Machine Learning Lab (AI332L)			
Class:	BS Artificial Intelligence	**Semester:**	5th (Fall 2023)
Instructor:	Dr. M. Ehatisham-ul-Haq	**Email:**	ehtisham@mail.au.edu.pk

|Lab 12|

Building Neural Networks in Python

Lab Objective:

This lab tutorial introduces you to build a simple neural network in Python using TensorFlow. The tutorial will guide you through the entire process of building/implementing and evaluating a neural network.

Introduction to TensorFlow and Keras

TensorFlow is an open-source machine learning framework developed by the Google Brain team. It is widely used for various machine learning and deep learning tasks, including neural networks, natural language processing, computer vision, and more. **TensorFlow** provides a comprehensive set of tools and libraries for building and deploying machine learning models.

Keras is an open-source high-level neural networks API written in Python. It is designed to be user-friendly, modular, and extensible. Originally developed as a standalone library by François Chollet, Keras has now become an integral part of the **TensorFlow** project and serves as **TensorFlow's** official high-level API.

Building a Neural Network Model

Importing Libraries:

```
# importing tensorflow
import tensorflow as tf
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay

import seaborn as sns
import matplotlib.pyplot as plt
```

Importing and Pre-Processing Dataset:

We will use a publically available dataset, i.e., UCI Heart Disease Prediction, as an example to build and evaluate a simple neural network. The dataset consists of a mix of 13 numerical and categorical features related to human health. The dataset distinguishes the presence of heart disease (values 1, 2, 3, 4) from absence (value 0).

```
# URL of the UCI Heart Disease Prediction Dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-
disease/processed.cleveland.data"

# Listing the feature names in the dataset
feature_names = ["age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
"thalach", "exang", "oldpeak", "slope", "ca", "thal", "target"]

# Reading the features data from the .csv file through the specified URL
data = pd.read_csv(url, names=feature_names, na_values="?")

# Handling missing values: simply dropping rows with missing values
data = data.dropna()

# Convert the target variable to a binary variable (1 for heart disease, 0 for no
heart disease)
data["target_binary"] = data["target"].map(lambda x: 1 if x > 0 else 0)

# Listing the names of categorical features in the dataset
categorical_features = ["sex", "cp", "fbs", "restecg", "exang", "slope", "ca",
"thal"]

# Applying one-hot encoding to categorical features
encoded_data = pd.get_dummies(data, columns=categorical_features)


# Separate features and target variable
X = encoded_data.drop(["target", "target_binary"], axis=1)

# Assigning "target_binary" feature as output variable
y_binary = data["target_binary"]


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.3,
random_state=1)

# Standardize numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Building Neural Network Model:

Building the feed-forward neural network model with a single hidden layer with 64 neurons and ReLU activation**.** The output layer has just one neuron with sigmoid activation due to binary classification

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu',
    input_shape=(X_train_scaled.shape[1],)),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Compiling Neural Network Model:

```
# Compiling the model with Adam optimizer, cross entropy as loss, and accuracy as the metric to be optimized during training
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
batch_size = 32
epochs = 10
model.fit(X_train_scaled, y_train, batch_size=batch_size, epochs=epochs)
```

Evaluating Neural Network Model:

```
# Evaluate the model on the test set
loss_and_metrics = model.evaluate(X_test_scaled, y_test)
print('Loss = ',loss_and_metrics[0])
print('Accuracy = ',loss_and_metrics[1])
```

Plotting Confusion Matrix:

```
# Generate and plot the confusion matrix
y_pred = model.predict(X_test_scaled)
y_pred_binary = (y_pred > 0.5).astype(int)  # Convert probabilities to binary predictions
conf_matrix = confusion_matrix(y_test, y_pred_binary)

# Plot the confusion matrix using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",xticklabels=["Predicted: No Heart Disease", "Predicted: Heart Disease"],yticklabels=["Actual: No Heart Disease", "Actual: Heart Disease"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

Lab Task:

1. **Explore different hyperparameter values of the NN model in the above example and compare the result.**
2. **Use the above UCI Heart Disease dataset for multi-class classification problem and evaluate the performance the NN model with different hyperparameters.**
3. **Use a multi-class image dataset of your own choice, apply neural network model over it, and evaluate the classification performance.**
