecture for the majority of the project's duration. However, the model seemed to rise to its maximum validation accuracy too quickly, giving itself no chance to steadily improve every epoch, which is what these models typically do. After that, we then added a Sequential layer, but instead of two ReLU and Dropout layers headed by the Linear() and LayerNorm(), we had three. However, this led to epochs taking a significantly greater amount of time and very minimal improvement in our overall results. Finally, reducing the number of these layers from three to two made epochs shorter and allowed the main architecture of the model to analyze for itself.

7.2 Learning Rate Scheduler:

A learning rate scheduler in deep learning is a technique used to dynamically adjust the learning rate during training to optimize convergence. The learning rate is one of the most critical hyperparameters when training a neural network. If the learning rate is set too high, the model may fail to converge or oscillate without settling on an optimal solution. Conversely, if the learning rate is too low, the model may converge too slowly or get stuck in sharp local minima. By using a learning rate scheduler, the model benefits from faster and more stable convergence. Without a scheduler, the learning rate remains constant throughout training, which can lead to slow convergence or failure to find the best set of weights. A scheduler allows the model to adapt during different phases of training, improving both speed and stability. It also helps avoid local minima; initially, a high learning rate encourages exploration of various minima, while later, a lower learning rate enables the model to settle into a good solution. Additionally, starting with a high learning rate can help prevent overfitting, and gradually lowering it allows the model to fine-tune its weights effectively. This technique existed throughout the process, ending up in the final implementation as a more consistent rise in accuracy was found.

8. Conclusion

The experiment aimed to train a Vision Transformer (ViT) model to classify diabetic retinopathy severity levels from fundus images, evaluating the model's performance based on various metrics. The model was configured with a learning rate of 0.0002 and a batch size of 52, aiming to balance convergence stability with generalization capabilities. Early stopping was applied to prevent overfitting, resulting in training cessation at epoch 457 out of a maximum of 1000 epochs. The final model achieved a validation accuracy of 77.6%, a loss of 1.546, recall of 0.776, precision of 0.79, and an F1-score of 0.776, providing a comprehensive insight into its predictive quality.

The selected hyperparameters played a significant role in achieving these results. The learning rate of 0.0002 is modest, allowing the model to make gradual and stable improvements without risking drastic oscillations in performance metrics. This rate suited the model's architecture, as ViTs tend to benefit from smaller learning rates to optimize complex self-attention mechanisms effectively. Additionally, the moderate batch size of 160 was a pragmatic choice, given the balance it offered between computational efficiency and the quality of weight updates. This batch size also likely contributed to generalization, as larger batch sizes can lead to overfitting by making the model overly reliant on the training patterns.

Analyzing the results, the model demonstrated competent performance, with a reasonable balance between recall and precision. The recall of 0.776 indicates that the model successfully identified most positive cases, essential in medical diagnosis scenarios where under-detection (false negatives) could have serious implications. However, a precision of 0.70 suggests that while the model correctly identifies positives, there are some false positives, which may lead to unnecessary follow-ups. The F1-score of 0.675, as a harmonic mean of precision and recall, reflects a balanced performance, though room for improvement remains in harmonizing these metrics further.

Overall, the experiment illustrates that the Vision Transformer model, with the chosen hyperparameters, is effective but has areas where fine-tuning could enhance performance. Future improvements could involve experimenting with learning rate schedules, such as cosine decay or warm-up strategies, to adapt the learning rate dynamically throughout training. Additionally, slight adjustments to the batch size or use of techniques like gradient accumulation might optimize the balance between training efficiency and model generalization. These adjustments could potentially push the validation metrics higher, making the model even more robust for practical applications in diabetic retinopathy severity classification.

9. Acknowledgements

I would like to express my heartfelt gratitude to my mentor, Abdullah Rashed Ahmed, whose guidance and expertise have been invaluable throughout this project. As a high school student venturing into the world of artificial intelligence and Vision Transformer models for the first time, I faced many challenges, from understanding complex model architectures to fine-tuning hyperparameters and interpreting results.

Abdullah's patience and clarity in explaining the foundational and advanced concepts of AI have been instrumental in my learning process. Through their mentorship, I gained not only technical knowledge about Vision Transformers and their applications but also a deeper appreciation for the nuances and potential of AI in real-world applications. Their encouragement and constructive feedback helped me navigate the intricacies of model training and evaluation, transforming what initially seemed overwhelming into an engaging and rewarding experience.

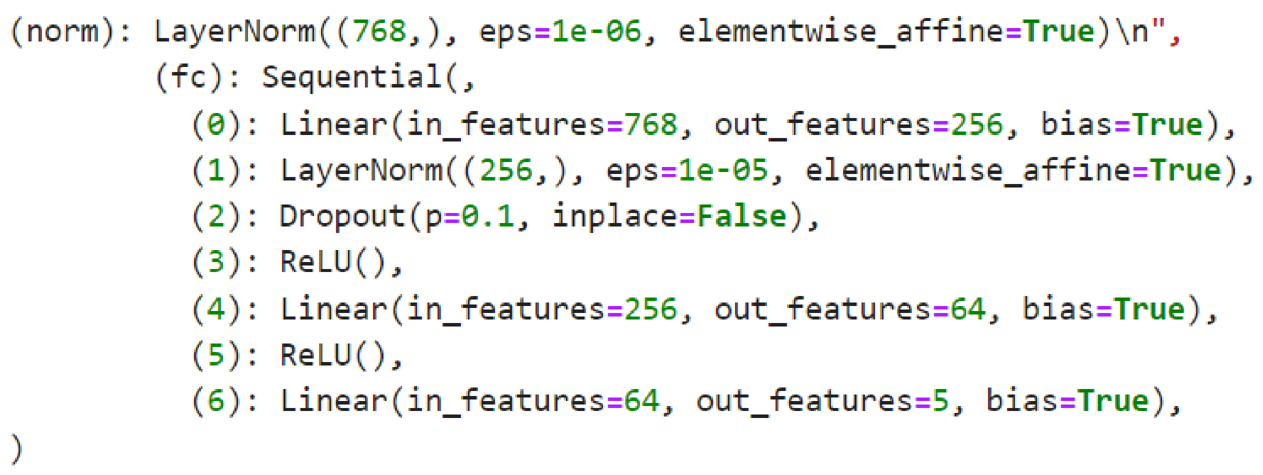
I am deeply grateful for the time, effort, and dedication Abdullah invested in supporting my growth and understanding. This experience has sparked a profound interest in AI research, and I look forward to building on this foundation in my academic and professional journey. Thank you for your unwavering support and mentorship.

10. References

1. Gadekallu TR, Khare N, Bhattacharya S, Singh S, Maddikunta PKR, Srivastava G. Deep neural networks to predict diabetic retinopathy. *J Ambient Intell Humaniz Comput*. 2020:1-14. https://doi.org/10.1007/s12652-020-01963-7.
2. Roy P, Tennakoon R, Cao K, et al. A novel hybrid approach for severity assessment of diabetic retinopathy in colour fundus images. In: 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017). IEEE; 2017:1078-1082. https://doi.org/10.1109/isbi.2017.7950703
3. Gulshan V, Peng L, Coram M, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*. 2016;316(22):2402-2410. https://doi.org/10.1001/jama.2016.17216.
4. Gulshan V, Rajan RP, Widner K, et al. Performance of a deep-learning algorithm vs manual grading for detecting diabetic retinopathy in India. *JAMA Ophthalmol*. 2019;137(9):987-993. https://doi.org/10.1001/jamaophthalmol.2019.2004.
5. Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929. 2020
6. Krishnamurthy S, Srinivasan K, Qaisar SM, Vincent PM, Chang CY. Evaluating deep neural network architectures with transfer learning for pneumonitis diagnosis. *Comput Math Methods Med*. 2021. https://doi.org/10.1155/2021/8036304.
7. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. arXiv preprint arXiv:1706.03762. 2017.
8. Vision Transformer‐based recognition of diabetic retinopathy grade, Wu J., Hu R., Xiao Z., Chen J., Liu J. Medical Physics, 2021
9. Alyoubi W.L., Abulkhair M.F., Shalash W.M. Diabetic Retinopathy Fundus Image Classification and Lesions Localization System Using Deep Learning. Sensors. 2021;21:3704. doi: 10.3390/s21113704
10. Bazi Y, Bashmal L, Rahhal MMA, Dayil RA, Ajlan NA. Vision trans-formers for remote sensing image classification. *Remote Sens*. 2021; 13(3): 516. https://doi.org/10.3390/rs13030516.

Supplementary Materials

Additional Models



```
(norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)\n",
        (fc): Sequential(,
          (0): Linear(in_features=768, out_features=256, bias=True),
          (1): LayerNorm((256,), eps=1e-05, elementwise_affine=True),
          (2): Dropout(p=0.1, inplace=False),
          (3): ReLU(),
          (4): Linear(in_features=256, out_features=64, bias=True),
          (5): ReLU(),
          (6): Linear(in_features=64, out_features=5, bias=True),
)
```

Benefits :

- Increases Model Capacity
  - Good for learning intricate patterns
  - Deep feature extraction
- ReLU Layer
  - Adding more of these layers enables non-linear decision boundaries
    - Possible for more complex classification problems
- Linear Layer
  - Transforms input data
  - Learns relationship between features
    - Bias allows for flexible shifting of activations
- Regularization
  - With proper regularization, deeper networks can achieve better generalization

Drawbacks :

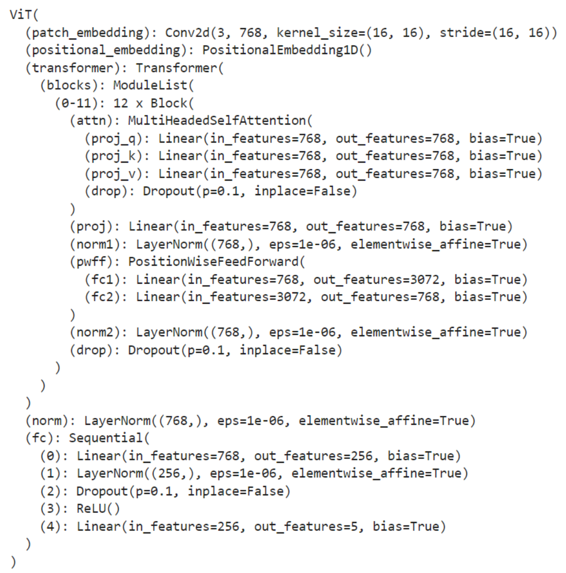
- Increased risk of overfitting
  - Poor validation performance
- Vanishing Gradient Problem
  - Early layers learn very slowly.
  - Too many ReLU layers can cause vanishing gradients
- More Layers = More Computational Cost
  - Increases parameters → more compute needed.
- Not always the best for small datasets
  - If a dataset is small, adding more layers increases the risk of overfitting without improving performance.

Pytorch code used

==

ViT(

```
(patch_embedding): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
(positional_embedding): PositionalEmbedding1D()
(transformer): Transformer(
(blocks): ModuleList(
(0-11): 12 x Block(
(attn): MultiHeadedSelfAttention(
(proj_q): Linear(in_features=768, out_features=768, bias=True)
(proj_k): Linear(in_features=768, out_features=768, bias=True)
(proj_v): Linear(in_features=768, out_features=768, bias=True)
(drop): Dropout(p=0.1, inplace=False)
)
(proj): Linear(in_features=768, out_features=768, bias=True)
(norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
(pwff): PositionWiseFeedForward(
(fc1): Linear(in_features=768, out_features=3072, bias=True)
(fc2): Linear(in_features=3072, out_features=768, bias=True)
)
(norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
(drop): Dropout(p=0.1, inplace=False)
)
)
)
(norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
(fc): Linear(in_features=768, out_features=5, bias=True.
)
```



```
ViT(
  (patch_embedding): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
  (positional_embedding): PositionalEmbedding1D()
  (transformer): Transformer(
    (blocks): ModuleList(
      (0-11): 12 x Block(
        (attn): MultiHeadedSelfAttention(
          (proj_q): Linear(in_features=768, out_features=768, bias=True)
          (proj_k): Linear(in_features=768, out_features=768, bias=True)
          (proj_v): Linear(in_features=768, out_features=768, bias=True)
          (drop): Dropout(p=0.1, inplace=False)
        )
        (proj): Linear(in_features=768, out_features=768, bias=True)
        (norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
        (pwff): PositionWiseFeedForward(
          (fc1): Linear(in_features=768, out_features=3072, bias=True)
          (fc2): Linear(in_features=3072, out_features=768, bias=True)
        )
        (norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
        (drop): Dropout(p=0.1, inplace=False)
      )
    )
  )
  (norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
  (fc): Sequential(
    (0): Linear(in_features=768, out_features=256, bias=True)
    (1): LayerNorm((256,), eps=1e-06, elementwise_affine=True)
    (2): Dropout(p=0.1, inplace=False)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=5, bias=True)
  )
)
```

Additional Datasets

Second Dataset

a. Dataset: Diabetic Retinopathy 224x224 Gaussian Filtered
b. Link:
https://www.kaggle.com/datasets/sovitrath/diabetic-retinopathy-224x224-gaussian-filtered c.

Source: Kaggle

d. Description: The images consist of gaussian filtered retina scan images to detect diabetic retinopathy. The original dataset is available at APTOS 2019 Blindness Detection

Third Dataset

a. Dataset: diabetic_retinopathy_detection

b. Link:
https://www.tensorflow.org/datasets/catalog/diabetic_retinopathy_detection c.
Source: Tensorflow

d. Description: A large set of high-resolution retina images taken under a variety of imaging conditions