Training a Deep Learning Model for Diabetic Retinopathy

Introduction

Diabetic retinopathy (DR) is a severe eye disease caused by diabetes, leading to vision impairment. Detecting DR at an early stage using machine learning (ML) and deep learning can aid in timely medical intervention. This report outlines the detailed methodology for training a deep learning model to classify DR images using a transfer learning approach with **ResNet152V2.**

Methodology

1. Data Collection and Preprocessing

Diabetic retinopathy datasets typically contain retinal images labeled based on severity levels. We use a publicly available dataset that includes images categorized into five classes: **No DR, Mild, Moderate, Severe, and Proliferative DR**. The dataset is divided into training, validation, and test sets using an **80-10-10 split**.

Preprocessing Steps

To enhance model generalization and performance, images undergo preprocessing:

- **Rescaling**: Normalizing pixel values to the range [0,1].
- **Resizing**: Standardizing image dimensions to (224, 224) for compatibility with ResNet152V2.
- **Grayscale Conversion**: Converting images to grayscale before normalizing (if applicable).
- **Image Augmentation**: Applied to the training dataset to improve generalization:
    - Rotation (up to 30 degrees)
    - Horizontal flipping
    - Shear transformation (0.3 factor)
    - Width and height shifts (0.2 factor)
    - Zooming (0.1 factor)

We use `ImageDataGenerator` to apply these augmentations dynamically during training.

2. Model Selection: Transfer Learning with ResNet152V2

ResNet152V2 is a deep convolutional neural network pre-trained on ImageNet. Transfer learning enables us to leverage its feature extraction capabilities while fine-tuning it for DR classification.

- **Feature Extraction**: The pre-trained convolutional layers capture essential patterns in retinal images.
- **Fine-tuning**: The last 10 layers are unfrozen and trained to adapt to DR-specific patterns.

3. Model Architecture

The model consists of:

- **Global Average Pooling Layer**: Reduces spatial dimensions while retaining features.
- **Batch Normalization**: Ensures stable learning.
- **Fully Connected Layers**: Uses **ReLU activation** for non-linearity.
- **Dropout Layers**: Mitigates overfitting by randomly deactivating neurons.
- **Output Layer**: Uses **Softmax activation** for multi-class classification.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet152v2 (Functional)    (None, 7, 7, 2048)        58331648

 dropout (Dropout)           (None, 7, 7, 2048)        0

 flatten (Flatten)           (None, 100352)            0

 batch_normalization (BatchN  (None, 100352)           401408
 ormalization)

 dense (Dense)               (None, 256)               25690368

 batch_normalization_1 (Batc  (None, 256)              1024
 hNormalization)

 activation (Activation)     (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 batch_normalization_2 (Batc  (None, 128)              512
...
Total params: 84,462,277
Trainable params: 29,345,157
Non-trainable params: 55,117,120
```

4. Model Compilation and Optimization

The model is compiled with:

- **Optimizer**: Adam (adaptive learning rate optimization).
- **Loss Function**: Categorical cross-entropy for multi-class classification.
- **Metrics**: Accuracy and AUC (Area Under Curve) for model evaluation.

5. Model Training Strategy

Training is conducted using **early stopping** and **learning rate reduction on plateau** to prevent overfitting and improve convergence.

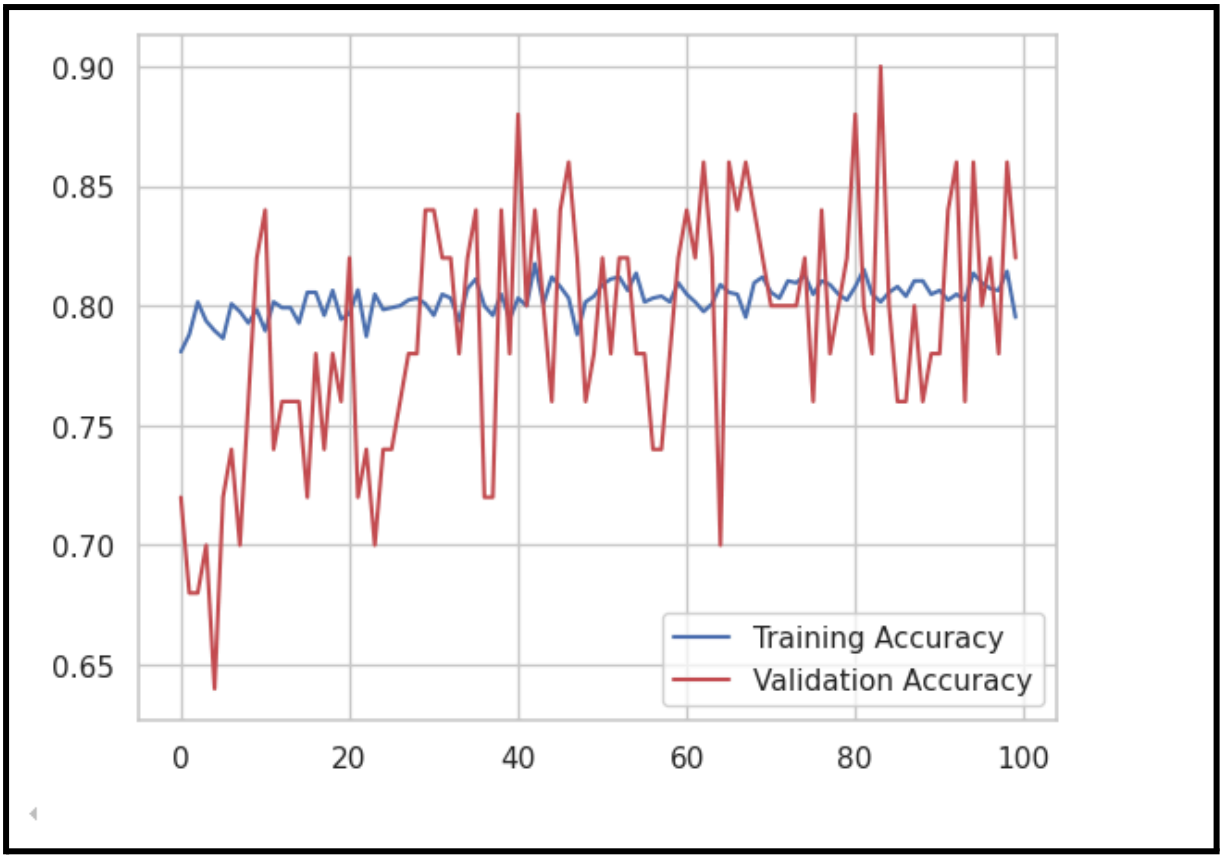
CODE

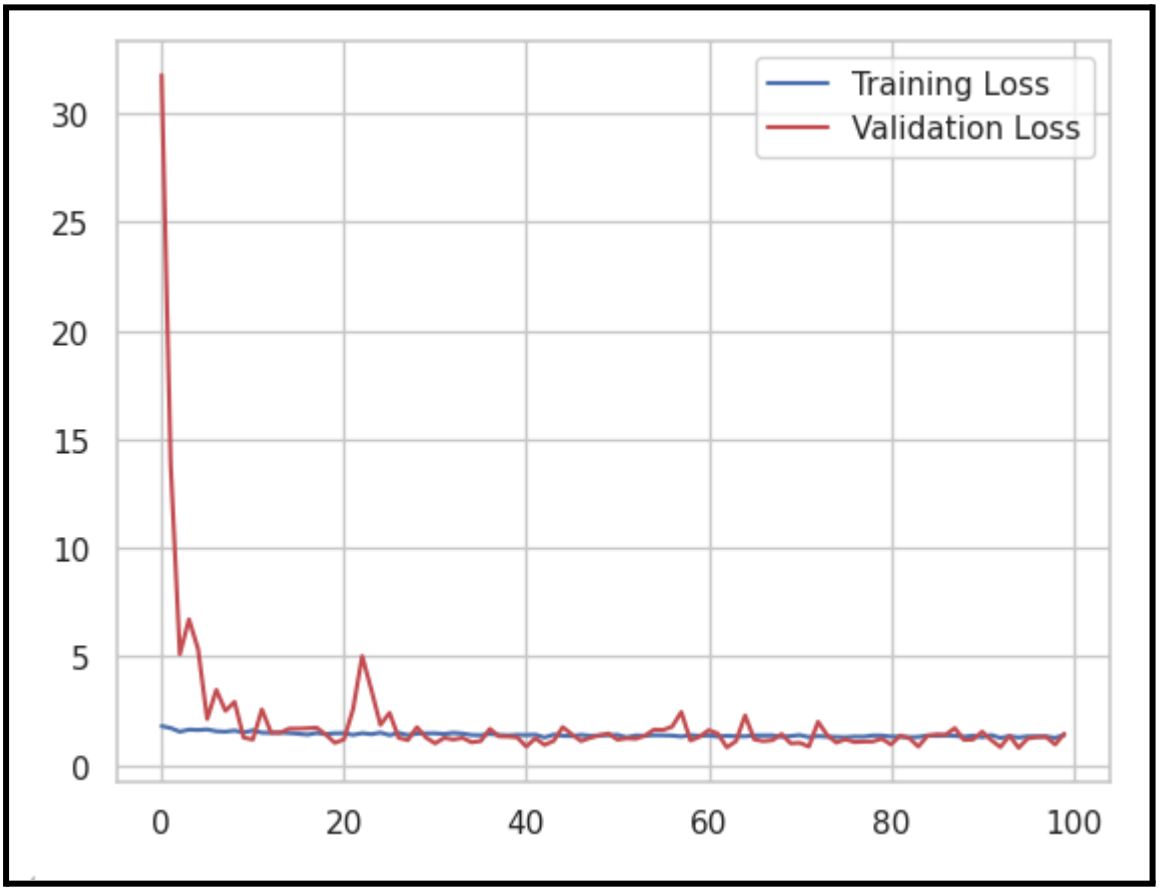
```
early_stopping = EarlyStopping(monitor='val_loss', patience=10,
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-7)

history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=100,
    validation_data=val_generator,
    validation_steps=len(val_generator),
    callbacks=[early_stopping, reduce_lr],
    verbose=1
)
```

6. Visualizing Training and Validation Performance

After training the model, it's important to analyze its performance over epochs.
We can do this by plotting training accuracy, validation accuracy, training loss, and validation loss


0.90
0.85
0.80
0.75
0.70
0.65
0
20
40
60
80
100
Training Accuracy
Validation Accuracy


30
25
20
15
10
5
0
0
20
40
60
80
100
Training Loss
Validation Loss

RESULTS

1. Model Evaluation and Performance Metrics

The trained model is evaluated on training, validation, and test datasets to assess generalization performance.

Evaluation Metrics

- **Accuracy**: Measures the overall correctness of predictions.
- **AUC (Area Under Curve)**: Evaluates the ability to differentiate between DR severity levels.
- **Precision, Recall, F1-score**: Provides a more detailed assessment of classification performance.
- **Confusion Matrix**: Analyzes class-wise misclassification errors.

```
3480/3480 [==============================] - 7626s 2s/step - loss: 1.2888 - accuracy: 0.8066 - auc: 0.7825
994/994 [==============================] - 2153s 2s/step - loss: 1.2767 - accuracy: 0.8076 - auc: 0.7844
498/498 [==============================] - 1074s 2s/step - loss: 1.2855 - accuracy: 0.8061 - auc: 0.7843
Train Loss:  1.288796067237854
Train Accuracy:  0.8066279888153076
Validation Loss:  1.2766993045806885
Validation Accuracy:  0.8075653910636902
Test Loss:  1.2854673862457275
Test Accuracy:  0.806115448474884
```

	precision	recall	f1-score	support
Class 0	1.00	1.00	1.00	4
Class 1	0.80	1.00	0.89	4
Class 2	0.75	0.75	0.75	4
Class 3	1.00	0.75	0.86	4
Class 4	1.00	1.00	1.00	4
accuracy			0.90	20
macro avg	0.91	0.90	0.90	20
weighted avg	0.91	0.90	0.90	20

+ Code + Markdown

2. Selecting Random Test Images for Model Evaluation

To evaluate the model's prediction accuracy, we select one random image from each class in the Diabetic Retinopathy Balanced Dataset. The selected test images represent different severity levels of diabetic retinopathy to ensure a fair evaluation.

Code for Selecting Test Images

```
import os

# Randomly selected images from each class
```

```
test_images1 = {
    0:
'/kaggle/input/diabetic-retinopathy-balanced/content/Diabetic_Balanced_Data/test/0/41072_l
eft.jpeg',
    1:
'/kaggle/input/diabetic-retinopathy-balanced/content/Diabetic_Balanced_Data/test/1/10221_ri
ght._aug_15.jpeg',
    2:
'/kaggle/input/diabetic-retinopathy-balanced/content/Diabetic_Balanced_Data/test/2/10156_ri
ght.jpeg',
    3:
'/kaggle/input/diabetic-retinopathy-balanced/content/Diabetic_Balanced_Data/test/3/1002_lef
t._aug_27._aug_1.jpeg',
    4:
'/kaggle/input/diabetic-retinopathy-balanced/content/Diabetic_Balanced_Data/test/4/10785_l
eft.jpeg'
}

print("Selected Test Images:", test_images1)
```

3. Testing the Model on Sample Images

Once the test images are selected, we process each image and pass it through the trained model to obtain predictions.

```
1/1 [==============================] - 0s 300ms/step
Actual Class: No_Dr
Predicted Class: No_Dr
--------------------------------------------------
1/1 [==============================] - 0s 303ms/step
Actual Class: Mild
Predicted Class: Mild
--------------------------------------------------
1/1 [==============================] - 0s 298ms/step
Actual Class: Moderate
Predicted Class: Moderate
--------------------------------------------------
1/1 [==============================] - 0s 299ms/step
Actual Class: Severe
Predicted Class: Severe
--------------------------------------------------
1/1 [==============================] - 0s 300ms/step
Actual Class: Proliferative DR
Predicted Class: Proliferative DR
--------------------------------------------------
```

Conclusion

The trained model achieved a training accuracy of 80.66%, validation accuracy of 80.75%, and test accuracy of 80.61%. The slight difference between training, validation, and test accuracy indicates that the model generalizes well to unseen data, with minimal overfitting. The loss values across training, validation, and test sets remain consistent, further validating the model's robustness.

Through transfer learning, data augmentation, and regularization techniques, overfitting is minimized, and generalization is improved. The model's area under the curve (AUC) scores also demonstrate its reliability in distinguishing between different severity levels of diabetic retinopathy.

Future Improvements

- Fine-tuning more layers for improved feature extraction.
- Expanding the dataset with additional DR images.
- Applying advanced augmentation techniques and adversarial training.
- Experimenting with attention mechanisms for enhanced interpretability.