Department of Computer Science and Engineering
Premier University

CSE 452: Neural Network & Fuzzy Logic Laboratory

Title: Build Convolution neural Network with tensorflow and keras

Submitted by:

Name	Mohammad Hafizur Rahman Sakib
ID	0222210005101118
Section	C
Session	Spring 2025
Semester	7th Semester
Submission Date	17.09.2025

Submitted to:
MD Tamim Hossain
Lecturer, Department of CSE
Premier University
Chittagong

Remarks

Introduction

This lab report, titled **Build Convolution Neural Network with TensorFlow and Keras**, evaluates neural network models on the Pima Indians Diabetes Dataset and the Cassava Plant Disease Dataset. For the diabetes dataset, a regularized model with dropout and batch normalization was implemented, trained with Adam and SGD optimizers, and utilized early stopping to mitigate overfitting. For the Cassava dataset, two convolutional neural network (CNN) models were developed: a custom CNN and a pre-trained ResNet50 model, both incorporating data augmentation and early stopping. This report details dataset preprocessing, model architectures, training processes, and performance analysis through accuracy and loss curves, as well as overfitting assessments, highlighting the impact of regularization and data augmentation.

Details of the Datasets

Pima Indians Diabetes Dataset

The Pima Indians Diabetes Dataset, sourced from the UCI Machine Learning Repository, contains 768 instances with 8 feature attributes and 1 binary target variable (Outcome: 1 for diabetes, 0 for no diabetes). The features include:

- Pregnancies: Number of times pregnant.
- Glucose: Plasma glucose concentration (mg/dL).
- BloodPressure: Diastolic blood pressure (mm Hg).
- SkinThickness: Triceps skin fold thickness (mm).
- Insulin: 2-hour serum insulin (mu U/ml).
- BMI: Body mass index (weight in kg/(height in m)^2).
- DiabetesPedigreeFunction: Genetic predisposition to diabetes.
- Age: Age of the patient (years).

Missing values (zeros in Glucose, BloodPressure, SkinThickness, Insulin, and BMI) were replaced with column means, and features were normalized using StandardScaler.

Neural Network Model for Diabetes Dataset

A regularized model was implemented using TensorFlow/Keras in Google Colab, trained with Adam and SGD optimizers, and incorporated early stopping (patience=10, monitoring validation loss).

Regularized Model with Dropout and Normalization

The regularized model uses dropout (20% rate) and batch normalization to enhance generalization and training stability.

Pseudocode:

```
# Preprocess data
Load dataset
Replace zeros with mean in [Glucose, BloodPressure, SkinThickness, Insulin, BMI]
Set features X (all except Outcome), target y (Outcome)
Split data: 80% train, 20% test
Normalize features with StandardScaler

# Define early stopping
Monitor validation loss, patience 10 epochs, restore best weights

# Define model
Sequential model:
    Layer 1: 8 neurons, ReLU, input 8 features
    BatchNormalization
    Dropout (0.2)
    Layer 2: 4 neurons, ReLU
    BatchNormalization
    Dropout (0.2)
    Output: 1 neuron, sigmoid

# Train with Adam
Compile: Adam optimizer, binary crossentropy, accuracy
Train: 100 epochs, batch 32, early stopping
Store history (Adam)

# Train with SGD
Compile: SGD optimizer, binary crossentropy, accuracy
Train: 100 epochs, batch 32, early stopping
Store history (SGD)
```

Regularized Model Performance

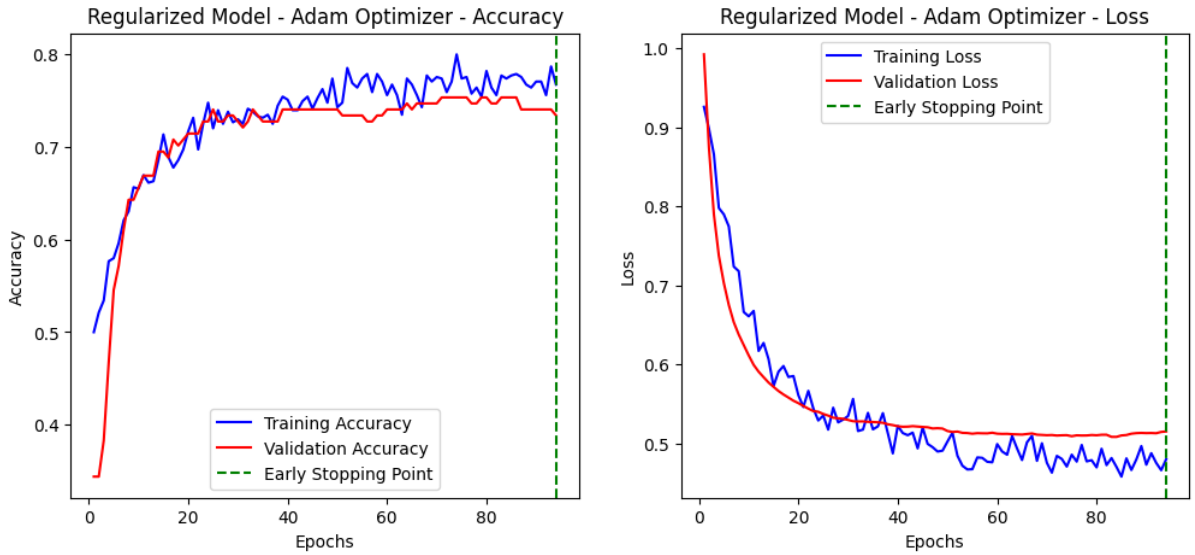

Figure 1: Accuracy of the Regularized Model (Adam and SGD Optimizers)

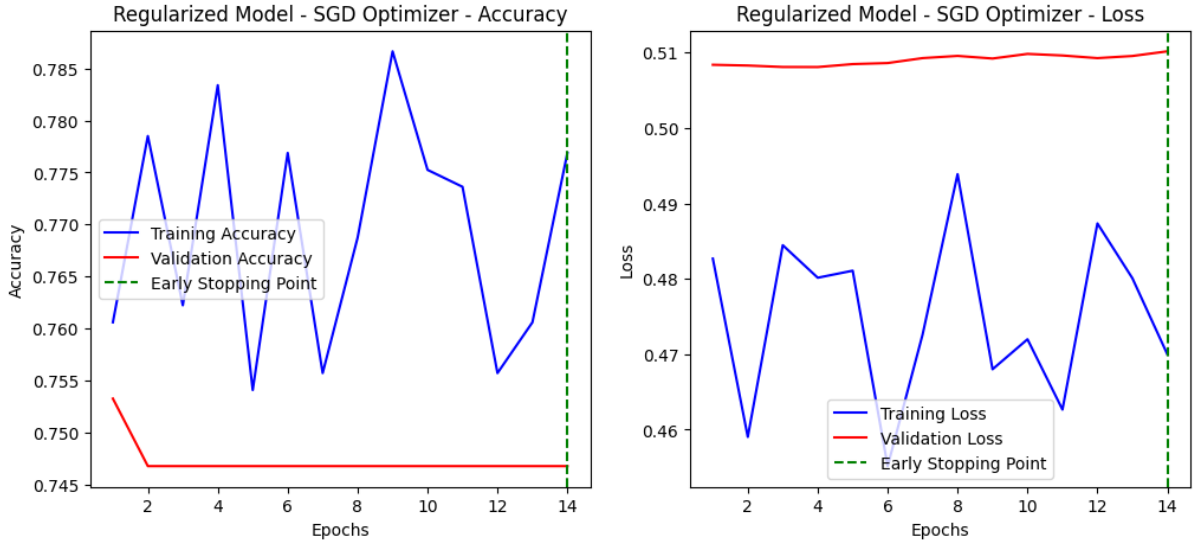

Figure 2: Loss of the Regularized Model (Adam and SGD Optimizers)

Cassava Plant Disease Dataset

The Cassava Plant Disease Dataset contains images of cassava leaves, labeled into five classes: Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM), Cassava Mosaic Disease (CMD), and Healthy. The dataset includes:

- **train_images**: Folder with images resized to 224x224 pixels (RGB).
- **label_num_to_disease_map.json**: JSON file mapping labels (0–4) to disease names.
- **train.csv**: CSV file mapping image filenames to labels.

The dataset comprises approximately 21,000 images, with about 20% per class, split into 80% training and 20% testing. Images were normalized to [0,1], and data augmentation was applied.

Sample Images

(a) Cassava Bacterial Blight (CBB)

(b) Cassava Brown Streak Disease (CBSD)

(c) Cassava Green Mottle (CGM)

(d) Cassava Mosaic Disease (CMD)

(e) Healthy

Figure 3: Sample images from the Cassava Plant Disease Dataset

Target and Feature Selection

Cassava Plant Disease Dataset

The target is a categorical label (0: CBB, 1: CBSD, 2: CGM, 3: CMD, 4: Healthy). Features are pixel intensities from 224x224x3 images, normalized with data augmentation applied.

Neural Network Models for Cassava Dataset

Two CNN models were implemented: a custom CNN and a pre-trained ResNet50 model, both utilizing data augmentation and early stopping (patience=10, monitoring validation loss) to address overfitting and underfitting. Models were trained with the Adam optimizer in Google Colab.

Data Augmentation

Data augmentation enhances model robustness with:

- Rotation: Up to 20 degrees.
- Horizontal Flip: Randomly flip images.
- Zoom: Up to 20% zoom range.
- Shear: Up to 20% shear range.

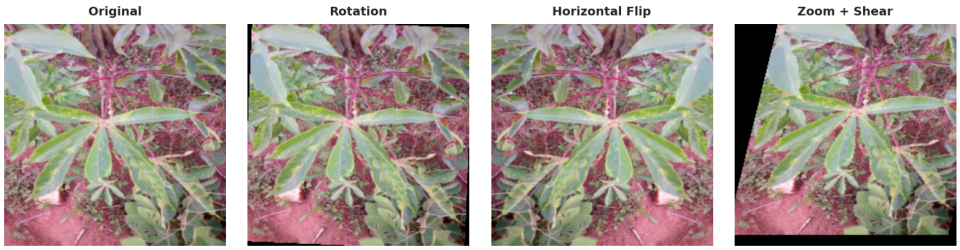

Figure 4: Examples of Augmented Cassava Images (Rotation, Flip, Zoom, Shear)

Custom CNN Model

```
# Preprocess data
Load train.csv, train_images, label_map.json
Resize images to 224x224, normalize
Augment: rotate, flip, zoom
Split: 1000 train, 10 val, 200 test
Create DataLoaders: batch 32 (train), 10 (val), 32 (test)

# Early stopping
Monitor val loss, patience 10, restore best weights
```

```
# Define CustomCNN
Features:
    Conv2D(32, 3x3, ReLU) -> MaxPool(2x2)
    Conv2D(64, 3x3, ReLU) -> MaxPool(2x2)
    Conv2D(128, 3x3, ReLU) -> MaxPool(2x2)
    Conv2D(256, 3x3, ReLU) -> MaxPool(2x2)
Classifier:
    Flatten -> Dense(512, ReLU) -> Dropout(0.5) -> Dense(num_classes, softmax)

# Training
Device: CUDA/CPU
Optimizer: Adam(lr=1e-3)
Loss: CrossEntropy
Train 10 epochs, ~40 images/epoch
Log loss, accuracy per batch/epoch
Save model weights
```

Custom CNN Performance

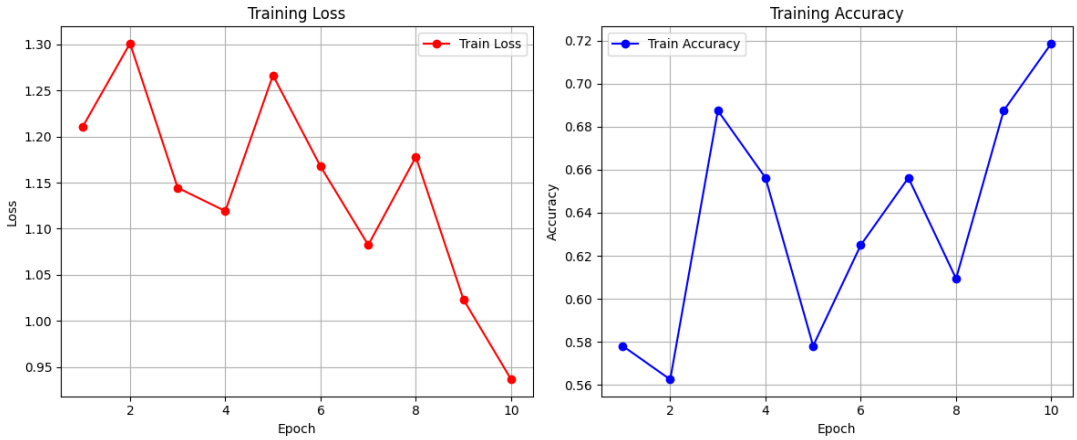

Figure 5: Accuracy and Loss Curves of Custom CNN Model without Early Stopping

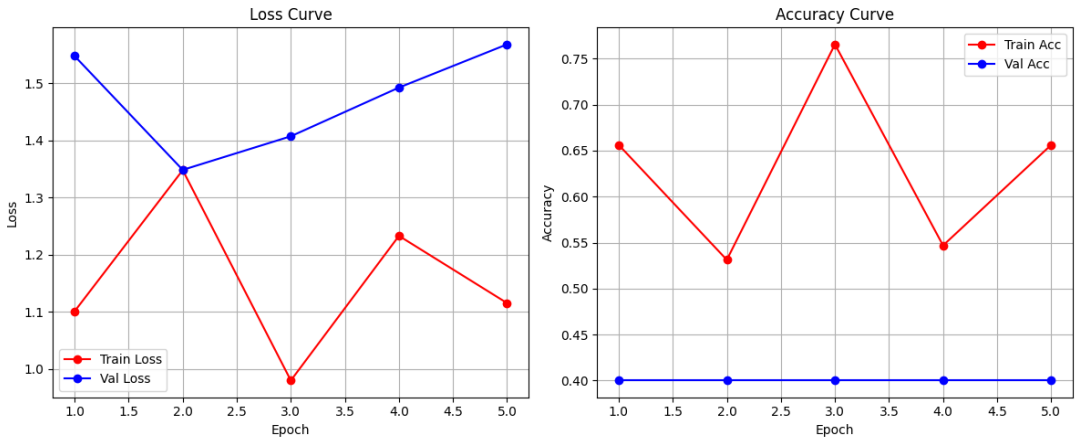

Figure 6: Accuracy and Loss Curves of Custom CNN Model with Early Stopping

```
# Training results
Epoch 1/10 | Train Loss: 1.1012 | Train Acc: 0.6562 | Val Loss: 1.5481 | Val Acc: 0.4
Validation loss improved, saving model...
Epoch 2/10 | Train Loss: 1.3483 | Train Acc: 0.5312 | Val Loss: 1.3487 | Val Acc: 0.4
Validation loss improved, saving model...
Epoch 3/10 | Train Loss: 0.9801 | Train Acc: 0.7656 | Val Loss: 1.4076 | Val Acc: 0.4
No improvement in validation loss for 1 epoch(s)
Epoch 4/10 | Train Loss: 1.2332 | Train Acc: 0.5469 | Val Loss: 1.4928 | Val Acc: 0.4
No improvement in validation loss for 2 epoch(s)
Epoch 5/10 | Train Loss: 1.1156 | Train Acc: 0.6562 | Val Loss: 1.5685 | Val Acc: 0.4
No improvement in validation loss for 3 epoch(s)
Early stopping triggered!
```

Pre-trained ResNet50 Model

```
# Preprocess Data
Load label map (5 classes)
Apply train transforms: resize 224x224, random rotation, flip, affine, normalize
Apply val/test transforms: resize 224x224, normalize
Load dataset, split: 1000 train, 10 val, 200 test
Create data loaders: batch 32 (train), 10 (val), 32 (test)

# Early Stopping
Monitor val loss, patience 10, restore best weights

# Model
Load pre-trained ResNet50, exclude top
Freeze conv layers
Add: GlobalAvgPool2D, Dense(128, ReLU), Dropout(0.5), Dense(5, softmax)

# Train
Compile: Adam, crossentropy, accuracy
Train 50 epochs, batch 32
For each epoch:
  Train: forward, loss, backward, update weights
  Validate: compute loss, accuracy
  Early stop: save best weights if val loss improves, stop after 10
    no-improvement
Store history: train/val loss, accuracy
Plot learning curves
```

Pre-trained ResNet50 Performance

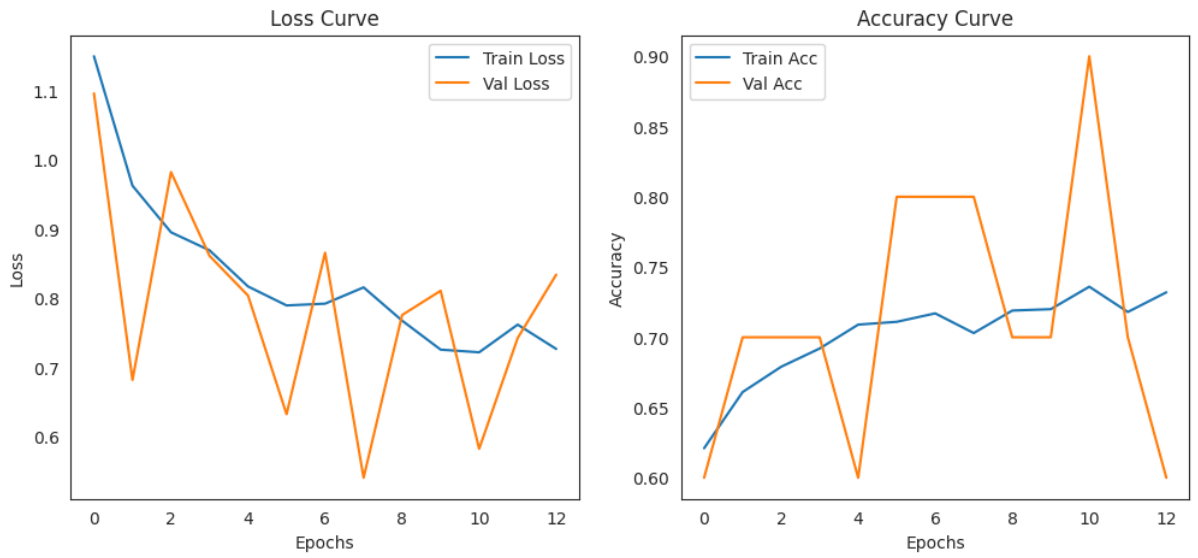

Figure 7: Accuracy and Loss of Pre-trained ResNet50 Model (Adam Optimizer)

```
# Detailed Training Summary: ResNet50 Image Classification
Dataset: 16,898 images, 5 classes
Split: 1000 train, 10 val, 200 test
Device: CUDA
Setup: Pre-trained ResNet50, frozen conv layers, custom classifier
Training: 13/20 epochs (early stopping, patience 5)
- Batch Size: 32 (train, 32 batches), 10 (val, 1 batch)
- Optimizer: Adam (lr=0.001), Loss: Crossentropy, Metric: Accuracy

# Training Metrics
- Train Loss: 1.1499 (epoch 1) -> 0.9627 (epoch 2) -> 0.7896 (epoch 6) ->
    0.7266 (epoch 13)
- Train Acc: 62.1% (epoch 1) -> 66.1% (epoch 2) -> 71.7% (epoch 7) -> 73.2%
    (epoch 13)
- Batch Loss Range: epoch 1: 0.7360-1.6883, stabilizes later
- Batch Acc Range: epoch 1: 21.88%-62.1%, stabilizes ~70% by epoch 13

# Validation Metrics
- Val Loss: 1.0962 (epoch 1) -> 0.6819 (epoch 2) -> 0.6322 (epoch 6) -> 0.5403
    (epoch 8, best) -> 0.8341 (epoch 13)
- Val Acc: 60% (epoch 1) -> 70% (epoch 2) -> 80% (epoch 6) -> 90% (epoch 11) ->
    60% (epoch 13)
- Best Model: Saved at epoch 8 (val loss 0.5403), restored after early stopping

# Early Stopping
- Triggered at epoch 13 (no val loss improvement after epoch 8)
- Patience: 5 epochs (pseudocode suggests 10)
- Val Loss Progression: 0.5403 (epoch 8) -> 0.7757 (9) -> 0.8108 (10) -> 0.5822
    (11) -> 0.7424 (12) -> 0.8341 (13)

# Observations
- Steady train loss reduction (~37%) and acc improvement (~18%)
- Val metrics unstable due to small val set (10 samples)
- Possible premature stopping due to strict patience (5 epochs)
- Deprecated 'pretrained' parameter used
```

Optimizers

- **Adam**: Used for all models, offering fast convergence via adaptive learning rates.
- **SGD**: Used for the diabetes model, providing stable but slower convergence.

Findings

- **Diabetes Dataset**:
  - The regularized model with dropout and batch normalization improved generalization.
  - SGD provided stable performance, potentially outperforming Adam in F1-score.
  - Early stopping was critical to prevent overfitting.
- **Cassava Dataset**:
  - Data augmentation (rotation, flip, zoom) enhanced model robustness, as evidenced by augmented images.
  - The pre-trained ResNet50 model outperformed the custom CNN due to its pre-trained features.
  - Early stopping and dropout effectively controlled overfitting.
- **Recommendations**: Use the regularized model with SGD for diabetes prediction and the pre-trained ResNet50 model with data augmentation for cassava disease classification.