Hackathon Report for Diabetic Retinopathy Classification

TEAM EHF (ERAJ TANWEER,HUNAIZA KHAN,FAISAL SHAHID)

Introduction

This report describes the implementation of a deep learning model for **Diabetic Retinopathy Classification** using TensorFlow and Keras. The project follows a structured pipeline, including data acquisition, preprocessing, augmentation, model design, training, and evaluation. The goal is to create a robust model capable of accurately classifying retinal images.

1. Data Acquisition

The dataset used for this project is the **Diabetic Retinopathy Balanced** dataset, downloaded from Kaggle using the `kagglehub` library.

```
# Create directory.
!mkdir -p ~/.kaggle

# Copy kaggle json to directory.
!cp kaggle.json ~/.kaggle/
```

Details:

- The `kaggle.json` file contains API credentials required to access Kaggle datasets.
- The `kagglehub.dataset_download` function downloads the dataset from Kaggle.
- The dataset path is printed to verify successful download.

2. Data Preprocessing

Data preprocessing involves organizing the dataset into training, validation, and test sets.

```python
path = kagglehub.dataset_download("kushagratandon12/diabetic-retinopathy-balanced")
print("Path to dataset files:", path)
print(os.listdir(path))

# Define paths
dataset_dir = path  # Replace with your dataset directory
train_dir = '/content/train'
test_dir = '/content/test'
val_dir = '/content/val'
```

Details:

- The dataset is split into training, validation, and test sets.
- A helper function `GetDatasetSize` counts the number of images in each folder to verify dataset integrity.

3. Data Augmentation

Data augmentation improves the model's generalization ability by introducing variability in the input data.

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define batch size
BATCH_SIZE = 32
IMG_SIZE = (224, 224)

# Create an ImageDataGenerator for real-time loading and augmentation
train_datagen = ImageDataGenerator(
    rescale=1.0/255.0,  # Normalize images
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    validation_split=0.1  # 10% for validation
)
```

Details:

- **Rescaling:** Normalizes pixel values between 0 and 1.
- **Rotation, Shifting, Shearing, and Zooming:** Introduces slight variations to improve model robustness.
- **Horizontal Flip:** Adds variation by flipping images horizontally.
- **Flow from Directory:** Loads images from the dataset directory for training.\

4. Model Architecture

The model is based on a Convolutional Neural Network (CNN) with multiple layers to extract spatial features from images.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, kernel_size=3, activation='relu', input_shape=(224, 224, 3)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(64, kernel_size=3, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(256, kernel_size=3, activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

    tf.keras.layers.Conv2D(512, kernel_size=3, activation='relu', padding='same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size=(2,2)),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(1024, activation='relu')
```

Layer Details:

Layer Type	Parameters	Purpose
Conv2D	(32, 64, 128 filters)	Extracts spatial features
MaxPooling2D	Pool size = 2x2	Reduces dimensionality
Flatten	—	Converts to 1D array
Dense	512 neurons	Fully connected layer
Dropout	50%	Prevents overfitting
Dense	5 neurons	Output layer for classification

5. Compilation and Training:

The model is compiled and trained using categorical cross-entropy and Adam optimizer.

```python
model.compile(
    optimizer='Adam',
    loss="categorical_crossentropy",
     metrics=METRICS
)
```

```python
history=model.fit(
    train_generator,
    steps_per_epoch=9,
    epochs=20,
    validation_data=val_generator,
    validation_steps=1,
    verbose=1,
)
```

Details:

- **Optimizer:** Adam optimizer for fast convergence.
- **Loss:** Categorical cross-entropy for multi-class classification.
- **Metrics:** Accuracy is used to evaluate model performance.

6. Evaluation

The trained model is evaluated on the test set to measure performance.

```python
train_score = model.evaluate(train_generator, verbose= 1)
valid_score = model.evaluate(val_generator, verbose= 1)
test_score = model.evaluate(test_generator, verbose= 1)

print("Train Loss: ", train_score[0])
print("Train Accuracy: ", train_score[1])
print('-' * 20)
print("Validation Loss: ", valid_score[0])
print("Validation Accuracy: ", valid_score[1])
print('-' * 20)
print("Test Loss: ", test_score[0])
print("Test Accuracy: ", test_score[1])
```

Details:

- The model's accuracy and loss are reported.
- The results reflect the model's ability to generalize to unseen data.

7.RESULTS:

```
979/979 ━━━━━━━━━━━━━━━━━━━━ 404s 413ms/step - accuracy: 0.7993 - auc: 0.4961 - loss
109/109 ━━━━━━━━━━━━━━━━━━━━ 49s 447ms/step - accuracy: 0.8000 - auc: 0.8042 - loss:
156/156 ━━━━━━━━━━━━━━━━━━━━ 17s 112ms/step - accuracy: 0.7974 - auc: 0.6404 - loss:
Train Loss:  1.7169848680496216
Train Accuracy:  0.7991713285446167
--------------------
Validation Loss:  1.283920168876648
Validation Accuracy:  0.7999429106712341
--------------------
Test Loss:  1.6919437646865845
Test Accuracy:  0.7975863814353943
```

8. Challenges and Solutions

✅ **Class Imbalance:** The dataset was balanced to prevent bias toward any specific class.
✅ **Overfitting:** Dropout and data augmentation techniques were used to reduce overfitting.
✅ **Training Time:** GPU acceleration was used to speed up training.

9. Conclusion

The model successfully classifies diabetic retinopathy with high accuracy using CNN-based architecture. The balanced dataset, effective data augmentation, and deep learning model architecture contributed to robust performance. Further improvements could involve fine-tuning the model and increasing data diversity.