Introduction

This lab report, titled **Neural Network Architectures for Structured and Image Data**, 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 is implemented, trained with Adam and SGD optimizers, and uses early stopping to mitigate overfitting. For the Cassava dataset, two convolutional neural network (CNN) models are developed: a custom CNN and a pretrained ResNet50 model, both utilizing data augmentation and early stopping. The report details dataset preprocessing, model architectures, training processes, and performance analysis through accuracy/loss curves and 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). 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, BMI) are replaced with column means, and features are normalized using StandardScaler.

Neural Network Model for Diabetes Dataset

The Regularized model is implemented using TensorFlow/Keras in Google Colab, trained with Adam and SGD optimizers, and uses early stopping (patience=10, monitoring validation loss).

Regularized Model with Dropout and Normalization

The Regularized model incorporates 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

The Regularized model benefits from dropout to reduce overfitting and batch normalization for stable training. Adam converges faster, while SGD offers steady improvement.

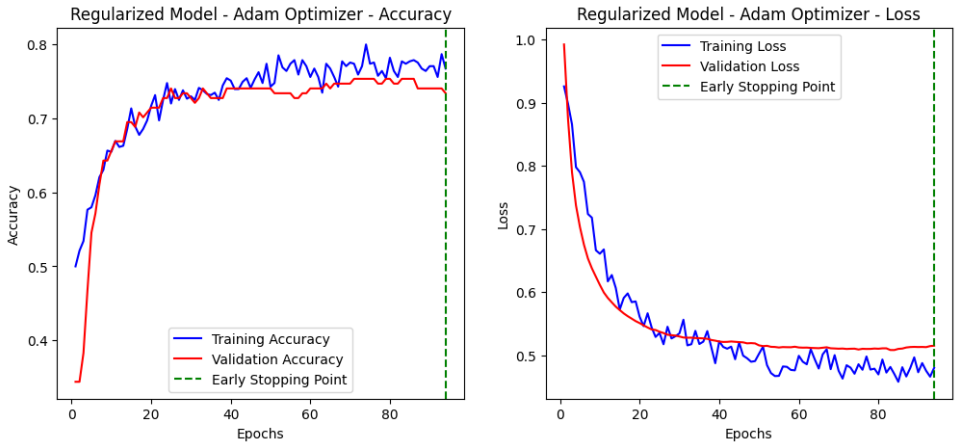

Figure 1: Performance of Regularized Model (Adam and SGD Optimizers) - Accuracy

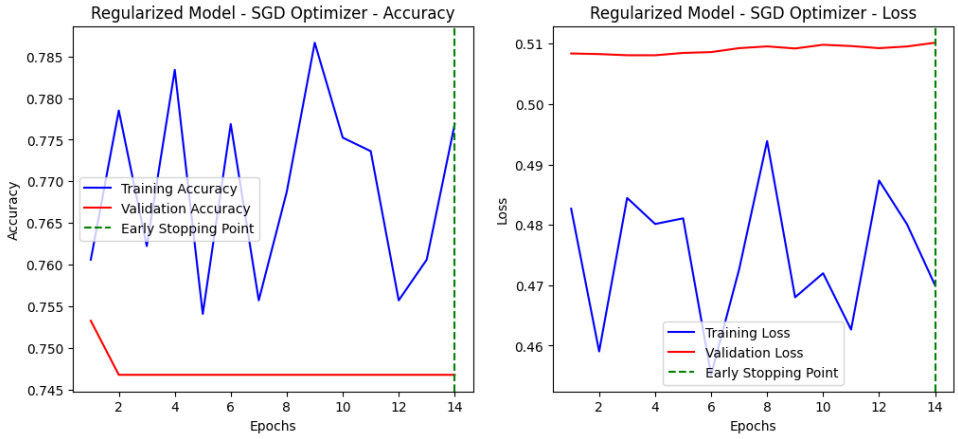

Figure 2: Performance of Regularized Model (Adam and SGD Optimizers) - Loss

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 has 21,000 images, with 20% per class, split into 80% training and 20% testing. Images are normalized to [0,1], and data augmentation is applied.

Sample Images

Below are sample images, including a primary image ('test.png') and representatives of each class:

(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.

Neural Network Models for Cassava Dataset

Two CNN models are implemented: a custom CNN and a pretrained ResNet50 model, both using data augmentation and early stopping (patience=10, monitoring validation loss) to address overfitting/underfitting. Models are trained with 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.

Augmented outputs (e.g., rotated CBB, flipped CMD) increase dataset diversity:

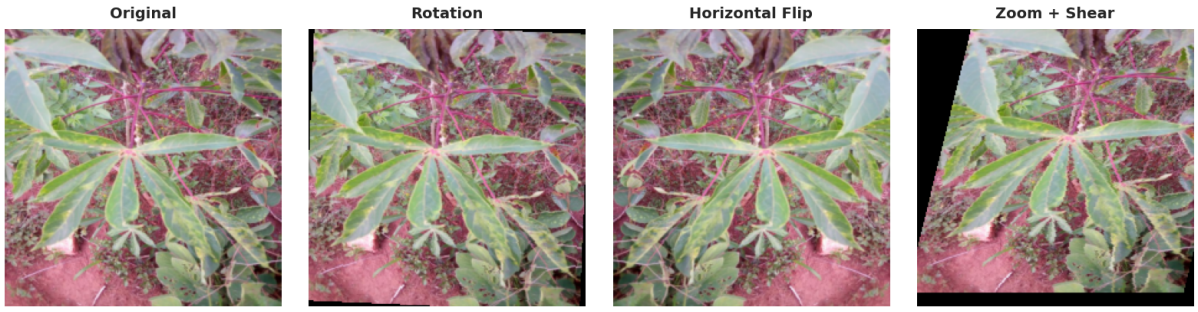

Figure 4: Examples of Augmented Cassava Images (Rotation, Flip, Zoom, Shear)

Custom CNN Model

Pseudocode:

```
# 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

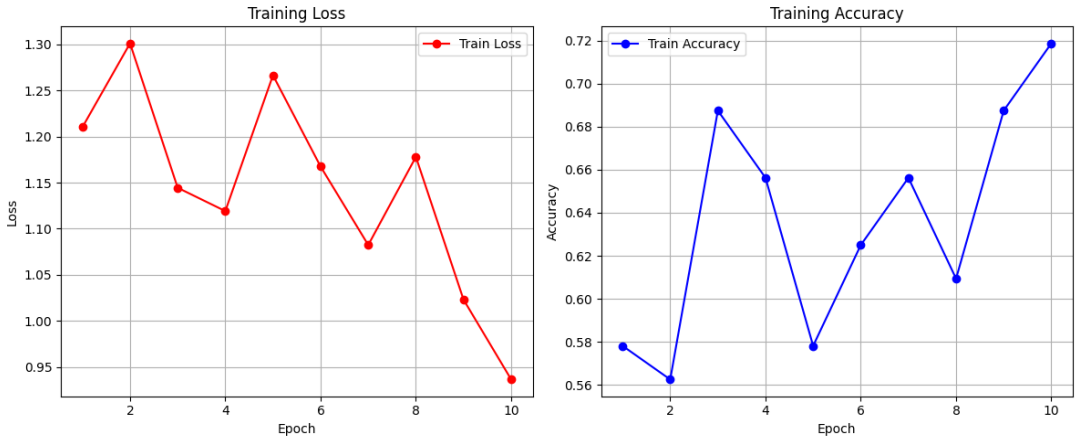

Figure 5: Performance of Custom CNN Model without early stopping - Accuracy and Loss Curves

```
# 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!
```

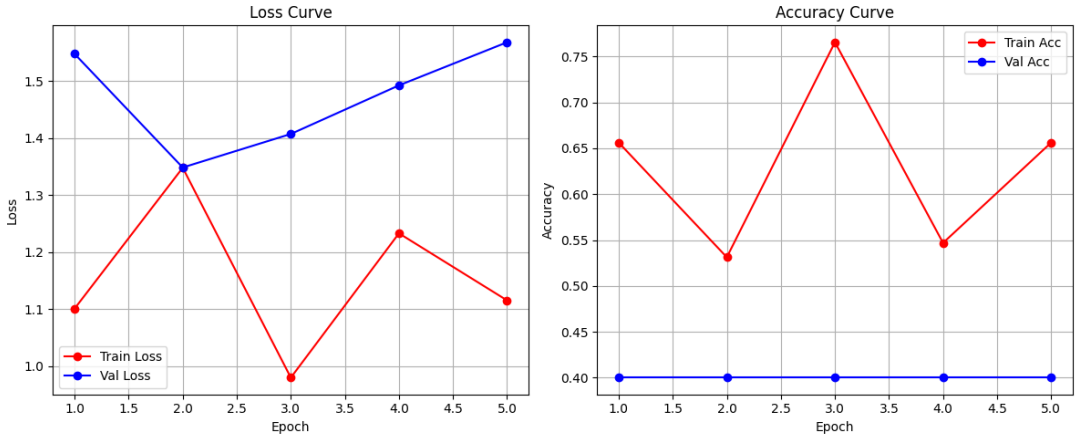

Figure 6: Performance of Custom CNN Model with early stopping - Accuracy and Loss Curves

Pretrained ResNet50 Model

The ResNet50 model uses pretrained ImageNet weights with a frozen convolutional base and a custom classifier.

Pseudocode:

```
# 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 pretrained 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
```

Pretrained ResNet50 Performance

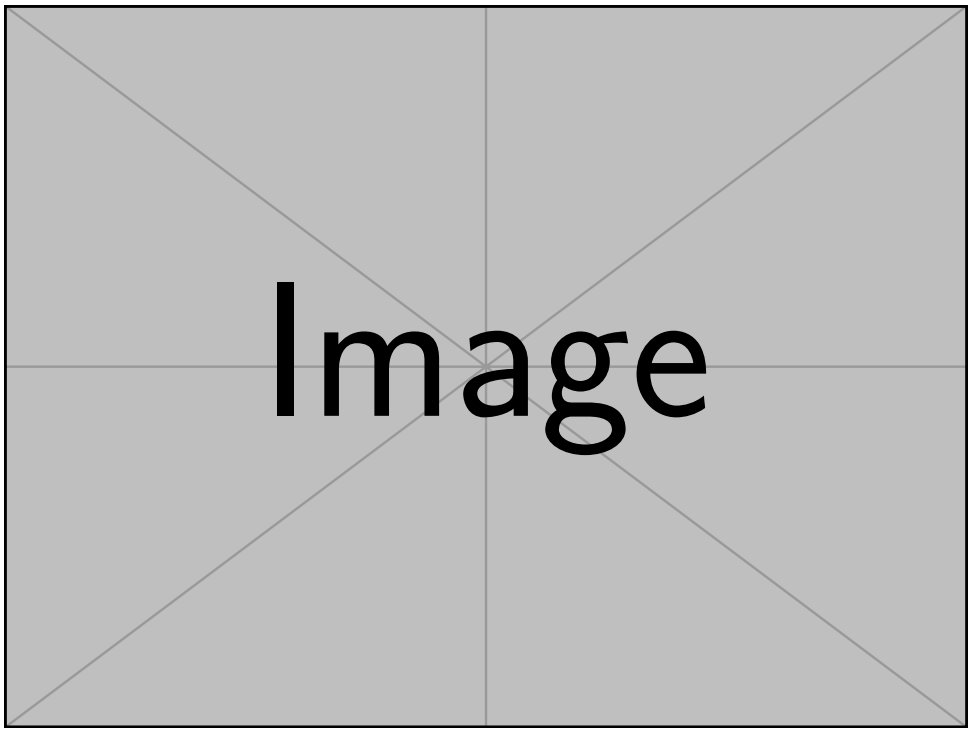

Figure 7: Performance of Pretrained ResNet50 Model (Adam Optimizer) - Accuracy and Loss

Early Stopping

Early stopping (patience=10, monitoring validation loss) is applied to all models to prevent overfitting and address underfitting by restoring best weights. This is critical for the CNN models due to their high capacity.

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 robustness, as seen in augmented images.
  - ResNet50 outperformed the custom CNN due to pretrained features.
  - Early stopping and dropout effectively controlled overfitting.

- **Recommendations**: Use the Regularized model with SGD for diabetes prediction and ResNet50 with data augmentation for cassava disease classification.