**HAP774 – AI IN HEALTHCARE**

**DIABETIC RETINOPATHY CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK**

**Taught By – Janusz Wojtusiak**

**Lakshmi Indira Kuppa**

**G01451480**

**Abstract**

Retinal imaging is a powerful tool for disease screening and diagnosis. With the use of Deep learning methods, there is a large scope for technology to improve the accuracy and reduce costs in thee processes. In the project, I have used a deep learning model, Inception v3 on Mobile Brazilian dataset to predict the presence or absence of diabetic retinopathy.

**Introduction**

In healthcare, imaging serves as a pivotal tool for diagnosing and monitoring different disease conditions, especially in ophthalmology. The utilization of retinal imaging and telemedicine presents a promising avenue for remote diabetic retinopathy screening. Coupled with timely treatment, these strategies play a pivotal in preventing visual impairment.

**Data Description**

The dataset contains a metadata file along with associated collection of images.

1. Labels\_mbrset.csv – This file contains patient demographics, clinical characteristics and

final\_icdr: ICDR score

* 1. 0 No retinopathy.
  2. 1 Mild non-proliferative diabetic retinopathy.
  3. 2 Moderate non-proliferative diabetic retinopathies.
  4. 3 Severe non-proliferative diabetic retinopathies.
  5. 4 Proliferative diabetic retinopathy and post-laser status.

1. **Images folder –** It contains 5164 images from 1291 diabetic patients.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class | Count | Description |  |  |  |  |  |  |
| 0 | 3750 | Normal: No abnormalities | |  |  |  |  |  |
| 1 | 272 | Mild non-proliferative DR: Microaneurysms only | | | |  |  |  |
| 2 | 568 | Moderate non-proliferative DR: More than microaneurysms but less than severe NPDR | | | | | | |
| 3 | 82 | Severe non-proliferative DR: Signs of severe NPDR | | | |  |  |  |
| 4 | 212 | Proliferative DR: Neovascularization or hemorrhages | | | |  |  |  |

**Method**

The project was executed in three distinct phases-

Phase 1 – Initial model training using TensorFlow – I used a pretrained Inception V3 model in TensorFlow to train the dataset. Additionally, I tested the model’s predictions using randomly selected images from Google and saved the model for further use.

Phase 2 – Fine-tuning for accuracy improvement – The saved model was fine tuned to enhance the performance, and the accuracies were compared with the initial training results to assess improvements.

Phase 3 – Model Comparison using PyTorch – I employed pytorch framework with a pretrained Inception V3 model. The results were analyzed and contrasted with the TensorFlow based model to evaluate their accuracies.

**PHASE 1 – Initial Model Training**

1. I imported the necessary libraries from TensorFlow for training and building the model and Pandas and NumPy for data manipulation.
2. Loaded the Csv files and assigned integer labels to the final\_icdr codes for binary classification. If the class is 0 then it is assigned Not DR, and anything else other than that is considered DR that is Presence of Diabetic retinopathy.
3. Data Preprocessing and defining parameters is done.
4. Data is split into train and validation sets.
5. Creation of the binary classification model.
6. High class imbalance is handled by computing the weights.
7. The tarin data is passed through the model. And this model is saved for further fine-tuning.
8. Accuracies are determined along with confusion matrix.

A screenshot of a computer

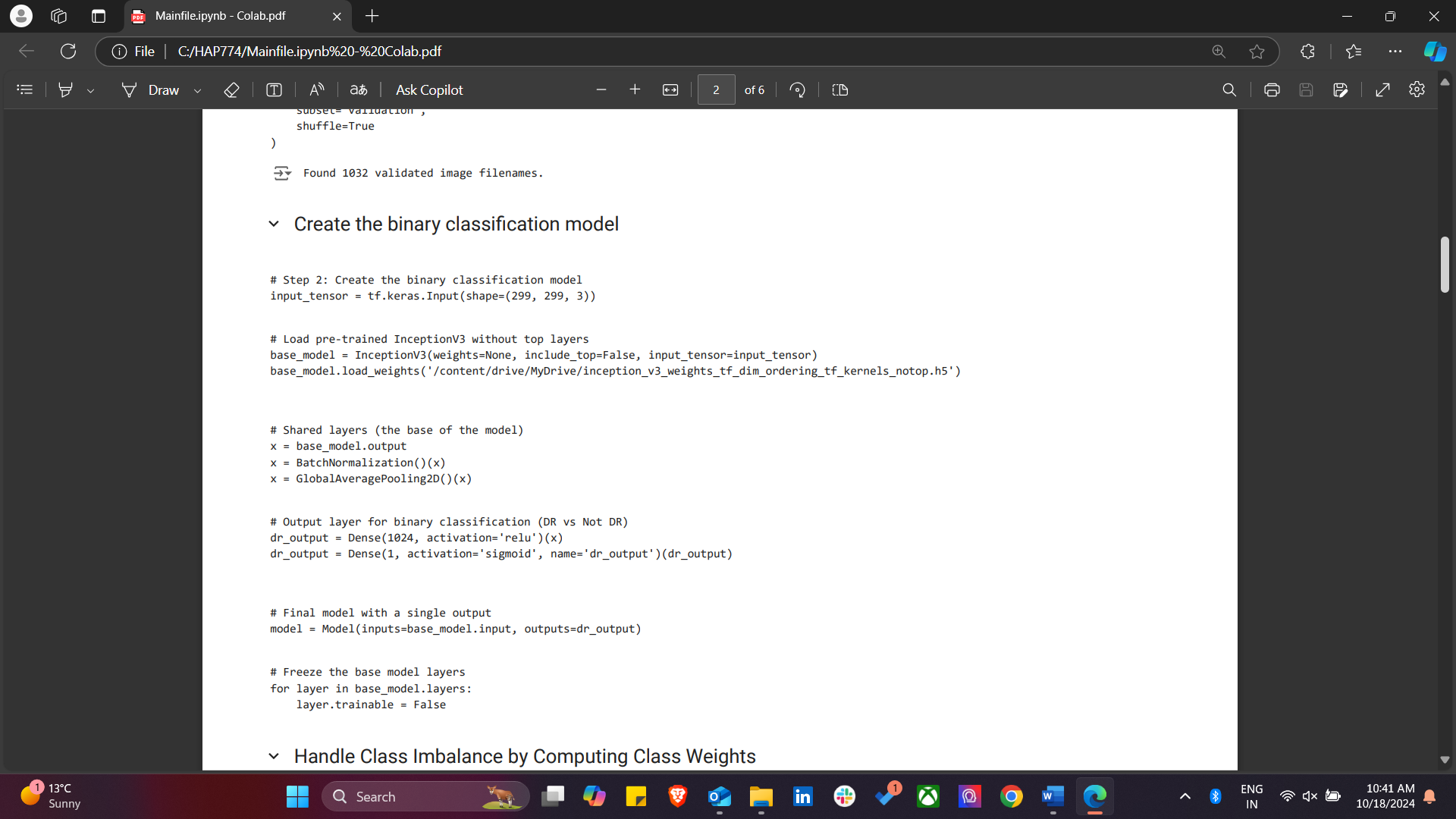
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Results -**

|  |  |
| --- | --- |
| Accuracy | 61% |

|  |  |
| --- | --- |
| Precision (Not DR) | 0.73 |

|  |  |
| --- | --- |
| Recall (Not DR) | 0.74 |

|  |  |
| --- | --- |
| F1-Score (Not DR) | 0.73 |

|  |  |
| --- | --- |
| ROC-AUC Score | 0.50 |

**TESTING WITH RANDOM IMAGES FROM GOOGLE**

Here, I assessed the model by passing random retinal images and it predicted them correctly.

A screenshot of a computer

Description automatically generated

**A computer screen shot of a computer screen

Description automatically generated**

**Phase 2 – Fine-tuned the saved model.**

1. Imported the necessary libraries.
2. Loaded the saved pretrained model.
3. Followed the similar steps from above, of splitting into train and test.
4. Training the model again.
5. Obtained results.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Results –**

Best validation accuracy – 87.50% indicating improvement in model performance. But the confusion matrix shows there was not much difference between the two models.

**Comparison between the confusion matrix of Phase 1 and Phase 2 –**

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **True Positives (Not DR)** | **True Negatives (DR)** | **False Positives (DR)** | **False Negatives (DR)** | **Accuracy** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initial Model | 548 | 200 | 79 | 205 | 61% |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fine-tuned Model | 485 | 263 | 174 | 180 | 58% |

**Phase 3 – Using PyTorch framework.**

Here, unlike the first model, I have used Pytorch framework for the same dataset.

1. Imported necessary libraries.
2. Image transformations
3. Custom Dataset Class
4. Data loading
5. Loading the pretrained Inception V3 model
6. Loss function and optimizer
7. Training the model
8. Model evaluation
9. Accuracies are obtained.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Results –**

* **Test Accuracy: 84.03%**
* **Precision: 0.79**
* **Recall: 0.60**
* **F1-Score: 0.68**

**Overall Results –**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | TensorFlow | PyTorch File (Inception V3) | | |
| Test Accuracy | 61% | 84.03% |  |  | |
| Precision | 0.73 | 0.79 |  |  | |
| Recall | 0.74 | 0.6 |  |  | |
| F1-Score | 0.73 | 0.68 |  |  | |
| ROC-AUC Score | 0.5 | N/A |  |  | |

**Key findings –**

Test accuracy – the PyTorch model performed better, achieving 84.03% accuracy, significantly higher than the 61% from the initial TensorFlow model.

Precision – The PyTorch model had a better precision (0.79) compared to TensorFlow (0.73), making it more effective at identifying true positives.

Recall – The TensorFlow model had a higher recall than the PyTorch model, indicating that it better captured actual Not DR cases.

F1 score – the TensorFlow model performed slightly better with an F1 score of 0.73, compared to 0.68 for the PyTorch model, indicating a better balance between precision and recall.

ROC-AUC – Only available for the TensorFlow model, with a score of 0.50, indicating moderate ability to distinguish between DR and Not DR cases.

**Conclusion –**

PyTorch Inception v3 model performed better than the TensorFlow model in terms of test accuracy and precision. However, the TensorFlow model had higher recall and F1-scores, which suggests it performed better at balancing between identifying DR and Not DR cases.