An autonomous institution approved by RTU, AICTE & UGC • NAAC A+ Accredited

ISI-6, RIICO Institutional Area, Sitapura, Jaipur-302022, Rajasthan
Phone/Fax: 0141-2770790-92, www.pce.poornima.org

DEEP LEARNING AND ITS APPLICATION LAB MANUAL

(Lab Code: 7CAI4-21)

7th Semester, 4th Year

Department of Advance Computing

Session: 2025-26

TABLE OF CONTENT

S.No.	Topic/Name of Experiment	Page Number
	GENERAL DETAILS	
1	Vision &Mission of Institute and Department	iii
2	RTU Syllabus and Marking Scheme	iv
3	Lab Out comes and its Mapping with Pos and PSOs	v-vi
4	Rubrics of Lab	vii-viii
5	Lab Conduction Plan	ix
6	Lab Rotor Plan	x
7	General Lab Instructions	xi
8	Lab Specific Safety Rules	xii
	LIST OF EXPERIMENTS (AS PER RTU SYLLABUS)	
1	Zero Lab	1-3
2	Build a deep neural network model start with linear regression using a) Single variable b) Multiple variables	4-6
3	Write a program to convert : a) Speech into text b) Text into speech c) Video into frames	7-9
4	Build a feed forward neural network for prediction of logic gates.	10-11
5	Write a program for character recognition using: a) CNN b) RNN	12-16
6	Write a program to predict a caption for a sample image using : a) LSTM b) CNN	17-20
7	Write a program to develop : a) Auto encoders using MNIST Handwritten Digits. b) GAN for Generating MNIST Handwritten Digits.	21-25
	Beyond the Syllabus	
8	Write a program to implement L layer neural network.	26-27
9	Write a program to implement Random Forest Algorithm	28-29

INSTITUTE VISION & MISSION

VISION

To create knowledge based society with scientific temper, team spirit and dignity of labor to face the global competitive challenges.

MISSION

To evolve and develop skill-based systems for effective delivery of knowledge so as to equip young professionals with dedication & commitment to excellence in all spheres of life

DEPARTMENT VISION & MISSION

VISION

Become most preferred department for the latest advanced computing programs through creating appropriate teaching-learning and skill up gradation environment that fulfill current industry needs.

MISSION

1. To create experiential learning environment that will enable students to compete globally in advanced computing domain.To contribute significantly to the research and the discovery of new.
2. To adapt latest technological tools and contribute significantly for the advancement of knowledge in computer engineering application in industry, society and environment.
3. To inculcate essential characteristic in the students for their all-round professional development, interaction with industry and society and lifelong learning.
4. To create R & D infrastructure and center of excellence in various advanced computing sub domains.

RTU SYLLABUS AND MARKING SCHEME

7CAI4-21:Deep Learning and its application Lab	
Credit:1	**Max.Marks:100(IA:60,ETE:40)**
0L+0T+2P	**EndTermExam:2Hours**

S.No.	NAME OF EXPERIMENTS
1	Build a deep neural network model start with linear regression using a) Single variable b) Multiple variables
2	Write a program to convert : a) Speech into text b) Text into speech c) Video into frames
3	Build a feed forward neural network for prediction of logic gates.
4	Write a program for character recognition using: a) CNN b) RNN
5	Write a program to predict a caption for a sample image using : a) LSTM b) CNN
6	Write a program to develop : a) Auto encoders using MNIST Handwritten Digits. b) GAN for Generating MNIST Handwritten Digits.

EVALUATION SCHEME

I+II Mid Term Examination			Attendance and performance			End Term Examination			Total Marks
Experiment	Viva	Total	Attendance	Performance	Total	Experiment	Viva	Total	
30	10	40	10	30	40	30	10	40	100

DISTRIBUTION OF MARKS FOR EACH EXPERIMENT

Attendance	Record	Performance	Total
2	3	5	10

LAB OUTCOME AND ITS MAPPING WITH PO& PSO

LAB OUTCOMES

After completion of this course, students will be able to–

8AID4-21.1	To understand how to build the neural network and fundamentals of deep learning.
8AID4-21.2	Identify The Deep Learning Algorithms For Various Types of Learning Tasks in various domains.
8AID4-21.3	Implement Deep Learning Algorithms And Solve Real-world problems.

LO-PO-PSOMAPPINGMATRIXOFCOURSE

LO/PO/PSO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**
8AID4-21.1	3	-	-	-	-	-	-	-	-	-	-	-	2	-	-
8AID4-21.2	-	3	-	-	-	-	-	-	-	-	-	-	2	-	-
8AID4-21.3	-	-	3	-	-	-	-	-	-	-	-	-	2	-	-

PROGRAM OUTCOMES (POs)

PO1	**Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems
PO2	**Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources,and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1	The ability to understand and apply knowledge of mathematics, system analysis & design, Data Modelling, Cloud Technology, and latest tools to develop computer based solutions in the areas of system software, Multimedia, Web Applications, Big data analytics, IOT, Business Intelligence and Networking systems
PSO2	The ability to understand the evolutionary changes in computing, apply standards and ethical practices in project development using latest tools & Technologies to solve societal problems and meet the challenges of the future.
PSO3	The ability to employ modern computing tools and platforms to be an entrepreneur, lifelong learning and higher studies

RUBRICS FOR LAB

Laboratory Evaluation Rubrics:

<table>
<tr><th rowspan="2">S. No.</th><th rowspan="2">Criteria</th><th colspan="3">Sub Criteriaand Marks Distribution</th><th rowspan="2">Outstanding(>90%)</th><th rowspan="2">Admirable(70-90%)</th><th rowspan="2">Average(40-69%)</th><th rowspan="2">Inadequate(<40%)</th></tr>
<tr><th>Mid-Term</th><th>End-Team</th><th>Continues Evaluation</th></tr>
<tr><td rowspan="3">A</td><td rowspan="3">PERFORMANCE(PO1,PO8,PO9)</td><td>Procedure Followed
M.M.100=6</td><td>Procedure Followed
M.M.100=6</td><td>ProcedureF ollowed
M.M.100=2</td><td>• All possible system and Input/Output variables are taken intoaccount
• Performancemeasuresarepro perlydefined
• Experimental scenarios arevery welldefined</td><td>•Most of the system andInput/ Output variables aretaken intoaccount
• Most of the Performancemeasures are properlydefined
• Experimental scenariosaredefined correctly</td><td>• Some of the system and Input/Output variables are taken intoaccount
• Some of the Performancemeasuresareproperly defined
• Experimental scenarios aredefinedbutnot sufficient</td><td>•System and Input/ Outputvariablesarenotdefined
• Performance measures are notproperly defined • Experimentalscenariosnotdefined</td></tr>
<tr><td>Individual/Team Work
M.M.100=6</td><td>Individual/Team Work
M.M.100=6</td><td>Individual/Team Work
M.M.100=2</td><td>•Coordinationamongthegroupmem bersinperforming theexperimentwas excellent</td><td>•Coordination among thegroup members inperforming the experimentwasgood</td><td>•Coordinationamongthegroupmem bersinperforming theexperimentwas average</td><td>•Coordinationamongthegroupmem bersinperforming theexperimentwas verypoor</td></tr>
<tr><td>Precision indatacollection
M.M.100=6</td><td>Precision indatacollection
M.M.100=6</td><td>Precision indatacollection
M.M.100=4</td><td>•Data collected is correct insize and from the experimentperformed</td><td>•Datacollectedisappropriate in size and butnotfromproper sources.</td><td>•Data collected is not soappropriate in size and but frompropersources.</td><td>•Datacollectedis neitherappropriate in size and norfrompropersources</td></tr>
<tr><td rowspan="2">B</td><td rowspan="2">LAB RECORD/WRITTENWORK(</td><td>NA</td><td>NA</td><td>Timing ofEvaluation ofExperiment
M.M.100=6</td><td>• On the Same Date ofPerformance</td><td>• On the Next Turn fromPerformance</td><td>• BeforeDead Line</td><td>• On theDead Line</td></tr>
<tr><td>DataAnalysis
M.M.100=6</td><td>DataAnalysis
M.M.100=6</td><td>DataAnalysis
M.M.100=4</td><td>•Data collected is exhaustivelyanalyzed & appropriate featuresareselected</td><td>•Datacollectedisanalyzed& but appropriate featuresarenot selected</td><td>•Data collected is not analyzedproperly. •Features selected arenotappropriate</td><td>•Datacollectedis notanalyzed & the featuresarenot selected</td></tr>
</table>

		Results andDiscussion M.M.100=6	Results andDiscussion M.M.100=6	Results andDiscussion M.M.100=4	•All results are very wellpresentedwithallvariables • Well prepared neatdiagrams/plots/ tables for allperformancemeasured • Discussed critically behaviorof the system with reference toperformancemeasures • Very well discussed pros ncons ofoutcome	•All results presented butnotallvariablesmentioned • Prepared diagrams /plots/tables for all performancemeasured butnotsoneat • Discussed behavior of thesystem with reference toperformance measures butnotcritical • Discussed pros n cons ofoutcomeinbrief	•Partialresultsareincluded • Prepared diagrams /plots/tables partially for theperformancemeasures • Behavior of the system withreference to performancemeasures has been superficiallypresented • Discussed pros n cons ofoutcomebutnotsorelevant	•Results are included but not asperexperimental scenarios • No proper diagrams /plots/tablesareprepared • Behavior of the system withreference to performancemeasureshasnotbeenpresented • Did not discuss pros n cons ofoutcome
C	**VIVA(PO1, PO10)**	Wayofpresentation M.M.100=5	Wayofpresentation M.M.100=5	Wayofpresentation M.M.100=4	•Presentationwasverygood	•Presentationwasgood	•Presentationwassatisfactory	•Presentation waspoor
		ConceptExplanation M.M.100=5	ConceptExplanation M.M.100=5	ConceptExplanation M.M.100=4	•Conceptualexplanationwasexcellent	•Conceptual explanationwasgood	•Conceptualexplanationwassomewhatgood	•Conceptualexplanationwas Poor
D	**ATTENDANCE**	NA	NA	**A**ttendance M.M.100 =10	•Presentmorethan90%oflabsessions	•Present more than 75% oflab sessions	•Presentmorethan60%oflabsessions	•Presentinlessthan60%labsessions

LAB CONDUCTION PLAN

Total number of Experiments –6
Total numbers of turns required - 12

Experiment Number	Scheduled Week
Experiment-1	Week1
Experiment-2a,b	Week2
Experiment-2c	Week3
Experiment-3	Week4
Experiment-4	Week5
I Mid Term	**Week6**
Experiment-5a	Week7
Experiment-5b	Week8
Experiment-6a	Week9
Experiment-6b	Week10
Experiment-7	Week 11
Experiment-8	Week 12
II MidTerm	**Week13**

DISTRIBUTION OF LAB HOURS

S. No.	Activity	Distribution of Lab Hours	
		Time(180minute)	Time(120minute)
1	Attendance	5	5
2	Explanation of Experiment & Logic	30	30
3	Performing the Experiment	60	30
4	File Checking	40	20
5	Viva/Quiz	30	20
6	Solving of Queries	15	15

LAB ROTAR PLAN

ROTOR-1

Ex.No.	**NAME OF EXPERIMENTS**
1	Zero Lab
2	Build a deep neural network model start with linear regression using a) Single variable b) Multiple variables
3	Write a program to convert : a) Speech into text b) Text into speech c) Video into frames
4	Build a feed forward neural network for prediction of logic gates.
5	Write a program for character recognition using CNN, RNN

ROTOR-2

Ex.No.	**NAME OF EXPERIMENTS**
7	Write a program to predict a caption for a sample image using : a) LSTM b) CNN
8	Write a program to develop : a) Auto encoders using MNIST Handwritten Digits. b) GAN for Generating MNIST Handwritten Digits.
9	Build a Soft Decision Tree with a Deep Neural Network
10	Write a program to implement Random Forest Algorithm

GENERAL LAB INSTRUCTIONS

DO'S

1. Enter the lab on time and leave at propertime.
2. Wait for the previous class to leave before the next class enters.
3. Keep the bag outside in the respective racks.
4. Utilize lab hours in the corresponding.
5. Turn off the machine before leaving the lab unless a member of lab staff has specifically told you not to do so.
6. Leave the labs at least as nice as you found them.
7. If you notice a problem with a piece of equipment (e.g., a computer doesn't respond) or the room in general (e.g., cooling, heating, lighting) please report it to lab staff immediately. Do not attempt to fix the problem yourself.

DON'TS

1. Don't abuse the equipment.
2. Do not adjust the heat or air conditioners. If you feel the temperature is not properly set, inform lab staff; we will attempt to maintain a balance that is healthy for people and machines.
3. Do not attempt to reboot a computer. Report problems to lab staff.
4. Do not remove or modify any software or file without permission.
5. Do not remove printers and machines from the network without being explicitly told to do so by lab staff.
6. Don't monopolize equipment. If you're going to be away from your machine for more than 10 or 15 minutes, logout before leaving. This is both for the security of your account, and to ensure that others are able to use the lab resources while you are not.
7. Don't use internet, internet chat of any kind in your regular lab schedule.
8. Do not download or upload of MP3, JPG or MPEG files.
9. No games are allowed in the lab sessions.
10. No hardware including USB drives can be connected or disconnected in the labs without prior permission of the lab in-charge.
11. No food or drink is allowed in the lab or near any of the equipment. Aside from the fact that it leaves a mess and attracts pests, spilling any thing on a keyboard or other piece of computer equipment could cause permanent, irreparable, and costly damage. (and in fact *has*) If you need to eat or drink, take a break and do so in the canteen.
12. Don't bring any external material in the lab, except your lab record, copy and books.
13. Don't bring the mobile phones in the lab. If necessary, then keep them in silencemode.

14. Please be consider ate of those around you, especially interims of noise level. While labs are natural place for conversations of all types, kindly keep the volume turned down.
15. If you are having problems or questions, please go to either the faculty, lab in-charge or the lab supporting staff. They will help you. We need your full support and cooperation for smooth functioning of the lab.

LAB SPECIFIC SAFETY RULES

Before entering in the lab

1. All the students are supposed to prepare the theory regarding the next experiment/Program.
2. Students are supposed to bring their lab records as per their lab schedule.
3. Previous experiment/program should be written in the lab record.
4. If applicable trace paper/graph paper must be pasted in lab record with proper labeling.
5. All the students must follow the instructions, failing which he/she may not be allowed in the lab.

While working in the lab

1. Adhere to experimental schedule as instructed by the lab in-charge/faculty.
2. Get the previously performed experiment/ program signed by the faculty/ lab incharge.
3. Get the output of current experiment/program checked by the faculty/lab incharge in the lab copy.
4. Each student should work on his/her assigned computer at each turn of the lab.
5. Take responsibility of valuable accessories.

Zero Lab

Introduction about Lab

Software Required

Anaconda Navigator or Google Colab

Package required to run the program

Math, Scipy,Numpy, Matplotlib, Pandas, Sklearn, Tensorflow, Keras etc.

What is Deep Learning?

Deep learning is a type of machine learning that uses artificial neural networks to learn from data. Artificial neural networks are inspired by the human brain, and they can be used to solve a wide variety of problems, including image recognition, natural language processing, and speech recognition.

Deep learning algorithms

Deep learning algorithms are typically trained on large datasets of labeled data. The algorithms learn to associate features in the data with the correct labels. For example, in an image recognition task, the algorithm might learn to associate certain features in an image (such as the shape of an object or the color of an object) with the correct label (such as "dog" or "cat").

Once a deep learning algorithm has been trained, it can be used to make predictions on new data. For example, a deep learning algorithm that has been trained to recognize images of dogs can be used to identify dogs in new images.

How does deep learning work

Deep learning works by using artificial neural networks to learn from data. Neural networks are made up of layers of interconnected nodes, and each node is responsible for learning a specific feature of the data. Building on our previous example with images – in an image recognition network, the first layer of nodes might learn to identify edges, the second layer might learn to identify shapes, and the third layer might learn to identify objects.

As the network learns, the weights on the connections between the nodes are adjusted so that the network can better classify the data. This process is called training, and it can be done using a variety of techniques, such as supervised learning, unsupervised learning, and reinforcement learning.

Once a neural network has been trained, it can be used to make predictions with new data it’s received.

Deep learning vs. machine learning

Both deep learning and machine learning are branches of artificial intelligence, with machine learning being a broader term encompassing various techniques, including deep learning. Both machine learning and deep learning algorithms can be trained on labeled or unlabeled data, depending on the task and algorithm.

Machine learning and deep learning are both applicable to tasks such as image recognition, speech recognition, and natural language processing. However, deep learning often outperforms traditional machine learning in complex pattern recognition tasks like image classification and object detection due to its ability to learn hierarchical representations of data.

Types of deep learning

There are many different types of deep learning models. Some of the most common types include:

Convolutional neural networks (CNNs)

CNNs are used for image recognition and processing. They are particularly good at identifying objects in images, even when those objects are partially obscured or distorted.

Deep reinforcement learning

Deep reinforcement learning is used for robotics and game playing. It is a type of machine learning that allows an agent to learn how to behave in an environment by interacting with it and receiving rewards or punishments.

Recurrent neural networks (RNNs)

RNNs are used for natural language processing and speech recognition. They are particularly good at understanding the context of a sentence or phrase, and they can be used to generate text or translate languages.

Benefits of using deep learning models

There are a number of benefits to using deep learning models, including:

- Can learn complex relationships between features in data: This makes them more powerful than traditional machine learning methods.

- Large dataset training: This makes them very scalable, and able to learn from a wider range of experiences, making more accurate predictions.

- Data-driven learning: DL models can learn in a data-driven way, requiring less human intervention to train them, increasing efficiency and scalability. These models learn from data that is constantly being generated, such as data from sensors or social media.

Challenges of using deep learning models

Deep learning also has a number of challenges, including:

- **Data requirements:** Deep learning models require large amounts of data to learn from, making it difficult to apply deep learning to problems where there is not a lot of data available.
- **Overfitting:** DL models may be prone to overfitting. This means that they can learn the noise in the data rather than the underlying relationships.
- **Bias:** These models can potentially be biased, depending on the data that it's based on. This can lead to unfair or inaccurate predictions. It is important to take steps to mitigate bias in deep learning models.

EXPERIMENT-1

OBJECTIVE

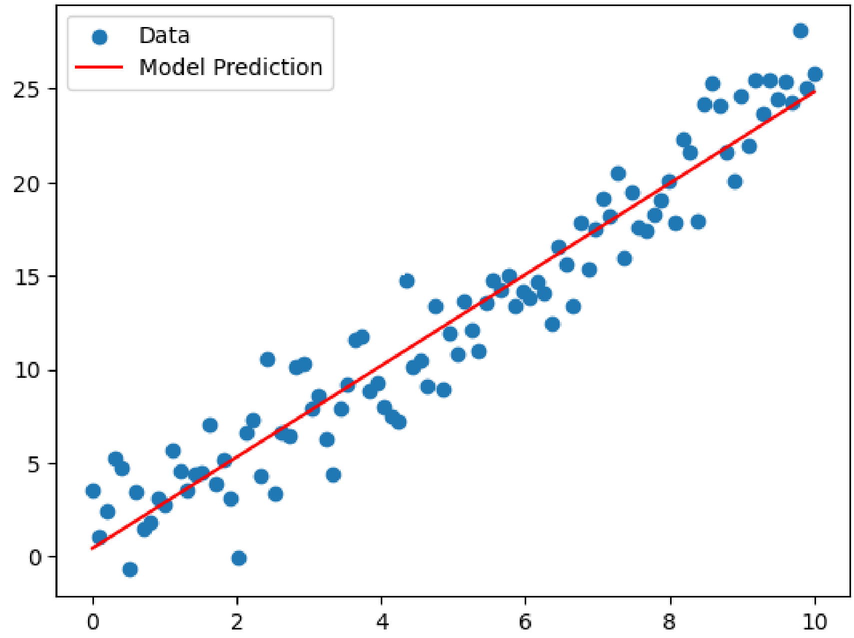
Build a deep neural network model start with linear regression using a) Single variable b) Multiple variables

PROGRAM

a)

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(0)
x = np.linspace(0, 10, 100)
y = 2.5 * x + np.random.normal(0, 2, 100)  # y = 2.5x + noise

# Normalize data
x = x.reshape(-1, 1)  # Reshape for model compatibility

# Build the model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(1,)),  # Input layer
    tf.keras.layers.Dense(1)  # Linear regression layer
])

# Compile the model
model.compile(optimizer='sgd', loss='mse', metrics=['mae'])

# Train the model
history = model.fit(x, y, epochs=100, verbose=0)

# Predict and visualize
y_pred = model.predict(x)

plt.scatter(x, y, label='Data')
plt.plot(x, y_pred, color='red', label='Model Prediction')
plt.legend()
plt.show()
```

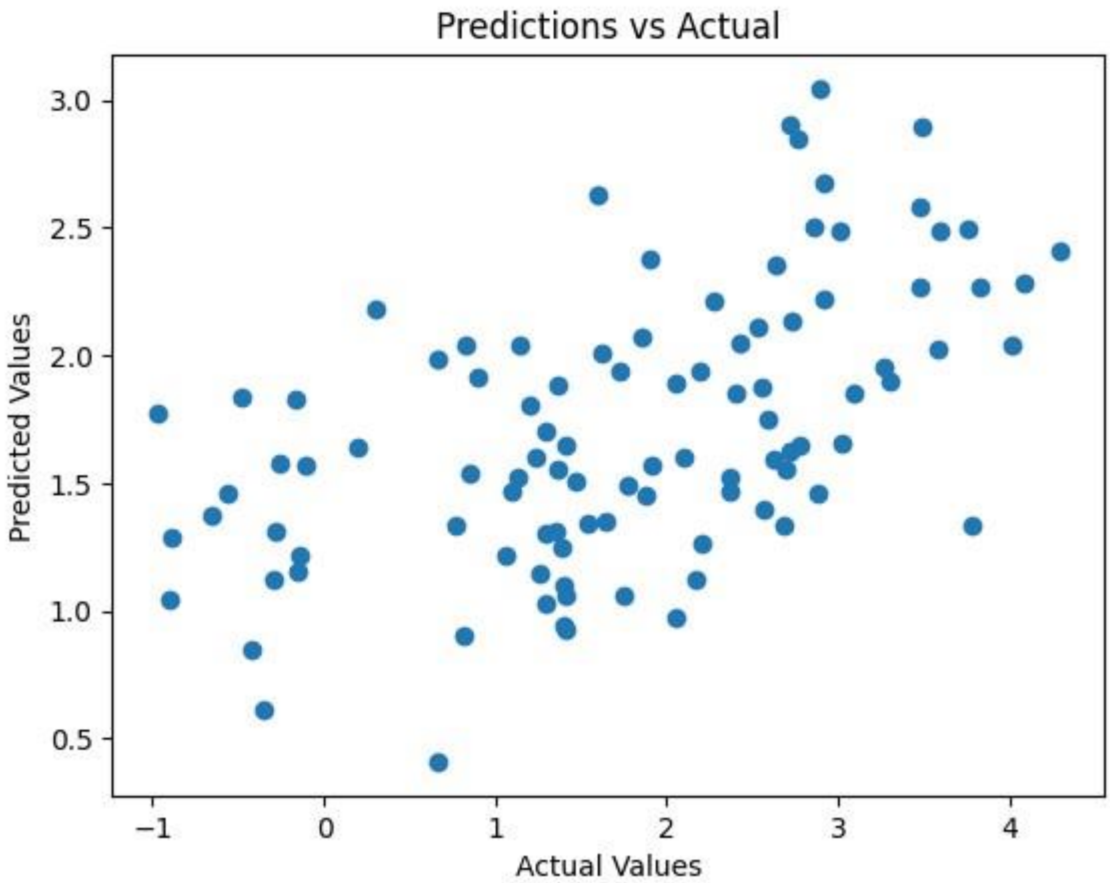
Output:



b) # Generate synthetic data
np.random.seed(0)
X = np.random.rand(100, 3) # 3 features
weights = [3.0, -1.5, 2.0] # True weights
y = np.dot(X, weights) + np.random.normal(0, 0.5, 100) # y = Xw + noise

```
# Build the model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(3,)),  # Input layer for 3 features
    tf.keras.layers.Dense(1)  # Linear regression layer
])

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

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

# Predict and visualize
y_pred = model.predict(X)

# Plot predictions vs actual values
plt.scatter(y, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Predictions vs Actual')
plt.show()
```

Output:



Viva Questions:

1. How do you interpret a linear regression model?
2. What are the basic assumptions of linear regression algorithm?
3. Explain the difference between Correlation and regression.
4. Differentiate between linear regression and logistic regression?
5. List down some of the metrics used to evaluate a Regression Model.

EXPERIMENT-2

OBJECTIVE

Write a program to convert : a) Speech into text b) Text into speech c) Video into frames

Program:

a)
```
import speech_recognition as sr

def speech_to_text():
    recognizer = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        text = recognizer.recognize_google(audio)
        print("You said:", text)
        return text
    except sr.UnknownValueError:
        print("Sorry, could not understand the audio.")
    except sr.RequestError:
        print("Could not request results. Check your internet connection.")

if __name__ == "__main__":
speech_to_text()
```

b)
```
import pyttsx3

def text_to_speech(text):
    engine = pyttsx3.init()  # Initialize the speech engine

    # Set properties (optional)
    engine.setProperty('rate', 150)  # Speed of speech
    engine.setProperty('volume', 1)  # Volume level (0.0 to 1.0)

    # Speak the text
    engine.say(text)
    engine.runAndWait()  # Wait until speech is finished
```

```
if __name__ == "__main__":
    text = input("Enter text to convert to speech: ")
text_to_speech(text)
```

c) import cv2

```
import os


def video_to_frames(video_path, output_folder):

    # Open the video file

    cap = cv2.VideoCapture(video_path)


    # Check if video opened successfully

    if not cap.isOpened():

        print("Error: Could not open video.")

        return


    # Create output folder if it doesn't exist

    if not os.path.exists(output_folder):

        os.makedirs(output_folder)


    frame_count = 0

    while True:

        ret, frame = cap.read()

        if not ret:

            break  # Exit the loop if no more frames
```

```
        # Save the frame as an image file
        frame_filename = os.path.join(output_folder, f"frame_{frame_count:04d}.jpg")
        cv2.imwrite(frame_filename, frame)

        frame_count += 1

    # Release the video capture object
    cap.release()
    print(f"Extracted {frame_count} frames and saved to {output_folder}")

# Example usage
video_to_frames("input_video.mp4", "output_frames")
```

Viva Questions:

1. What Python module is used for text-to-speech?
2. Which library is used for speech to text in Python?
3. Which python version is best for TTS.
4. What Python library is commonly used for video processing?
5. How does the program open a video file?

EXPERIMENT- 3

OBJECTIVE

Build a feed forward neural network for prediction of logic gates.

PROGRAM

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Define data for logic gates
def get_logic_gate_data(gate_type):
    if gate_type == "AND":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([[0], [0], [0], [1]])
    elif gate_type == "OR":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([[0], [1], [1], [1]])
    elif gate_type == "XOR":
        X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
        y = np.array([[0], [1], [1], [0]])
    else:
        raise ValueError("Unsupported logic gate!")
    return X, y

# Select the logic gate to predict
logic_gate = "XOR"  # Change to "AND" or "OR" for other gates

# Load data
X, y = get_logic_gate_data(logic_gate)

# Build the model
model = Sequential([
    Dense(4, input_dim=2, activation='relu'),  # Hidden layer with 4 neurons
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])

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

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

# Evaluate the model
accuracy = model.evaluate(X, y, verbose=0)
print(f"{logic_gate} Gate Prediction Accuracy: {accuracy[1] * 100:.2f}%")
```

```
# Make predictions
predictions = model.predict(X)
print("\nPredictions:")
for i in range(len(X)):
    print(f"Input: {X[i]}, Predicted: {predictions[i][0]:.2f}, Actual: {y[i][0]}")
```

Output:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step

Predictions:
Input: [0 0], Predicted: 0.46, Actual: 0
Input: [0 1], Predicted: 0.44, Actual: 0
Input: [1 0], Predicted: 0.55, Actual: 0
Input: [1 1], Predicted: 0.54, Actual: 1
```

Viva Questions:

1. How can you use an FNN to predict the output of a logic gate (e.g., AND, OR, XOR)?
2. What are the input and output layers for predicting a logic gate?
3. Explain the role of activation functions in FNNs.
4. What is back propagation, and how is it used in training FNNs?
5. How does the number of layers and neurons in an FNN affect its performance?

EXPERIMENT- 4

OBJECTIVE

Write a program for character recognition using: a) CNN b) RNN

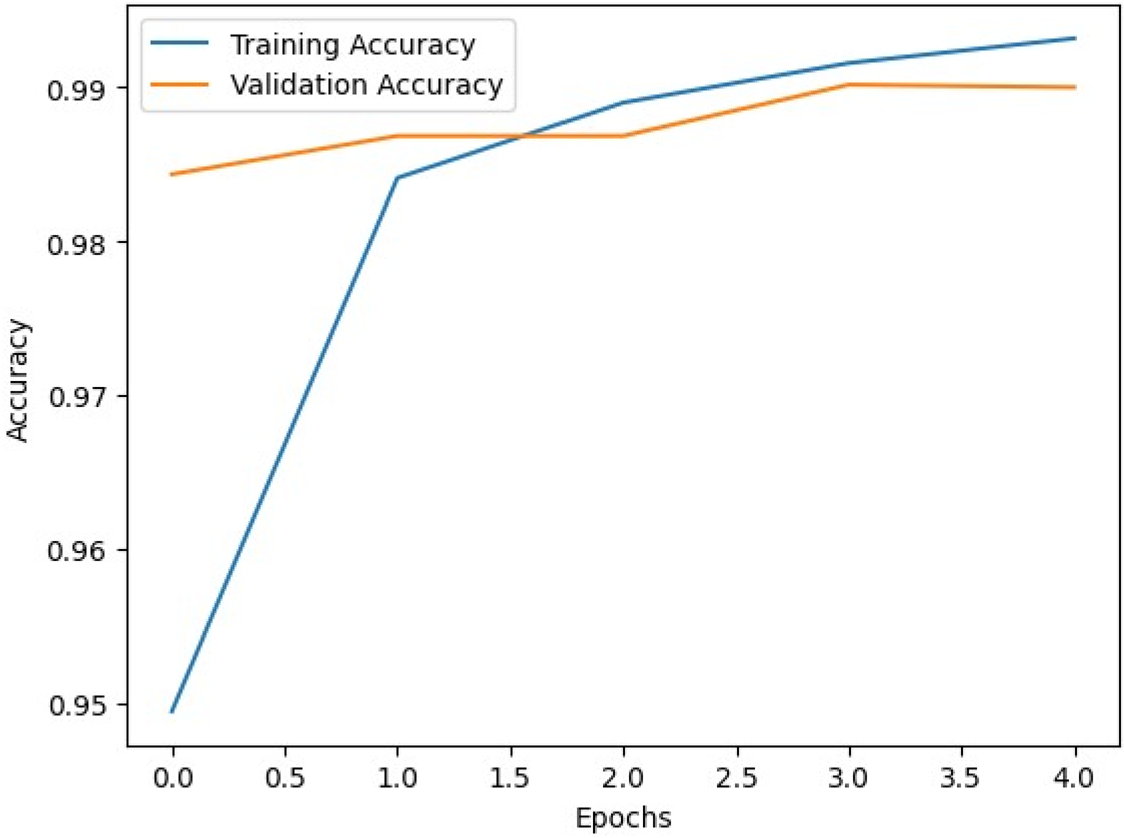
PROGRAM:

a)
```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Load MNIST dataset (handwritten digits)
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.mnist.load_data()

# Preprocess data
X_train = X_train.reshape(-1, 28, 28, 1).astype('float32') / 255.0  # Normalize and reshape
X_test = X_test.reshape(-1, 28, 28, 1).astype('float32') / 255.0    # Normalize and reshape

# One-hot encode the labels
y_train = tf.keras.utils.to_categorical(y_train, 10)
y_test = tf.keras.utils.to_categorical(y_test, 10)

# Build the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),  # First convolutional layer
    layers.MaxPooling2D((2, 2)),  # First pooling layer
    layers.Conv2D(64, (3, 3), activation='relu'),  # Second convolutional layer
    layers.MaxPooling2D((2, 2)),  # Second pooling layer
    layers.Flatten(),  # Flatten the 2D feature maps into 1D
    layers.Dense(128, activation='relu'),  # Fully connected layer
    layers.Dense(10, activation='softmax')  # Output layer for 10 classes (digits 0-9)
])

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

# Train the model
history = model.fit(X_train, y_train, epochs=5, batch_size=64, validation_split=0.1)

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc * 100:.2f}%")

# Visualize training history
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

```
plt.legend()
plt.show()

# Predict and display a few test samples
predictions = model.predict(X_test)
for i in range(5):  # Display first 5 predictions
    plt.imshow(X_test[i].reshape(28, 28), cmap='gray')
    plt.title(f"Predicted: {predictions[i].argmax()}, Actual: {y_test[i].argmax()}")
    plt.axis('off')
    plt.show()
```

Output:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 46s 53ms/step - accuracy: 0.8867 - loss: 0.3856 - val_accuracy: 0.9843 - val_loss: 0.0545
Epoch 2/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 80s 51ms/step - accuracy: 0.9819 - loss: 0.0562 - val_accuracy: 0.9868 - val_loss: 0.0480
Epoch 3/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 82s 51ms/step - accuracy: 0.9892 - loss: 0.0362 - val_accuracy: 0.9868 - val_loss: 0.0441
Epoch 4/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 42s 50ms/step - accuracy: 0.9911 - loss: 0.0272 - val_accuracy: 0.9902 - val_loss: 0.0430
Epoch 5/5
844/844 ━━━━━━━━━━━━━━━━━━━━ 82s 50ms/step - accuracy: 0.9939 - loss: 0.0180 - val_accuracy: 0.9900 - val_loss: 0.0406
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9851 - loss: 0.0407
Test Accuracy: 98.81%
```



b)) import tensorflow as tf

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense, Flatten, Reshape
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
import numpy as np

# Load and preprocess the MNIST dataset
def preprocess_data():
    (x_train, y_train), (x_test, y_test) = mnist.load_data()
    # Normalize the data to range [0, 1]
    x_train = x_train / 255.0
    x_test = x_test / 255.0
    # Convert labels to one-hot encoding
    y_train = to_categorical(y_train, num_classes=10)
    y_test = to_categorical(y_test, num_classes=10)
    return (x_train, y_train), (x_test, y_test)

# Build the RNN model
def build_rnn_model(input_shape, num_classes):
    model = Sequential([
        Reshape((28, 28), input_shape=input_shape),  # Reshape input to (timesteps, features)
        SimpleRNN(128, activation='relu', return_sequences=False),
        Dense(num_classes, activation='softmax')
    ])
    return model
```

```
# Train and evaluate the model
def train_and_evaluate_model(model, x_train, y_train, x_test, y_test, epochs=5, batch_size=64):
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    model.fit(x_train, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.1)
    test_loss, test_accuracy = model.evaluate(x_test, y_test)
    print(f"Test Accuracy: {test_accuracy * 100:.2f}%")

# Main function
if __name__ == "__main__":
    # Load and preprocess data
    (x_train, y_train), (x_test, y_test) = preprocess_data()
    input_shape = x_train.shape[1:]
    num_classes = 10

    # Build the RNN model
    model = build_rnn_model(input_shape, num_classes)
    model.summary()

    # Train and evaluate the model
    train_and_evaluate_model(model, x_train, y_train, x_test, y_test)
```

Output:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/reshape.py:39: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃ Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ reshape (Reshape)                    │ (None, 28, 28)              │ 0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ simple_rnn (SimpleRNN)               │ (None, 128)                 │ 20,096 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 10)                  │ 1,290 │
```

```
└────────────────────────────────────┴────────────────────────────┴──────────
──────┘
 Total params: 21,386 (83.54 KB)
 Trainable params: 21,386 (83.54 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 15s 15ms/step - accuracy: 0.6567
- loss: 0.9712 - val_accuracy: 0.9373 - val_loss: 0.2196
Epoch 2/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 19s 14ms/step - accuracy: 0.9378
- loss: 0.2141 - val_accuracy: 0.9610 - val_loss: 0.1359
Epoch 3/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - accuracy: 0.9554
- loss: 0.1509 - val_accuracy: 0.9627 - val_loss: 0.1283
Epoch 4/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 12s 15ms/step - accuracy: 0.9631
- loss: 0.1303 - val_accuracy: 0.9645 - val_loss: 0.1219
Epoch 5/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 12s 15ms/step - accuracy: 0.9667
- loss: 0.1168 - val_accuracy: 0.9713 - val_loss: 0.0971
Epoch 6/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 13s 15ms/step - accuracy: 0.9676
- loss: 0.1132 - val_accuracy: 0.9738 - val_loss: 0.0921
Epoch 7/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 20s 15ms/step - accuracy: 0.9696
- loss: 0.1086 - val_accuracy: 0.9725 - val_loss: 0.0947
Epoch 8/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 12s 14ms/step - accuracy: 0.9732
- loss: 0.0922 - val_accuracy: 0.9762 - val_loss: 0.0925
Epoch 9/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 21s 15ms/step - accuracy: 0.9737
- loss: 0.0906 - val_accuracy: 0.9748 - val_loss: 0.0846
Epoch 10/10
844/844 ━━━━━━━━━━━━━━━━━━━━ 12s 15ms/step - accuracy: 0.9754
- loss: 0.0844 - val_accuracy: 0.9755 - val_loss: 0.1095
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9697 -
loss: 0.1160
Test Accuracy: 97.56%
```

Viva Questions:

1. What are the key differences between RNNs and CNNs?
2. When would you choose an RNN over a CNN, and vice versa?
3. What is the "vanishing gradient problem" in RNNs, and how can it be addressed?
4. How do RNNs handle variable-length sequences?
5. What are pooling layers, and why are they used in CNNs?

EXPERIMENT-5

OBJECTIVE

Write a program to predict a caption for a sample image using :

a) LSTM
b) CNN

PROGRAM

```
a) import os
import numpy as np
import pickle
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import matplotlib.pyplot as plt
import cv2

def load_cnn_model():
    """Loads a pre-trained CNN model (InceptionV3) for feature extraction."""
    base_model = InceptionV3(weights='imagenet')
    model = Model(inputs=base_model.input, outputs=base_model.layers[-2].output)
    return model

def extract_features(image_path, model):
    """Extract features from an image using the pre-trained CNN model."""
    image = cv2.imread(image_path)
    image = cv2.resize(image, (299, 299))
    image = np.expand_dims(image, axis=0)
    image = image / 255.0  # Normalize
    features = model.predict(image)
    return features

def load_tokenizer(tokenizer_path):
    """Loads the tokenizer used for training."""
    with open(tokenizer_path, 'rb') as handle:
        tokenizer = pickle.load(handle)
    return tokenizer

def generate_caption(model, tokenizer, image_features, max_length=34):
```

```
    """Generates a caption for an image using the trained LSTM model."""
    in_text = 'startseq'
    for _ in range(max_length):
        sequence = tokenizer.texts_to_sequences([in_text])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        yhat = model.predict([image_features, sequence], verbose=0)
        yhat = np.argmax(yhat)
        word = None
        for w, index in tokenizer.word_index.items():
            if index == yhat:
                word = w
                break
        if word is None:
            break
        in_text += ' ' + word
        if word == 'endseq':
            break
    return in_text.replace('startseq', '').replace('endseq', '').strip()

# Paths
image_path = 'sample.jpg'  # Change this to the path of the test image
lstm_model_path = 'caption_model.h5'  # Trained LSTM model
cnn_model = load_cnn_model()
tokenizer_path = 'tokenizer.pkl'  # Tokenizer used for training

# Load tokenizer and model
tokenizer = load_tokenizer(tokenizer_path)
lstm_model = load_model(lstm_model_path)

# Extract image features
image_features = extract_features(image_path, cnn_model)
image_features = np.expand_dims(image_features, axis=0)

# Generate caption
caption = generate_caption(lstm_model, tokenizer, image_features)
print("Predicted Caption:", caption)

# Display image
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.imshow(image)
plt.axis('off')
```

```
plt.title(caption)
plt.show()
```

b)
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications.inception_v3 import InceptionV3, preprocess_input
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import LSTM, Embedding, Dense, Input, Dropout, add
from PIL import Image
import pickle
import matplotlib.pyplot as plt

def load_image(image_path):
    img = Image.open(image_path).resize((299, 299))
    img = np.array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img

def extract_features(image_path, model):
    img = load_image(image_path)
    feature = model.predict(img, verbose=0)
    return feature

def generate_caption(model, tokenizer, image_features, max_length):
    caption = 'startseq'
    for _ in range(max_length):
        sequence = tokenizer.texts_to_sequences([caption])[0]
        sequence = pad_sequences([sequence], maxlen=max_length)
        y_pred = model.predict([image_features, sequence], verbose=0)
        y_pred = np.argmax(y_pred)
        word = tokenizer.index_word.get(y_pred, None)
        if word is None or word == 'endseq':
            break
        caption += ' ' + word
    return caption.replace('startseq', '').replace('endseq', '').strip()

# Load pre-trained CNN (InceptionV3) for feature extraction
base_model = InceptionV3(weights='imagenet')
feature_extractor = Model(inputs=base_model.input, outputs=base_model.layers[-
```

```
2].output)

# Load trained captioning model and tokenizer
image_captioning_model = load_model('image_caption_model.h5')  # Load your trained
LSTM model
tokenizer = pickle.load(open('tokenizer.pkl', 'rb'))  # Load your pre-trained tokenizer
max_length = 34  # Set based on training dataset

# Predict caption for a given image
image_path = 'sample.jpg'  # Provide an image path
image_features = extract_features(image_path, feature_extractor)
caption = generate_caption(image_captioning_model, tokenizer, image_features,
max_length)

# Display the image and generated caption
plt.imshow(Image.open(image_path))
plt.axis('off')
plt.title(caption)
plt.show()
```

Viva Questions:

1. What are filters/kernels, and how do they work in CNNs?
2. Explain the difference between valid and same padding.
3. What are some techniques for preventing overfitting in CNNs?
4. How do LSTMs address the vanishing gradient problem in RNNs?
5. Explain the role of gates (input, forget, output) in LSTM cells.

EXPERIMENT-6

OBJECTIVE

Write a program to develop :

a) Auto encoders using MNIST Handwritten Digits.

b) GAN for Generating MNIST Handwritten Digits.

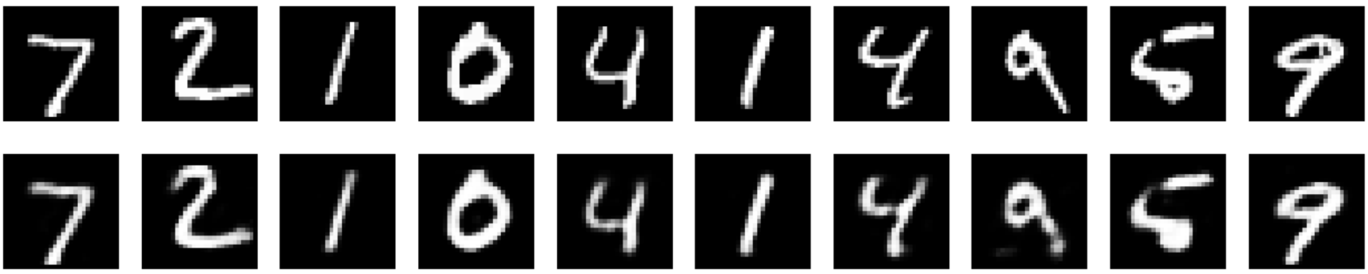
PROGRAM

a)
```
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Flatten, Reshape
from tensorflow.keras.models import Model
import matplotlib.pyplot as plt
import numpy as np

# Load MNIST dataset
(x_train, _), (x_test, _) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
x_train = np.expand_dims(x_train, axis=-1)
x_test = np.expand_dims(x_test, axis=-1)

# Define input shape
input_shape = (28, 28, 1)

# Encoder
input_layer = Input(shape=input_shape)
x = Flatten()(input_layer)
x = Dense(128, activation='relu')(x)
encoded = Dense(64, activation='relu')(x)

# Decoder
x = Dense(128, activation='relu')(encoded)
x = Dense(28 * 28, activation='sigmoid')(x)
decoded = Reshape((28, 28, 1))(x)

# Autoencoder model
autoencoder = Model(input_layer, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

```
# Train the model
autoencoder.fit(x_train, x_train, epochs=10, batch_size=256, shuffle=True,
validation_data=(x_test, x_test))

# Get reconstructed images
reconstructed_images = autoencoder.predict(x_test)

# Display original and reconstructed images
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
    plt.axis('off')

    # Reconstructed images
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(reconstructed_images[i].reshape(28, 28), cmap='gray')
    plt.axis('off')
plt.show()
```

Output:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
Epoch 1/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 8s 24ms/step - loss: 0.3285 - val_loss: 0.1422
Epoch 2/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 20ms/step - loss: 0.1344 - val_loss: 0.1141
Epoch 3/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 15ms/step - loss: 0.1130 - val_loss: 0.1038
Epoch 4/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 0.1032 - val_loss: 0.0972
Epoch 5/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - loss: 0.0973 - val_loss: 0.0934
Epoch 6/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - loss: 0.0936 - val_loss: 0.0906
Epoch 7/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 5s 19ms/step - loss: 0.0911 - val_loss: 0.0886
Epoch 8/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - loss: 0.0891 - val_loss: 0.0868
Epoch 9/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - loss: 0.0875 - val_loss: 0.0852
Epoch 10/10
235/235 ━━━━━━━━━━━━━━━━━━━━ 4s 19ms/step - loss: 0.0863 - val_loss: 0.0842
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
```



b) import tensorflow as tf

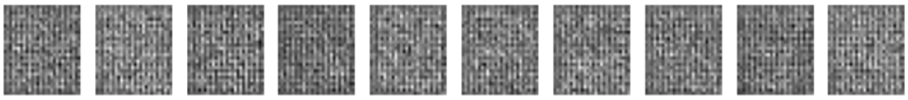
```
from tensorflow.keras import layers, models
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist

# Load and preprocess the MNIST dataset
(x_train, _), (_, _) = mnist.load_data()
x_train = (x_train.astype(np.float32) - 127.5) / 127.5  # Normalize to [-1, 1]
x_train = np.expand_dims(x_train, axis=-1)  # Add channel dimension

# Define GAN parameters
latent_dim = 100
batch_size = 128
epochs = 10000

# Generator Model
def build_generator():
    model = models.Sequential([
        layers.Dense(128 * 7 * 7, activation="relu", input_dim=latent_dim),
        layers.Reshape((7, 7, 128)),
        layers.Conv2DTranspose(128, (3, 3), strides=(2, 2), padding="same",
activation="relu"),
        layers.Conv2DTranspose(64, (3, 3), strides=(2, 2), padding="same",
activation="relu"),
        layers.Conv2DTranspose(1, (3, 3), activation="tanh", padding="same")
    ])
    return model

# Discriminator Model
def build_discriminator():
    model = models.Sequential([
        layers.Conv2D(64, (3, 3), strides=(2, 2), padding="same", input_shape=(28, 28, 1)),
        layers.LeakyReLU(alpha=0.2),
        layers.Dropout(0.3),
        layers.Conv2D(128, (3, 3), strides=(2, 2), padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Dropout(0.3),
        layers.Flatten(),
        layers.Dense(1, activation="sigmoid")
    ])
    return model
```

```
# Build and compile models
generator = build_generator()
discriminator = build_discriminator()
discriminator.compile(optimizer=tf.keras.optimizers.Adam(0.0002, 0.5),
loss="binary_crossentropy", metrics=["accuracy"])

discriminator.trainable = False

gan_input = layers.Input(shape=(latent_dim,))
generated_image = generator(gan_input)
gan_output = discriminator(generated_image)
gan = models.Model(gan_input, gan_output)
gan.compile(optimizer=tf.keras.optimizers.Adam(0.0002, 0.5), loss="binary_crossentropy")

# Training function
def train_gan(epochs, batch_size):
    half_batch = batch_size // 2
    for epoch in range(epochs):
        # Train Discriminator
        idx = np.random.randint(0, x_train.shape[0], half_batch)
        real_images = x_train[idx]
        fake_images = generator.predict(np.random.randn(half_batch, latent_dim))

        real_labels = np.ones((half_batch, 1))
        fake_labels = np.zeros((half_batch, 1))

        d_loss_real = discriminator.train_on_batch(real_images, real_labels)
        d_loss_fake = discriminator.train_on_batch(fake_images, fake_labels)
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

        # Train Generator
        noise = np.random.randn(batch_size, latent_dim)
        misleading_labels = np.ones((batch_size, 1))
        g_loss = gan.train_on_batch(noise, misleading_labels)

        # Print progress
        if epoch % 1000 == 0:
            print(f"Epoch {epoch}, D Loss: {d_loss[0]}, G Loss: {g_loss}")
            plot_generated_images(epoch, generator)

# Function to visualize generated images
```

```
def plot_generated_images(epoch, generator, examples=10):
    noise = np.random.randn(examples, latent_dim)
    generated_images = generator.predict(noise)
    generated_images = (generated_images + 1) / 2  # Rescale to [0,1]

    fig, axes = plt.subplots(1, examples, figsize=(10, 2))
    for i in range(examples):
        axes[i].imshow(generated_images[i, :, :, 0], cmap='gray')
        axes[i].axis('off')
    plt.show()

# Train the GAN
train_gan(epochs, batch_size)
```

Output:

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step
Epoch 0, D Loss: 0.669086217880249, G Loss: 0.692211925983429
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
```



```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 93ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
```

Viva Questions:

1. What is the difference between an encoder and a decoder?
2. How do autoencoders perform dimensionality reduction?
3. What is the difference between an autoencoder and a Variational Autoencoder (VAE)?
4. What is the difference between autoencoder and Gan?
5. Provide an Example of a Real-world Issue that GAN Was Used to Resolve.

BEYOND THE SYLLABUS

EXPERIMENT-1

OBJECTIVE

Build a Soft Decision Tree with a Deep Neural Network

PROGRAM:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Generate a toy dataset (moons dataset)
X, y = make_moons(n_samples=1000, noise=0.2, random_state=42)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define the soft decision tree model
def build_soft_decision_tree(input_dim, hidden_units=10):
    inputs = keras.Input(shape=(input_dim,))

    # Decision nodes: Dense layers with sigmoid activation
    x = layers.Dense(hidden_units, activation="sigmoid")(inputs)

    # Hidden layers for feature transformations
    x = layers.Dense(hidden_units, activation="relu")(x)
    x = layers.Dense(hidden_units, activation="relu")(x)

    # Output layer (soft classification)
    outputs = layers.Dense(1, activation="sigmoid")(x)
```

```
    model = keras.Model(inputs, outputs)
    return model

# Build and compile the model
model = build_soft_decision_tree(input_dim=X_train.shape[1])
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc:.4f}")
```

OUTPUT:

```
Epoch 1/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 5s 46ms/step - accuracy: 0.3507 - loss: 0.7161 - val_accuracy: 0.4750 - val_loss: 0.6887
Epoch 2/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.5250 - loss: 0.6853 - val_accuracy: 0.8000 - val_loss: 0.6665
Epoch 3/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.7824 - loss: 0.6643 - val_accuracy: 0.8150 - val_loss: 0.6453
Epoch 4/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - accuracy: 0.8047 - loss: 0.6375 - val_accuracy: 0.8400 - val_loss: 0.6147
Epoch 5/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8508 - loss: 0.6025 - val_accuracy: 0.8400 - val_loss: 0.5716
Epoch 6/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8564 - loss: 0.5531 - val_accuracy: 0.8500 - val_loss: 0.5156
Epoch 7/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - accuracy: 0.8912 - loss: 0.4919 - val_accuracy: 0.8750 - val_loss: 0.4530
Epoch 8/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.8754 - loss: 0.4284 - val_accuracy: 0.8750 - val_loss: 0.3909
Epoch 9/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - accuracy: 0.8709 - loss: 0.3702 - val_accuracy: 0.8750 - val_loss: 0.3447
Epoch 10/10
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.8950 - loss: 0.3159 - val_accuracy: 0.8550 - val_loss: 0.3175
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - accuracy: 0.8751 - loss: 0.3068
Test Accuracy: 0.8550
```

Viva Questions:

1. What do you mean by Multilayer perceptron?
2. What is data normalization? Why do we need it?
3. What is the role of Activation function in neural network?
4. How we calculate costfunction?
5. What do you understand by back propagation?

BEYOND THE SYLLABUS

EXPERIMENT-2

OBJECTIVE

Write a program to implement Random Forest Algorithm
.

PROGRAM

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

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

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Output : 100%

Viva Questions:

1. Why is Random Forest Algorithm popular?
2. Can Random Forest Algorithm be used both for Categorical and Continues target variables?
3. Why do we prefer Forest rather than a single tree?
4. What are the limitations of bagging trees?
5. Explain working of Random Forest algorithm?