Spartans

Tauha Imran
Moaz farooq
Sarim Rasheed

AI Classification Model for Diabetic Retinopathy (DR)

Diabetic Retinopathy (DR) is a complication of diabetes that affects the eyes. It can lead to vision loss if not detected early. This is an AI model that classifies retinal images into different DR severity levels. Using a dataset of fundus images, we built a solution that accurately identifies whether an image indicates DR and, if so, informs of its severity level.

Github repository :
https://github.com/tauhaimran/Spartans_AI-Classifican-Model-for-Diabetic-Retinopathy--DR-

Approach and Results:

1. Introduction

Diabetic Retinopathy (DR) is a severe eye condition affecting individuals with diabetes. Early detection is crucial for preventing vision loss. This report presents an approach for DR classification using a deep learning model. The methodology includes dataset preparation, model training, front-end development for deployment, and evaluation of results.

---

2. Methodology

2.1 Dataset

The dataset used for training and evaluation consists of images labeled with different DR severity levels. The images were obtained from the Kaggle dataset: `kushagratandon12/diabetic-retinopathy-balanced`.

2.2 Model Architecture

The classification model was built using deep learning frameworks such as TensorFlow/Keras. A convolutional neural network (CNN) was designed, leveraging a pre-trained model (e.g., ResNet50, InceptionV3, or VGG16) for feature extraction and fine-tuning.

📌 Model Architecture Overview

The following table presents the architecture of the **Diabetic Retinopathy Classification Model**. It includes **convolutional layers, pooling layers, batch normalization, dropout, and dense layers**, demonstrating the structure and parameters of the neural network.

Layer (Type)	Output Shape	Parameters
Input Layer (InputLayer)	(None, 224, 224, 3)	0
Cast Layer (Cast)	(None, 224, 224, 3)	0
Conv2D (block1_conv1)	(None, 224, 224, 64)	1,792
Conv2D (block1_conv2)	(None, 224, 224, 64)	36,928
MaxPooling2D (block1_pool)	(None, 112, 112, 64)	0
Conv2D (block2_conv1)	(None, 112, 112, 128)	73,856

Conv2D (block2_conv2)	(None, 112, 112, 128)	147,584
MaxPooling2D (block2_pool)	(None, 56, 56, 128)	0
Conv2D (block3_conv1)	(None, 56, 56, 256)	295,168
Conv2D (block3_conv2)	(None, 56, 56, 256)	590,080
Conv2D (block3_conv3)	(None, 56, 56, 256)	590,080
MaxPooling2D (block3_pool)	(None, 28, 28, 256)	0
Conv2D (block4_conv1)	(None, 28, 28, 512)	1,180,160
Conv2D (block4_conv2)	(None, 28, 28, 512)	2,359,808
Conv2D (block4_conv3)	(None, 28, 28, 512)	2,359,808
MaxPooling2D (block4_pool)	(None, 14, 14, 512)	0
Conv2D (block5_conv1)	(None, 14, 14, 512)	2,359,808
Conv2D (block5_conv2)	(None, 14, 14, 512)	2,359,808
Conv2D (block5_conv3)	(None, 14, 14, 512)	2,359,808
MaxPooling2D (block5_pool)	(None, 7, 7, 512)	0
Flatten (flatten)	(None, 25088)	0
Dense (dense)	(None, 512)	12,845,568
BatchNormalization	(None, 512)	2,048
Dropout (dropout)	(None, 512)	0
Dense (dense_1)	(None, 256)	131,328
BatchNormalization	(None, 256)	1,024

Dropout (dropout_1)	(None, 256)	0
Cast Layer (Cast_23)	(None, 256)	0
Dense (Output Layer - dense_2)	(None, 5)	1,285
Total Parameters:	-	**47,755,088 (182.17 MB)**
Trainable Parameters:	-	**20,059,141 (76.52 MB)**
Non-Trainable Parameters:	-	**7,636,800 (29.13 MB)**
Optimizer Parameters:	-	**20,059,147 (76.52 MB)**

2.3 Model Training

- Data Augmentation: Techniques like rotation, flipping, and brightness adjustments were applied to increase model robustness.
- Optimizer: Adam optimizer with a learning rate scheduler.
- Loss Function: Categorical Crossentropy for multi-class classification.
- Evaluation Metrics: Accuracy, Precision, Recall, and F1-score were used for performance evaluation.

---

3. Model Training Results Analysis

🔍 Overview

The model training was conducted using a pre-trained **VGG16 model** (without top layers) with a transfer learning approach. The training process included **15 epochs** with the following performance metrics:

- **Accuracy**
- **Loss**
- **Precision**
- **Recall**

Training and Validation Data

- **Training Data:** 34,792 images across 5 classes.
- **Validation Data:** 4,971 images across 5 classes.

Downloading Pre-trained Weights

The model used weights from:

bash
`https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5`

📑 Training Progress Analysis

Epoch	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Precision	Recall	Learning Rate
1	37.78%	1.7592	53.63%	1.1042	41.02%	29.44%	1.0000e-04
3	44.54%	1.3719	52.92%	1.1071	50.18%	34.30%	1.0000e-04
5	47.32%	1.2726	55.46%	1.0668	53.34%	35.67%	2.0000e-05
7	48.37%	1.2374	55.93%	1.0599	54.60%	36.84%	2.0000e-05
10	49.55%	1.2039	55.54%	1.0536	56.56%	37.61%	4.0000e-06
12	56.25%	1.0687	55.54%	1.0542	63.64%	43.75%	4.0000e-06
15	50.01%	1.1834	55.79%	1.0513	57.00%	38.18%	8.0000e-07

🔑 Key Observations

1. **Gradual Improvement:**
    - The model started with an accuracy of **37.78%** and improved to around **50.01%** by the end of training.

    - Validation accuracy consistently stayed around **55-59%**, indicating some improvement but also suggesting underfitting or the need for further tuning.
2. **Precision and Recall:**
    - Precision and recall improved slightly over epochs but remained relatively low, implying that the model struggles to identify classes effectively.
    - Peak precision observed was around **63.64%**.
3. **Loss Reduction:**
    - Both training and validation loss decreased steadily, indicating proper learning but potentially requiring more epochs or adjustments to fully converge.
4. **Learning Rate Adjustment:**
    - The learning rate was reduced several times during training, reflecting a careful attempt to stabilize training and improve learning.
5. **Warnings Encountered:**
    - Multiple warnings were shown suggesting that `model.save()` **should use the** `.keras` **format instead of** `.h5` for better compatibility.

To avoid this warning, save the model using python
`model.save('model_name.keras')`

---

4. Conclusion

This approach demonstrates the effectiveness of deep learning for DR classification. The model successfully identifies different DR stages from retinal images with reasonable accuracy. Further improvements could include hyperparameter tuning, larger datasets, and the integration of additional image preprocessing techniques. Deployment on cloud-based platforms or mobile applications can also enhance accessibility and usability.

---

5. Future Work

- Enhancing model performance with ensemble learning.
- Implementing explainable AI techniques to interpret predictions.
- Deploying the model as a web application for wider accessibility.

End of Report