DATA 603 – Principles of Machine Learning

Professor: Dr. Kemal Davaslioglu

By: Asutosh Dalei

UID: 120997754

e-Mail: asutoshd@umd.edu

Project Final Report for

Ocular Disease Classification

1. **Research Question:**
   The primary request question of this project was, how can we effectively use deep learning techniques to understand retinal images and identify the possibility of diseases a person may be having. The project sought to the detection of 3 ocular diseases, Diabetic Retinopathy, Cataract, Glaucoma and a normal retina. The project was structured to perform classification—training a deep learning model to categorize retinal images into these four classes using labeled data, which is a supervised learning approach.

   This work is important because it can enable early detection, leading to timely interventions that could improve patient outcomes. Automating retinal image analysis could also streamline processes in the healthcare industry, enhancing overall efficiency. Given that vision impairment is a significant global health issue, such a system could have a meaningful impact on public health.

2. **Dataset:**
   The nature of the data is Images, with an approximate size of about 4000 samples. The dataset is divided into 4 classes/labels, each having about 1000 samples. The raw features in this dataset are the pixel values in each of the images.

   The dataset was obtained from Kaggle, however, the contents were collected from credible sources such as, IDRiD, HRF and Ocular Recognition. These sources provide high-quality, clinically relevant retinal images to support the classification task.

3. **ML Methodology:**
   Since the task was of classification on Image based dataset, the natural choice of algorithm was a CNN based one. The project started with a basic understanding of the dataset and EDA (e.g. distribution of class samples). For the distribution of classes, the count was nearly equal for each of the 4 classes, at around 25% of the data samples. Since the dataset was pretty much clean, no treatment for missing data was performed. However, a minimal preprocessing was performed (e.g. image resizing, standardization) on the images to get them ready for the model training. The dataset was split into three sections, train, validation and test at a 70%, 15% & 15% split. For the purpose of data augmentation, techniques such as random image rotation and flipping was used. The final images were resized to (256,256).

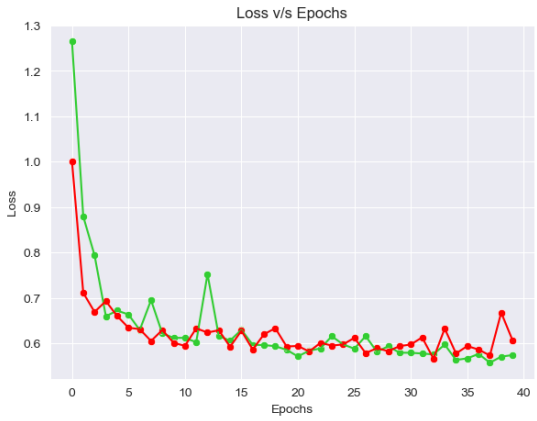
Initially, the neural network was built on a TensorFlow framework, however, midway I shifted to the PyTorch framework (due to compute issues).

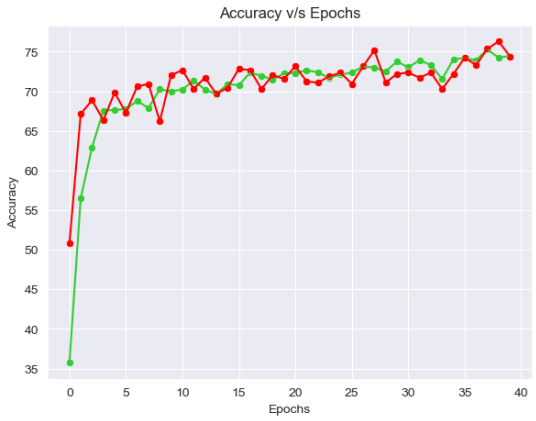
I implemented several models on both the models which underwent the training and testing on the same dataset, ensuring a common ground.
TensorFlow Models:

- A custom CNN model (32, 64, 128, 64, Flatten, 512, 4) in ReLU, with kernel size 3, stride (1,1) and MaxPool of (2,2). *Testing F1: 25%.* Notes: Training and Validation loss quickly diverged at around 10th epoch out of 30.
- A custom CNN model (16, 32, 64, 64, Flatten, 128, 32, 4) in ReLU, with kernel size 4, stride (1,1) and MaxPool of (4,4). *Testing F1: 26%.* Notes: Training and Validation loss diverged at around 25th epoch out of 30.
- AlexNet on TensorFlow (faced compute issues, shifted to PyTorch).
- ResNet transfer learning on TensorFlow (faced multiple bugs with no documentation, shifted to PyTorch)

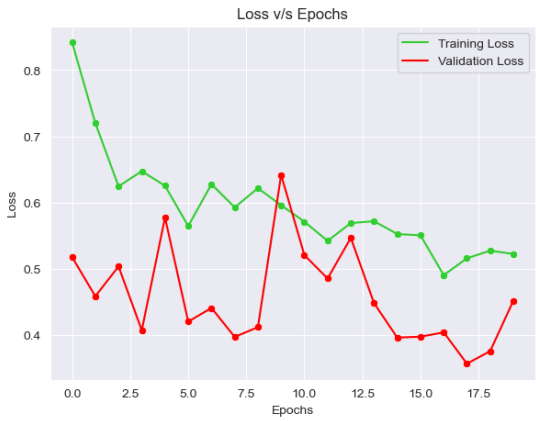
Similarly, I performed training of a models on PyTorch (which eventually became the final model):

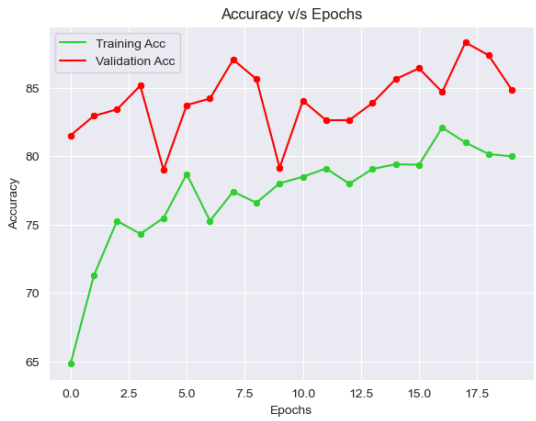
- Untrained AlexNet architecture was implemented on PyTorch. Model was trained for 40 epochs and performed exceptionally better than custom architectures at *testing F1: 72%.* The approach showed that AlexNet is a well optimized, proven design.





- Pre-Trained ResNet architecture was also implemented on PyTorch, which provided the highest accuracy of all models. The entire network was frozen except for the last classification layer (changing from 1000 classes to 4 classes). The test F1 accuracy was around 86% and is chosen to be the final model. Having been trained on a large dataset like ImageNet and implemented with sophisticated mechanisms, the model benefited from transfer learning.





4. Results:

Having trained multiple models on various custom and industry standard architectures, the model with best performance was a pre-trained ResNet which benefited from transfer learning.

However, an impressive candidate was the AlexNet implementation which was trained on the new data from scratch.

Custom models, for several reasons may not have reached the accuracy of the two models. AlexNet's proven combination of convolutional layers, ReLU activations, and dropout provides strong feature extraction and regularization. ResNet, with its residual connections, mitigates the vanishing gradient problem, enabling much deeper networks without loss of performance. Custom models may lack these well-optimized features, resulting in less effective training. Additionally, the depth and structural design of both AlexNet and ResNet are fine-tuned for better generalization, something custom models might not achieve without extensive optimization.

About computational requirements, each model training event (30 epochs) was for about 40 minutes with a GTX 1660 GPU. Increasing the model's complexity did increase the time by a bit. However, a significant improvement was noticed in shifting from TensorFlow to PyTorch, which allowed me to train the same AlexNet for more epochs in the same time frame.

	ModelName	testAccuracy	testF1	testPrecision	testRecall
0	custom1TensorFlow	0.252765	0.252312	0.252480	0.252169
1	custom2TensorFlow	0.260664	0.253997	0.256597	0.258692
2	AlexNetPyTorch_scratch	0.724684	0.716647	0.739218	0.729466
3	ResNetPyTorch_transfer	0.857595	0.851214	0.863308	0.855935

Figure 1: Model Metrics Comparison

5. Lessons Learned:

Through this project on ocular disease classification, several important lessons were learned. First, it became clear that selecting the right model architecture is crucial. Proven architectures like AlexNet and pre-trained models like ResNet significantly outperformed custom architectures, likely due to their established ability to capture hierarchical features and regularize effectively. Custom models, although flexible, struggled with issues like lack of depth, insufficient hyperparameter optimization, and overfitting, which reduced their performance. Data quality and preprocessing also proved vital—properly labeled and well-processed retinal images were key to training successful models.

Hyperparameter tuning, especially adjusting learning rates and batch sizes, also played a significant role in optimizing performance. I also learned the value of leveraging transfer learning with pre-trained models, which offered a solid foundation for improving model accuracy. Additionally, evaluating model performance using a variety of metrics, beyond just accuracy, was important to understanding its real-world application.

Overall, this project highlighted both the challenges and potential of applying deep learning in healthcare.

GitHub Repository: https://github.com/AsutoshDalei/Optic-Diseases