LAB No. 5

Implementation of Artificial Neural Network

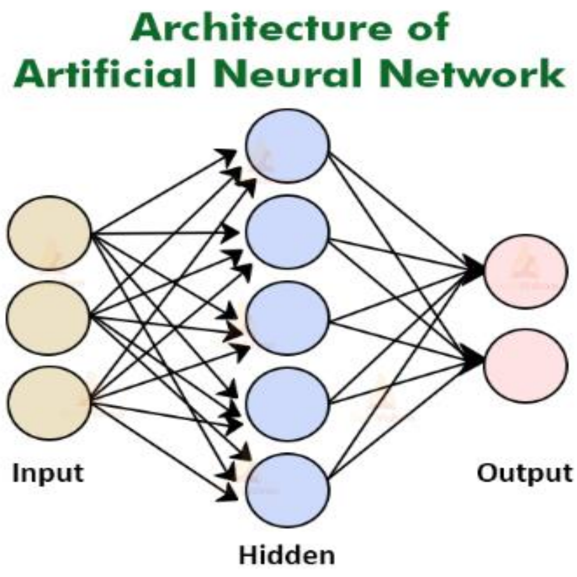
In this lab, students will learn the fundamentals and implementation of an Artificial Neural Network (ANN) for classification and prediction tasks. The lab introduces how neural networks are inspired by the human brain and how they learn patterns from data using layers of interconnected neurons. Students will first apply ANN on a small dataset to understand its working, and then implement ANN on a real-world dataset using Python libraries such as TensorFlow / Keras and Scikit-learn. Model performance will be evaluated using accuracy metrics.

Introduction

An **Artificial Neural Network (ANN)** is a supervised machine learning model composed of interconnected processing units called **neurons**. ANNs consist of:

- **Input layer** – receives input features
- **Hidden layer(s)** – performs computations using weights and activation functions
- **Output layer** – produces final prediction

Each neuron computes a weighted sum of inputs and applies an **activation function** (such as ReLU or Sigmoid). The network learns by adjusting weights using **backpropagation** to minimize error.



Key Concepts:

- **Weights & Bias** – control neuron behavior

- **Activation Functions** – introduce non-linearity
- **Loss Function** – measures prediction error
- **Optimizer** – updates weights (e.g., Adam)

ANNs are widely used in applications such as image recognition, speech processing, medical diagnosis, and pattern recognition.

Solved Examples:

Example 1:

Build a simple ANN to predict whether a student **passes or fails** based on study hours and attendance.

Solution:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Dataset
data = {
  'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
  'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
  'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
  df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']].values
y = df['Result'].values

# Build ANN model
model = Sequential()
model.add(Dense(4, activation='relu', input_shape=(2,)))
model.add(Dense(1, activation='sigmoid'))

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Train model
model.fit(X, y, epochs=100, verbose=0)

# Prediction
prediction = model.predict([[1, 1]])
print("Predicted Result (Pass=1, Fail=0):", int(prediction[0][0] > 0.5))
```

Question

Apply ANN to classify Iris flowers into three species.

Solution:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encode target
y = to_categorical(y)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)

# Build ANN
model = Sequential()
model.add(Dense(10, activation='relu', input_shape=(4,)))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))


# Compile and train
model.compile(optimizer='adam',                    loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=100, verbose=0)

# Evaluate
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Test Accuracy:", accuracy)
```

Example 3:

ANN for Handwritten Digit Classification (MNIST – Baseline) to classify handwritten digits (0–9).

Solution:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()

# Normalize and reshape
X_train = X_train.reshape(-1, 28*28) / 255.0
X_test = X_test.reshape(-1, 28*28) / 255.0

# One-hot encode labels
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Build ANN
model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(784,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Compile and train
model.compile(optimizer='adam',                    loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=128, verbose=1)

# Evaluate model
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)
```

Comparison Summary

Aspect	ANN
Handles Non-linearity	Yes
Suitable for Images	Limited
Training Time	Moderate
Interpretability	Low

LAB Assignment No. 5

Topic: Implementation of Artificial Neural Network

Question 1

Logic Gates with Neural Network. Implement a feed-forward neural network to learn the AND gate.

- Inputs: (0,0), (0,1), (1,0), (1,1)
- Output: 0, 0, 0, 1
  Tasks:

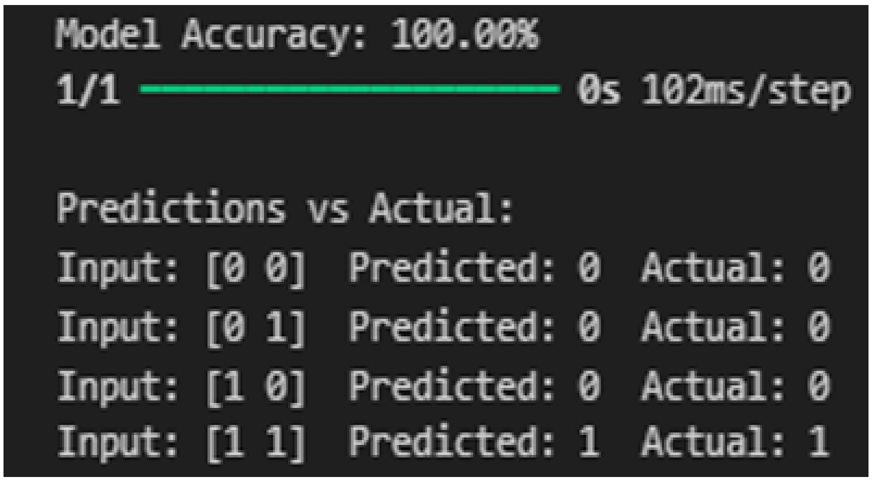
1. Create dataset using NumPy or pandas.
2. Build a neural network with one hidden layer using TensorFlow/Keras or PyTorch.
3. Train it and show accuracy.
4. Compare model predictions with actual outputs

Code:

```
import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

X = np.array([[0,0], [0,1], [1,0], [1,1]])

y = np.array([[0], [0], [0], [1]])

model = Sequential()

model.add(Dense(4, input_dim=2, activation='relu'))  neurons

model.add(Dense(1, activation='sigmoid'

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X, y, epochs=500, verbose=0)
```

```
loss, accuracy = model.evaluate(X, y, verbose=0)

print(f"Model Accuracy: {accuracy*100:.2f}%")

predictions = model.predict(X)

predicted_classes = (predictions > 0.5).astype(int)

print("\nPredictions vs Actual:")

for i in range(len(X)):

    print(f"Input: {X[i]}  Predicted: {predicted_classes[i][0]}  Actual: {y[i][0]}")
```

Output:



```
Model Accuracy: 100.00%
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step

Predictions vs Actual:
Input: [0 0]  Predicted: 0  Actual: 0
Input: [0 1]  Predicted: 0  Actual: 0
Input: [1 0]  Predicted: 0  Actual: 0
Input: [1 1]  Predicted: 1  Actual: 1
```

Question 2

Create a dataset $y = x^2$ + noise for x in range [-3,3]. Regression Task with Neural Network

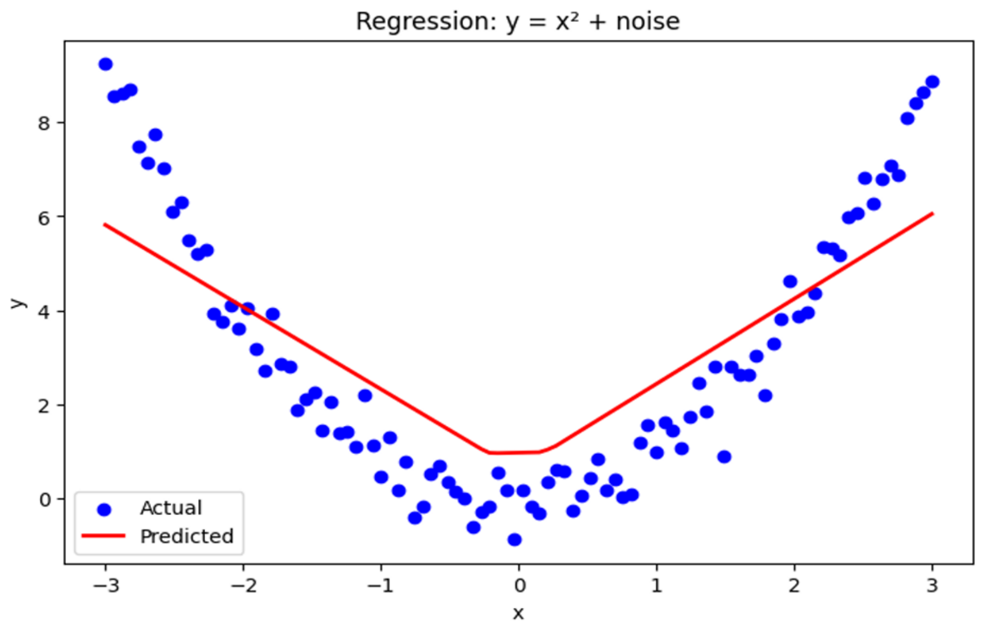
Tasks:

1. Generate 100 samples.
2. Build a neural network to predict y from x.
3. Plot actual vs. predicted results.
4. Discuss how increasing hidden neurons changes results.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
np.random.seed(42)
X = np.linspace(-3, 3, 100)
y = X**2 + np.random.normal(0, 0.5, X.shape)   # Add Gaussian noise
# Reshape for Keras (expects 2D inputs)
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)
model = Sequential()
model.add(Dense(10, input_dim=1, activation='relu'))   # Hidden layer (10 neurons)
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200, verbose=0)
y_pred = model.predict(X)
plt.figure(figsize=(8,5))
plt.scatter(X, y, label='Actual', color='blue')
plt.plot(X, y_pred, label='Predicted', color='red', linewidth=2)
plt.title("Regression: y = x2 + noise")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```

Output:



Question 3:

Use the XOR gate and train networks with different activation functions (sigmoid, tanh, ReLU).

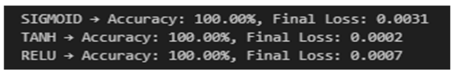
- Compare accuracy, loss, and convergence speed.
- Plot and discuss results.

Code:

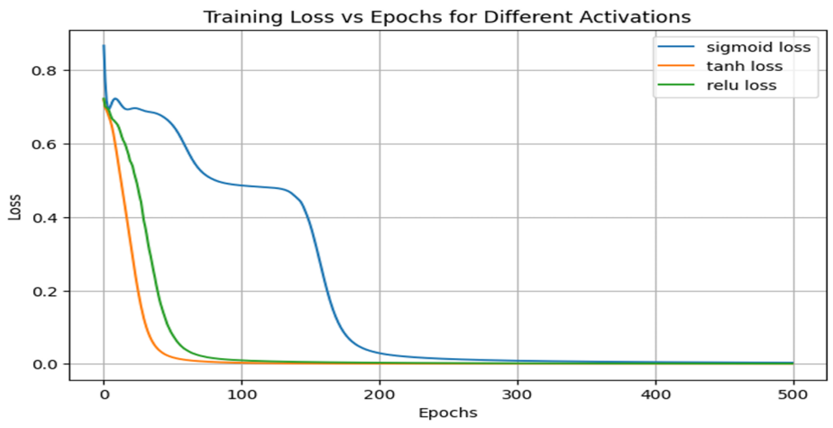
```
import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import Adam

X = np.array([[0,0], [0,1], [1,0], [1,1]])

y = np.array([[0], [1], [1], [0]])

def train_xor_model(activation, epochs=500):  model = Sequential([Dense(4,
input_dim=2, activation=activation),  Dense(1, activation='sigmoid') ])
```

```
  return model, history, loss, acc

activations = ['sigmoid', 'tanh', 'relu']

results = {}

for act in activations:model, history, loss, acc = train_xor_model(act)

results[act] = {'loss': loss, 'acc': acc, 'history': history}

    print(f"{act.upper()} → Accuracy: {acc*100:.2f}%, Final Loss: {loss:.4f}")

plt.figure(figsize=(8,5))

for act in activations:plt.plot(results[act]['history'].history['loss'], label=f'{act} loss')

plt.title("Training Loss vs Epochs for Different Activations")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.grid(True)

plt.show()
```

Output:



```
SIGMOID → Accuracy: 100.00%, Final Loss: 0.0031
TANH → Accuracy: 100.00%, Final Loss: 0.0002
RELU → Accuracy: 100.00%, Final Loss: 0.0007
```



Question 4

Binary Classification using Neural Network

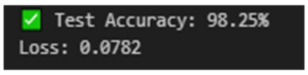
Objective: Build a neural network to classify whether a tumor is malignant or benign using the **Breast Cancer dataset**.

Code:

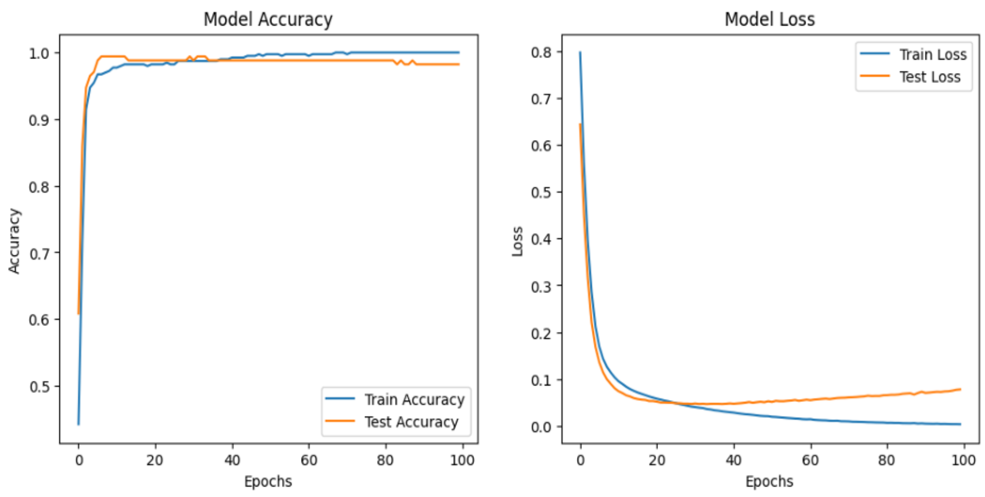
```
import numpy as np

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

data = load_breast_cancer()

X = data.data

y = data.target  # 0 = malignant, 1 = benign

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

model = Sequential([Dense(16, input_dim=X.shape[1],

activation='relu'),   layer 1Dense(8, activation='relu'),

Dense(1, activation='sigmoid')])model.compile(optimizer='adam',

 loss='binary_crossentropy', metrics=['accuracy'])
```

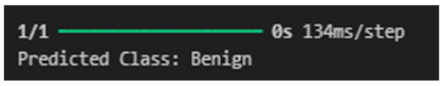
```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)

print(f"✅ Test Accuracy: {accuracy*100:.2f}%")

print(f"Loss: {loss:.4f}")

import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val_accuracy'], label='Test Accuracy')

plt.title("Model Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.title("Model Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend()

plt.show()

sample = X_test[0].reshape(1, -1)

prediction = model.predict(sample)

print(f"Predicted Class: {'Benign' if prediction[0][0] > 0.5 else 'Malignant'}")
```

Output:



```
✅ Test Accuracy: 98.25%
Loss: 0.0782
```





```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
Predicted Class: Benign
```

Question 5

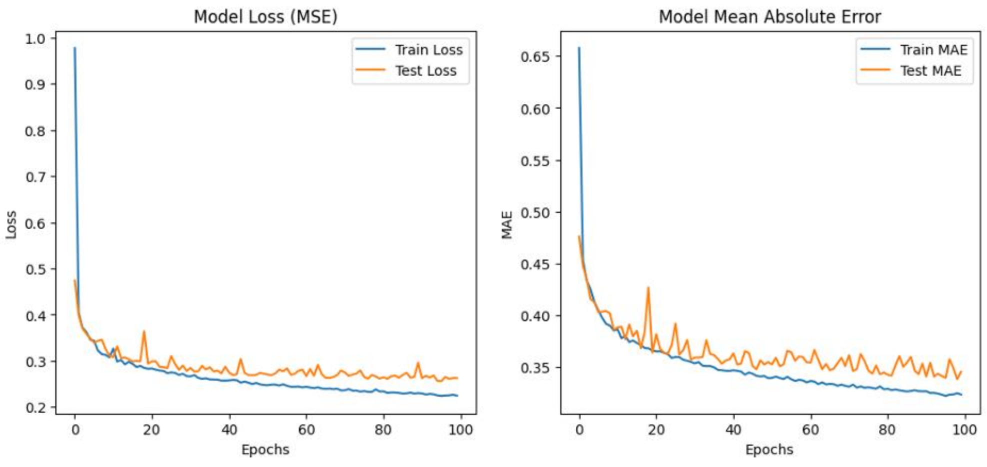
Multi-Class Classification on Iris Dataset

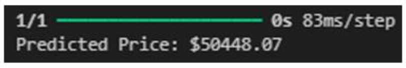
Objective: Train a neural network to classify flower species (Setosa, Versicolor, Virginica).

Code:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.utils import to_categorical

iris = load_iris()

X = iris.data

y = iris.target
```

```
y_encoded = to_categorical(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3,
random_state=42)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

model = Sequential([Dense(8, input_dim=4, activation='relu'),   # hidden layer 1

Dense(6, activation='relu'),Dense(3, activation='softmax') neurons = 3 classes)])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

history = model.fit(X_train, y_train, validation_data=(X_test, y_test),epochs=100,
batch_size=8, verbose=0)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val_accuracy'], label='Test Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.ylabel('Loss')

plt.legend()

plt.show()

sample = np.array([[5.1, 3.5, 1.4, 0.2]])  # Example: Setosa

sample_scaled = scaler.transform(sample)

prediction = model.predict(sample_scaled)

predicted_class = np.argmax(prediction)

print(f"Predicted Species: {iris.target_names[predicted_class]}")
```





```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted Price: $50448.07
```

Question 6

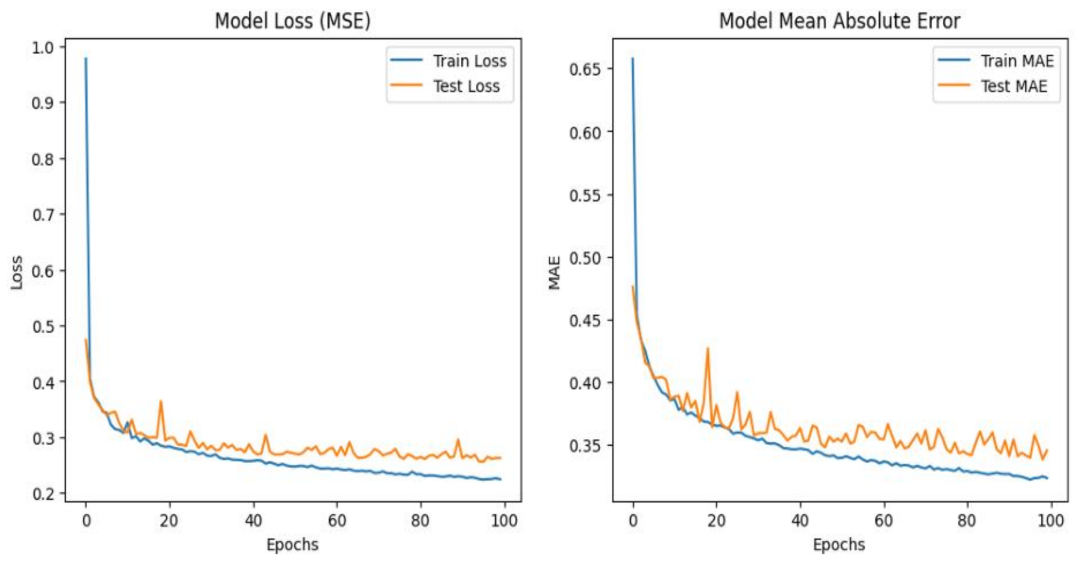
Regression Problem (House Price Prediction)

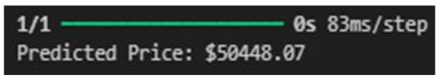
Objective: Predict house prices using the **California Housing dataset**.

Code:

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import fetch_california_housing

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

data = fetch_california_housing()

X = data.data

y = data.target   # Median house value (in 100,000s)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
model = Sequential([Dense(64, input_dim=X.shape[1],activation='relu'),   Dense(32,
activation='relu'), Dense(1)]

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

istory = model.fit(X_train, y_train, validation_data=(X_test, y_test),

epochs=100, batch_size=32, verbose=0)loss, mae = model.evaluate(X_test, y_test,
verbose=0)

print(f"✅ Test MAE (Mean Absolute Error): {mae:.3f}")

print(f"Test MSE: {loss:.3f}")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.title('Model Loss (MSE)')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['mae'], label='Train MAE')

plt.plot(history.history['val_mae'], label='Test MAE')

plt.title('Model Mean Absolute Error')

plt.xlabel('Epochs')

plt.ylabel('MAE')

plt.legend()

plt.show()

sample = X_test[0].reshape(1, -1)

predicted_price = model.predict(sample)[0][0]

print(f"Predicted Price: ${predicted_price * 100000:.2f}")
```





```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
Predicted Price: $50448.07
```

Question 7

Neural Network with Dropout Regularization

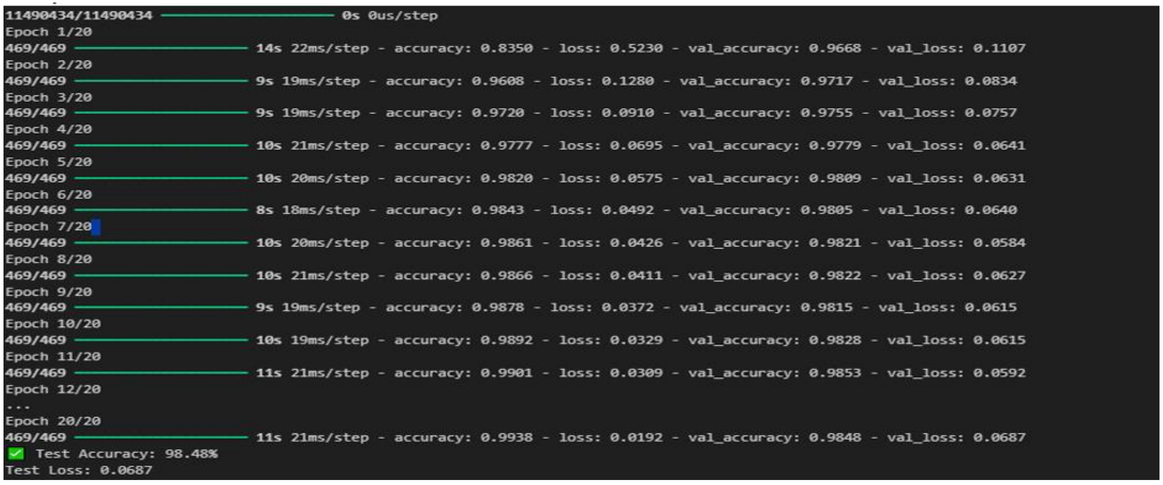
Objective: Prevent overfitting using **Dropout** layers on the **MNIST digit dataset**.

Code:

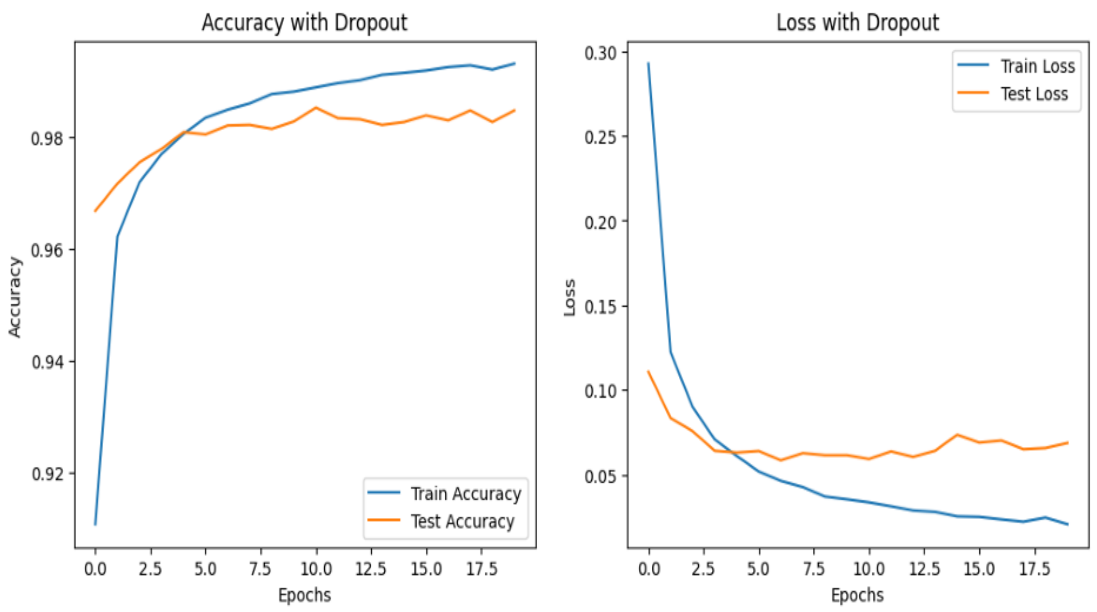
```
import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Flatten

from tensorflow.keras.utils import to_categorical

(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.astype('float32') / 255.0

X_test = X_test.astype('float32') / 255.0
```

```
X_train = X_train.reshape(-1, 28*28)

X_test = X_test.reshape(-1, 28*28)

y_train = to_categorical(y_train, 10)

y_test = to_categorical(y_test, 10)

model = Sequential([Dense(512, input_dim=784, activation='relu'),

Dropout(0.3),  Denes(256, activation='relu'),  Dropout(0.3),
Dense(10,activation='softmax') ])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

loss, accuracy = model.evaluate(X_test, y_test, verbose=0)

print(f"✅ Test Accuracy: {accuracy*100:.2f}%")

print(f"Test Loss: {loss:.4f}")

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)

plt.plot(history.history['val_accuracy'], label='Test Accuracy')

plt.title('Accuracy with Dropout')

plt.legend()

plt.subplot(1,2,2)

plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val_loss'], label='Test Loss')

plt.title('Loss with Dropout')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.show()
```

Output:



```
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 14s 22ms/step - accuracy: 0.8350 - loss: 0.5230 - val_accuracy: 0.9668 - val_loss: 0.1107
Epoch 2/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9608 - loss: 0.1280 - val_accuracy: 0.9717 - val_loss: 0.0834
Epoch 3/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9720 - loss: 0.0910 - val_accuracy: 0.9755 - val_loss: 0.0757
Epoch 4/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9777 - loss: 0.0695 - val_accuracy: 0.9779 - val_loss: 0.0641
Epoch 5/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9820 - loss: 0.0575 - val_accuracy: 0.9809 - val_loss: 0.0631
Epoch 6/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 8s 18ms/step - accuracy: 0.9843 - loss: 0.0492 - val_accuracy: 0.9805 - val_loss: 0.0640
Epoch 7/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 20ms/step - accuracy: 0.9861 - loss: 0.0426 - val_accuracy: 0.9821 - val_loss: 0.0584
Epoch 8/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 21ms/step - accuracy: 0.9866 - loss: 0.0411 - val_accuracy: 0.9822 - val_loss: 0.0627
Epoch 9/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 9s 19ms/step - accuracy: 0.9878 - loss: 0.0372 - val_accuracy: 0.9815 - val_loss: 0.0615
Epoch 10/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 10s 19ms/step - accuracy: 0.9892 - loss: 0.0329 - val_accuracy: 0.9828 - val_loss: 0.0615
Epoch 11/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 21ms/step - accuracy: 0.9901 - loss: 0.0309 - val_accuracy: 0.9853 - val_loss: 0.0592
Epoch 12/20
...
Epoch 20/20
469/469 ━━━━━━━━━━━━━━━━━━━━ 11s 21ms/step - accuracy: 0.9938 - loss: 0.0192 - val_accuracy: 0.9848 - val_loss: 0.0687
✅ Test Accuracy: 98.48%
Test Loss: 0.0687
```



LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines