Advancements in Neural Network-based Classification for Thyroid Disease Diagnosis

Done by: Shumokh Alhattami S21107192/
Hadeel Balahmar S20106481/
Deema Hamidah S20106517
Instructor: Dr Mohammed Nauman
Course: CS4082

Problem Description

- Detecting and classifying thyroid diseases accurately from clinical data
- Addressing the challenge of subtle and varied symptoms associated with thyroid disorders
- Improving early detection and diagnosis to enhance patient care and treatment outcomes
- Leveraging machine learning techniques to augment traditional diagnostic methods

Deema

Dataset

```
[38] # Load the dataset
     data_path = 'sample_data/thyroidDF.csv'
     df = pd.read_csv(data_path)

[39] # Print the initial shape of the dataset
     print("Initial dataset shape:", df.shape)

     Initial dataset shape: (9172, 31)

[40] print(df.iloc[:, :4], df['target'])

           age sex on_thyroxine query_on_thyroxine
     0      29   F            f                  f
     1      29   F            f                  f
     2      41   F            f                  f
     3      36   F            f                  f
     4      32   F            f                  f
     ...   ...  ..          ...                ...
     9167   56   M            f                  f
     9168   22   M            f                  f
     9169   69   M            f                  f
     9170   47   F            f                  f
     9171   31   M            f                  f

     [9172 rows x 4 columns] 0       -
     1       -
     2       -
     3       -
     4       S
            ..
     9167    -
     9168    -
     9169    I
     9170    -
     9171    -
     Name: target, Length: 9172, dtype: object
```

Dataset Description: Sourced from Kaggle, the dataset features a multiclass target variable representing various types of thyroid diseases diagnosed clinically.

Deema

Dataset

Data Preprocessing

1

```
[24]
# Handling missing values
for column in df.columns:
    if df[column].dtype in ['float64', 'int64']:
        df[column].fillna(df[column].median(), inplace=True)
    elif df[column].dtype == 'object':
        df[column].fillna(df[column].mode()[0], inplace=True)
```

```
[25] # Encode categorical variables using Label Encoder
for column in df.columns:
    if df[column].dtype == 'object':
        encoder = LabelEncoder()
        df[column] = encoder.fit_transform(df[column])
```

```
[26] # Balancing skewed dataset
# Separate majority and minority classes
df_majority = df[df.target == df.target.mode()[0]]
df_minority = df[df.target != df.target.mode()[0]]

# Upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=len(df_majority),    # to match majority class
                                 random_state=123) # reproducible results

# Combine majority class with upsampled minority class
df = pd.concat([df_majority, df_minority_upsampled])

# Shuffle the dataset to avoid any order bias
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

print("New dataset shape:", df.shape)  # Print the new shape of the dataset

New dataset shape: (13542, 31)
```

2

```
New dataset shape: (13542, 31)
```

```
[27] # Separate features and target variable
features = df.drop(['target', 'patient_id'], axis=1)
labels = df['target']
```

```
[28] # Scale the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

```
[29] # Split the data into training and testing sets
trainX, testX, trainY, testY = train_test_split(features_scaled, labels, test_size=0.2, random_state=42)
```

Deema

Model selection

Why neural network?

- Chosen for its effectiveness in handling multiclass classification tasks.

- Demonstrated capability to capture complex patterns in the dataset.

- Evidenced superior performance metrics compared to other models in the research paper.

Comparison with other models:

- Demonstrated efficacy in prior studies for similar classification tasks.

- Neural network architecture deemed suitable for the problem's complexity and dataset characteristics.

Shumokh

Training and Evaluation

Model

```
✓ 0s [46] # Define the neural network
          model = Sequential([
              Dense(128, activation='relu', input_shape=(trainX.shape[1],)),
              Dense(128, activation='relu'),
              Dense(64, activation='relu'),
              Dense(len(labels.unique()), activation='softmax')
          ])
```

```
✓ 0s [47] model.compile(optimizer='adam',
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy'])
```

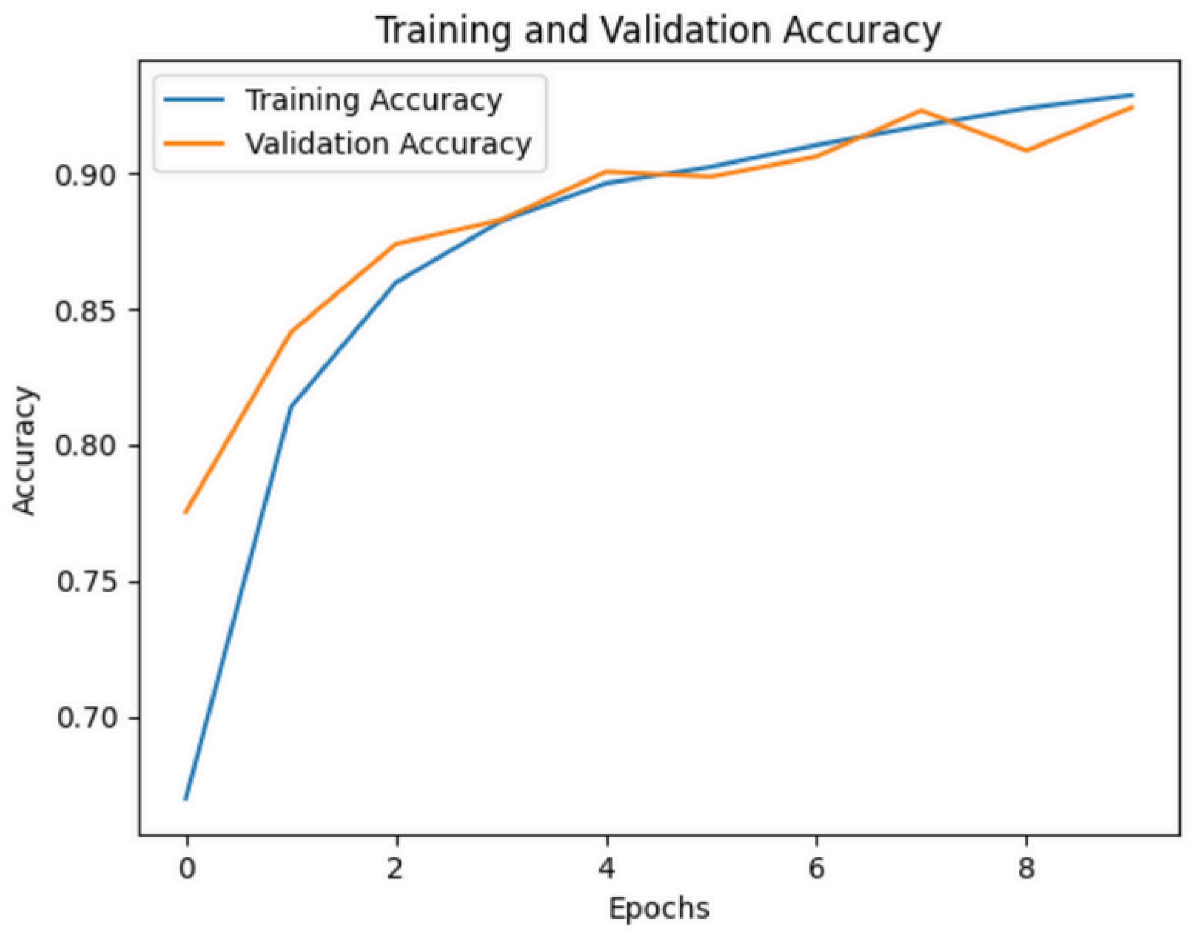
```
✓ 12s [48] # Train the model
           history = model.fit(trainX, trainY, epochs=10, validation_data=(testX, testY))
```

Shumokh

Training and Evaluation

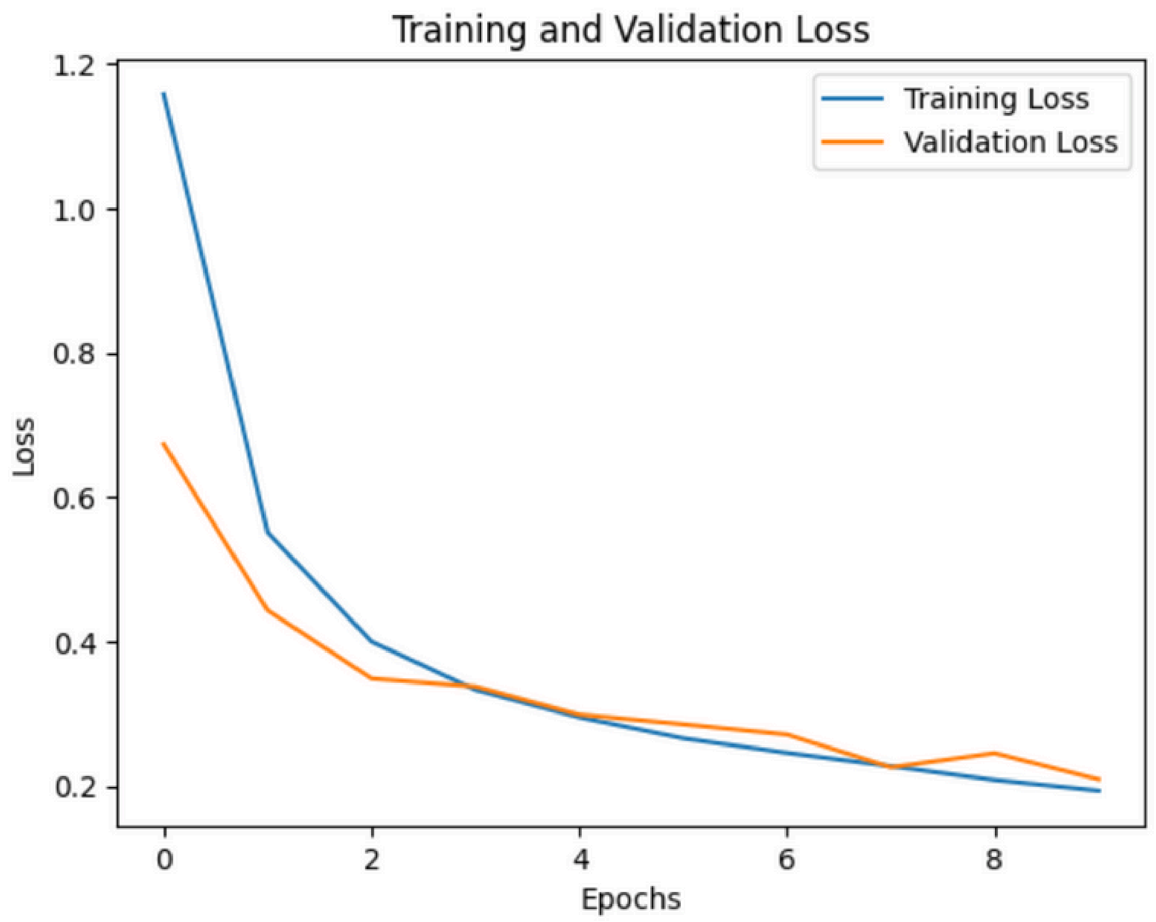
Plot Training

```python
# Plot training and validation accuracy over epochs
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()
```



Plotting Training and Validation Metrics

```python
# Plot training and validation loss over epochs
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```



Hadeel

Training and Evaluation

Confusion Metrics and Model summary

```
predictions = model.predict(testX)
predicted_classes = np.argmax(predictions, axis=1)

accuracy = accuracy_score(testY, predicted_classes)
precision = precision_score(testY, predicted_classes, average='weighted')
recall = recall_score(testY, predicted_classes, average='weighted')
f1 = f1_score(testY, predicted_classes, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
```

```
85/85 [==============================] - 0s 2ms/step
Accuracy: 0.9239571797711332
Precision: 0.9267401822828139
Recall: 0.9239571797711332
F1-score: 0.9233015582302116
```

```
[36] model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 128)               3840

 dense_5 (Dense)             (None, 128)               16512

 dense_6 (Dense)             (None, 64)                8256

 dense_7 (Dense)             (None, 32)                2080

=================================================================
Total params: 30688 (119.88 KB)
Trainable params: 30688 (119.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Hadeel

Interpretation of Results

Our study demonstrates the effectiveness of neural networks for thyroid disease classification while highlighting the need for future work to explore the impact of additional features on model performance.

Hadeel

Thank you for your attention