NEIL GOGTE INSTITUTE OF TECHNOLOGY

A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

DEEP LEARNING TECHNIQUES
LABORATORY MANUAL

For
B.E - VI Semester
Prepared By
…….
Assistant Professor

Name: …K.SRINIVAS……..
Academic Year: 2022-2023……..

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

VISION

To be a leader in producing industry-ready and globally competent engineers to **make** India a world leader in software products and services.

.

MISSION

- To provide a conducive learning environment that inculcates problem solving skills, professional and ethical standards, lifelong learning through multi modal platforms and prepare students to become successful professionals.
- To forge industry-institute partnerships to expose students to the technology trends, work culture and ethics in the industry.
- To provide quality training to the students in the state-of-the-art software technologies and tools.
- To encourage research-based projects/activities in emerging areas of technology.
- To nurture entrepreneurial spirit among the students and faculty.
- To induce the spirit of patriotism among the students that enables them to understand India's challenge and strive to develop effective solutions.

DEPARTMENT VISION & MISSION

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

DEPARTMENT VISION

To be a leading department in teaching, research and placements among the Computer Science and Engineering departments in the region producing globally competent and socially responsible graduates in the most conducive academic environment.

DEPARTMENT MISSION

- To provide the state-of-the-art infrastructure to the faculty and students that facilitates continuous professional development and research in fundamental aspects and emerging computing trends alike.
- To impart the technological know-how that helps students to design technical solutions for social needs and kindle entrepreneurial instincts.
- To forge collaborative research between academia and industry for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To inculcate responsibility through knowledge sharing and designing innovative computing solutions that benefits the society-at-large.
- To inculcate environmental sense with research, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

PROGRAM EDUCATIONAL OBJECTIVES

The following Program Educational Objectives (PEOs) have been adopted to attain the vision and mission of the Institution and the Department of Computer Science and Engineering.

- PEO1: Graduates will build successful career in software related industries or will pursue higher studies in elite institutions in India/ abroad.
- PEO2: Graduates will apply the computer engineering principles learnt to provide solutions for the challenging problems in their profession.
- PEO3: Graduates will adapt the challenging work environment through life-long learning for the continuous professional development.
- PEO4: Graduates will articulate to work in the teams and exhibit high level of professionalism and ethical standards.

LIST POs

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

PO1	**Engineering Knowledge:** Apply the knowledge of mathematics science, engineering fundamentals and an engineering specialization to the solution of complex engineering problem
PO2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science 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.

LIST PSOs

At the end of 4 years course period, Computer Science and Engineering graduates at NGIT will be able to:

PSO1	Shall have the ability to find or create opportunities to design and develop appropriate Computer Science solutions for improving the living standards of the people.
PSO2	Shall have expertise in few of the trending technologies like Python, Machine Learning, Deep Learning, Internet of Things (IOT), Data Science, Full stack development, Social Networks, Cyber Security, Big Data, Mobile Apps, CRM, ERP etc.

NEIL GOGTE INSTITUTE OF TECHNOLOGY

A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (AI & ML)

Course Code			Course Title			Core/ Elective
PC 651 CSM		**DEEP LEARNING TECHNIQUES LAB**				**CORE**
Prerequisite		Contact Hours Per Week			CIE	SEE	Credits
	L	T	D	P			
	-	-	-	2	25	50	1

Course Objectives

1. Understand the concepts of Artificial Neural Networks and Deep Learning concepts.
2. Implement ANN and DL algorithms with Tensorflow and Keras.
3. Gain knowledge on Sequence learning with RNN.
4. Gain knowledge on Image processing and analysis with CNN
5. Get information on advanced concepts of computer vision.

Course Outcomes

After learning the concepts of this course, the student is able to

1. Develop ANN without using Machine Learning/Deep learning librarie
2. Understand the Training ANN model with back propagation
3. Develop model for sequence learning using RNN
4. Develop image classification model using ANN and CNN.
5. Generate a new image with auto-encoder and GAN.

List of Programs

1. Create Tensors and perform basic operations with tensors
2. Create Tensors and apply split & merge operations and statistics operations.
3. Design single unit perceptron for classification of iris dataset without using predefined models
4. Design, train and test the MLP for tabular data and verify various activation functions and optimizers tensor flow.
5. Design and implement to classify 32x32 images using MLP using tensorflow/keras and check the accuracy.
6. Design and implement a simple RNN model with tensorflow / keras and check accuracy.
7. Design and implement LSTM model with tensorflow / keras and check accuracy.
8. Design and implement GRU model with tensorflow / keras and check accuracy.
9. Design and implement a CNN model to classify multi category JPG images with tensorflow / keras and check accuracy. Predict labels for new images.
10. Design and implement a CNN model to classify multi category tiff images with tensorflow / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit like regulizers, dropouts etc.
11. Implement a CNN architectures (LeNet, Alexnet, VGG, etc) model to classify multi category Satellite images with tensorflow / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit.
12. Implement an Auto encoder to de-noise image.
13. Implement a GAN application to convert images.

Text Books:

1. Data Science for Beginners- Comprehensive Guide to Most Important Basics in Data Science, Alex Campbell.
2. Artificial Intelligence Technologies, Applications, and Challenges- Lavanya Sharma, Amity University , Pradeep Kumar Garg, IIT Roorkee, India.
3. Artificial Intelligence Fundamentals and Applications- Cherry Bhargava and Pardeep Kumar Sharma, CRC Press.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Outcomes (CO's):

SUBJECT NAME: DEEP LEARNING TECHNIQUES LAB

CODE: PC 651 CSM **SEMESTER: VI**

CO No.	Course Outcomes
PC253CS.1	Develop ANN without using Machine Learning/Deep learning librarie
PC253CS.2	Understand the Training ANN model with back propagation
PC253CS.3	Develop model for sequence learning using RNN
PC253CS.4	Develop image classification model using ANN and CNN.
PC253CS.5	Generate a new image with auto-encoder and GAN.
PC253CS.6	Develop ANN without using Machine Learning/Deep learning librarie

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(AI & ML)

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the program / experiment details.
3. Student should enter into the laboratory with:
    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments.
    c. Formal dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, mustmaintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems,which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviours with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out. If anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING(AI & ML)

CODE OF CONDUCT FOR THE LABORATORY

- All students must observe the dress code while in the laboratory
- Footwear is NOT allowed
- Foods, drinks and smoking are NOT allowed
- All bags must be left at the indicated place
- The lab timetable must be strictly followed
- Be PUNCTUAL for your laboratory session
- All programs must be completed within the given time
- Noise must be kept to a minimum
- Workspace must be kept clean and tidy at all time
- All students are liable for any damage to system due to their own negligence
- Students are strictly PROHIBITED from taking out any items from the laboratory
- Report immediately to the lab programmer if any damages to equipment

BEFORE LEAVING LAB:

- Arrange all the equipment and chairs properly.
- Turn off / shut down the systems before leaving.
- Please check the laboratory notice board regularly for updates.

Lab In – charge

NEIL GOGTE INSTITUTE OF TECHNOLOGY
A unit of Keshav Memorial Technical Educational Society (KMTES)
(Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)
D.No:10TC-111, Kachivanisingaram (V), Uppal, Hyderabad–500088, Telangana.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(AI & ML)

LIST OF EXPERIMENTS

SI.No.	Name of the Experiment	Date of Experiment	Date of Submission	Faculty Signature
1	Create Tensors and perform basic operations with tensors			
2	Create common statistical operations that can be performed on tensors.			
3	Create tnsors and apply split and merge operations by taking input from user.			
4	Design single unit perception for classification of iris dataset without using predefined models.			
5	Design ,train and test the MLP for tabular data and verify various activation functions and optimizers tensor flow			
6	Design and implement a simple RNN model with tensorflow and check accuracy			
7	Design and implement a simple LSTM model with tensorflow and check			
8	Design and implement a simple GRU model with tensorflow and check accuracy.			
9	.Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.			
10	Write program to create a 3x3 matrix with values ranging from 2 to 10.			
11	Write program to create a null vector of size 10 and update sixth value to 11.			
12	Write a program to add, subtract, multiply, divide arguments element-wise.			
13	Write a program to compute the multiplication of two given matrices.			
14	Write a program to reverse an array (first element becomes last).			

15	Write a program to find the number of elements of an array, length of one array element in bytes and total bytes consumed by the elements.			
16	Write a NumPy program to change the dimension of an array.			
17	Create a numpy array list with Indian rivers L and print the list, L is predefined.			
18	Design and implement a CNN model to classify multi category JPG images with tensorflow / keras and check accuracy.Predict labels for new images.			
19	Design and implement a CNN model to classify multi category tiff images with tensorflow/keras and check the accuracy. Check whether your model is overfit/underfit/perfect fit and apply the techniques to avoid overfit and underfit like regulizers, dropouts etc.			
20	Implement a CNN architectures (LeNet, Alexnet, VGG, etc) model to classify multi category Satellite images with tensorflow / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit.			
21	Implement an Auto encoder to de-noise image.			
22	Implement a GAN application to convert images			

A tensor is a mathematical object that generalizes the concept of a vector and a matrix to higher dimensions. It can have any number of dimensions, and each dimension can have any size.

It is a multi-dimensional array of numbers or symbols that can represent a variety of data types, such as scalars, vectors, matrices, or even higher-order objects such as functions.

In physics, tensors are used to represent physical quantities that have both magnitude and direction, such as velocity, force, and stress. In machine learning and deep learning, tensors are used to represent data, such as images, sound, and text, as well as the weights and biases of neural networks.

Tensors have various properties, such as rank, shape, and dimensionality, and they can be added, multiplied, and manipulated in various ways to perform mathematical operations. Tensor calculus is the branch of mathematics that deals with the manipulation and transformation of tensors.

TensorFlow :

is an open-source machine learning framework that was developed by the Google Brain team. It allows developers to build and train machine learning models using a variety of techniques, such as neural networks, decision trees, and clustering algorithms.

Keras :

is a high-level neural network API that is written in Python and runs on top of TensorFlow. Keras is designed to make it easy to build and experiment with deep neural networks. It provides a simple interface for defining and training neural networks, which allows developers to quickly prototype and test their ideas.

By importing Keras from TensorFlow, developers can use the Keras API to build, train, and evaluate neural networks using the TensorFlow backend. This provides a powerful and flexible way to develop machine learning models, while also taking advantage of the speed and efficiency of TensorFlow's computational graph.

Proportion

Some variables are categorical and identify which category or group an individual belongs to. For example, “relationship status” is a categorical variable, and an individual could be single, married, divorced, and so on.

The actual number of individuals in any given category is called the *frequency* for that category. A *proportion,* or *relative frequency*, represents the percentage of individuals that falls into each category. The proportion of a given category, denoted by p, is the frequency divided by the total sample size.

So to calculate the proportion, you

1. Count up all the individuals in the sample who fall into the specified category.
2. Divide by *n*, the number of individuals in the sample.

Population Mean Formula

The ratio wherein the addition of the values to the number of the value is a population mean – if the possibilitie are equal. A population mean include each element from the set of observations that can be made.

The population mean can be found using the following formula:

$$\mu = \frac{\sum X_i}{N}$$

Where,

$\sum X_i$
= Sum of the values
N = Number of the value

Solved Example

Question: Find the population mean of the following numbers 1, 2, 3, 4, 5.

Solution:

Given,

X_i
= 1, 2, 3, 4, 5
$\sum X_i$
= 1 + 2 + 3 + 4 + 5 = 15
N = 5

Population Mean =

$\frac{\sum X_i}{N}$

μ
=
$\frac{15}{5}$

$\mu = 3$
Population Mean = 3

Statistics Formulas

Statistics is a branch of mathematics which deals with numbers and data analysis. Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data. Statistical theory defines a statistic as a function of a sample where the function itself is independent of the sample's distribution.

In short, Statistics is associated with collecting, classifying, arranging and presenting numerical data. It allows us to interpret various results from it and forecast many possibilities. Statistics deals with facts, observations and information which are in the form of numeric data only. With the help of statistics, we are able to find various measures of central tendencies and the deviation of different values from the center.

Statistics Formula Sheet

The important statistics formulas are listed in the chart below:

Mean	$\bar{x} = \frac{\sum x}{n}$	x = Observations given n = Total number of observations

Median	If n is odd, then
	M = $\left(\frac{n+1}{2}\right)^{th}$ term
	If n is even, then
	M = $\frac{\left(\frac{n}{2}\right)^{th}term + \left(\frac{n}{2}+1\right)^{th}term}{2}$
		n = Total number of observations

Mode	The value which occurs most frequently	
Variance	$\sigma^2 = \frac{\sum(x-\bar{x})^2}{n}$	x = Observations given $\bar{x}$ = Mean n = Total number of observations
Standard Deviation	$S = \sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$	x = Observations given $\bar{x}$ = Mean n = Total number of observations

In statistics, a population is a set of similar items or events which pertains to a question or experiment.

$x_1 = 1 \quad x_2 = 2 \quad x_3 = 3 \quad x_4 = 3 \quad x_5 = 9 \quad x_6 = 10$

Concept	Value	Symbol or formula	Calculation
Count	**6**	N	-
Sum	**28**	$\sum x_i = x_1 + x_2 + \ldots + x_N$	1 + 2 + 3 + 3 + 9 + 10
Min	**1**	-	-
Max	**10**	-	-
Range	**9**	max - min	10 - 1
Median	**3**	-	-
Mode	**3**	-	-
Mean	**4.667**	$\mu = \frac{\sum x_i}{N}$	$\frac{28}{6}$
Variance	**12.222**	$\sigma^2 = \frac{(x_1 - \mu)^2 + \cdots + (x_N - \mu)^2}{N}$	$\frac{73.333}{6}$
Standard deviation	**3.496**	$\sigma = \sqrt{\text{variance}} = \sqrt{\sigma^2}$	$\sqrt{12.222}$

To create a tensor in Python, we can use the NumPy library.

NumPy provides a multidimensional array object called ndarray, which can be used to represent tensors.

Example of how to create a 2x3 tensor:

```
import numpy as np
t = np.array([[1, 2, 3], [4, 5, 6]])
```

This creates a 2x3 tensor with values 1, 2, 3, 4, 5, and 6.

1) we created the tensor, we can perform some basic operations on it.

Here are a few examples:

```
# Element-wise multiplication
t1 = t * 2
# Element-wise addition
t2 = t + 3
# Element-wise exponentiation
t3 = np.power(t, 2)
# Transpose
t4 = np.transpose(t)
```

/*

```
import numpy as np
t = np.array([[1, 2, 3], [4, 5, 6]])

t1 = t * 2
print(" \n",t1)

# Element-wise addition
t2 = t + 3
print(" \n" , t2)

# Element-wise exponentiation
```

```
t3 = np.power(t, 2)

# Transpose
print(" \n" ,t3)
t4 = np.transpose(t)
print(" \n" ,t4)
*/
```

Output:

```
[[ 2  4  6]
 [ 8 10 12]]

 [[4 5 6]
 [7 8 9]]

 [[ 1  4  9]
 [16 25 36]]

 [[1 4]
 [2 5]
 [3 6]]
```

2)some common statistical operations that can be performed on tensors.

Assuming you already have a tensor with numerical values, here are some basic operations you can perform:

1. Mean: You can calculate the mean of a tensor by summing all of its values and then dividing by the total number of values. In Python, this can be done using the **mean** function in the NumPy library.
2. Standard deviation: The standard deviation of a tensor can be calculated using the **std** function in NumPy. This measures the spread of the values in the tensor.
3. Variance: Variance is another measure of spread, and can be calculated using the **var** function in NumPy.

4. Maximum and minimum: You can find the maximum and minimum values in a tensor using the **max** and **min** functions in NumPy.
5. Reshaping: You can reshape a tensor using the **reshape** function in NumPy. This allows you to change the dimensions of the tensor while maintaining the same number of values.
6. Transpose: You can transpose a tensor using the **transpose** function in NumPy. This swaps the rows and columns of the tensor.
7. Dot product: You can perform a dot product between two tensors using the **dot** function in NumPy. This calculates the sum of the products of the corresponding elements in each tensor.

```
import numpy as np

# Get user input for tensor dimensions
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

# Create an empty tensor with the specified dimensions
tensor = np.empty((rows, cols))

# Get user input for tensor values
for i in range(rows):
    for j in range(cols):
        tensor[i][j] = float(input(f"Enter value for [{i}][{j}]: "))

# Print the tensor
print("Tensor:\n", tensor)

# Calculate the mean of the tensor
mean = np.mean(tensor)
print("Mean:", mean)

# Calculate the standard deviation of the tensor
std = np.std(tensor)
print("Standard deviation:", std)

# Calculate the variance of the tensor
```

```
var = np.var(tensor)
print("Variance:", var)

# Find the maximum and minimum values in the tensor
max_val = np.max(tensor)
min_val = np.min(tensor)
print("Maximum value:", max_val)
print("Minimum value:", min_val)

# Reshape the tensor to a 1D array
reshaped_tensor = np.reshape(tensor, (1, -1))
print("Reshaped tensor:\n", reshaped_tensor)

# Transpose the tensor
transposed_tensor = np.transpose(tensor)
print("Transposed tensor:\n", transposed_tensor)

# Perform a dot product between the tensor and its transpose
dot_product = np.dot(tensor, transposed_tensor)
print("Dot product:\n", dot_product)
```

output:

```
Enter number of rows: 2
Enter number of columns: 2
Enter value for [0][0]: 1
Enter value for [0][1]: 2
Enter value for [1][0]: 3
Enter value for [1][1]: 4
Tensor:
 [[1. 2.]
 [3. 4.]]
Mean: 2.5
Standard deviation: 1.118033988749895
Variance: 1.25
Maximum value: 4.0
Minimum value: 1.0
Reshaped tensor:
 [[1. 2. 3. 4.]]
Transposed tensor:
 [[1. 3.]
 [2. 4.]]
Dot product:
 [[ 5. 11.]
 [11. 25.]]
1
```

3)Create tensors and apply split and merge operations by taking input from user

```
import numpy as np

        # Get user input for tensor dimensions
rows = int(input("Enter number of rows: "))
cols = int(input("Enter number of columns: "))

        # Create a tensor with random values
        #tensor = np.random.rand(rows, cols)
for i in range(rows):
    for j in range(cols):
       tensor[i][j] = float ( input ( f "Enter value for [{ i }][{ j }]: "))

# Print the tensor
print("Original tensor:\n", tensor)

# Get user input for split axis
split_axis = int(input("Enter the axis to split along (0 for rows, 1 for
columns): "))

# Split the tensor along the specified axis
tensor_split = np.split(tensor, 2, axis=split_axis)

# Print the split tensors
print("Split tensors:\n", tensor_split)

# Merge the split tensors along the same axis
tensor_merged = np.concatenate(tensor_split, axis=split_axis)

# Print the merged tensor
print("Merged tensor:\n", tensor_merged)
```

output:

```
Enter number of rows: 2
Enter number of columns: 2
Enter value for [0][0]: 1
Enter value for [0][1]: 2
Enter value for [1][0]: 3
Enter value for [1][1]: 4
Original tensor:
 [[1. 2.]
 [3. 4.]]
Enter the axis to split along (0 for rows, 1 for columns): 0
Split tensors:
 [array([[1., 2.]]), array([[3., 4.]])]
Merged tensor:
 [[1. 2.]
 [3. 4.]]
```

4) Design single unit perception for classification of iris dataset without using predefined models.

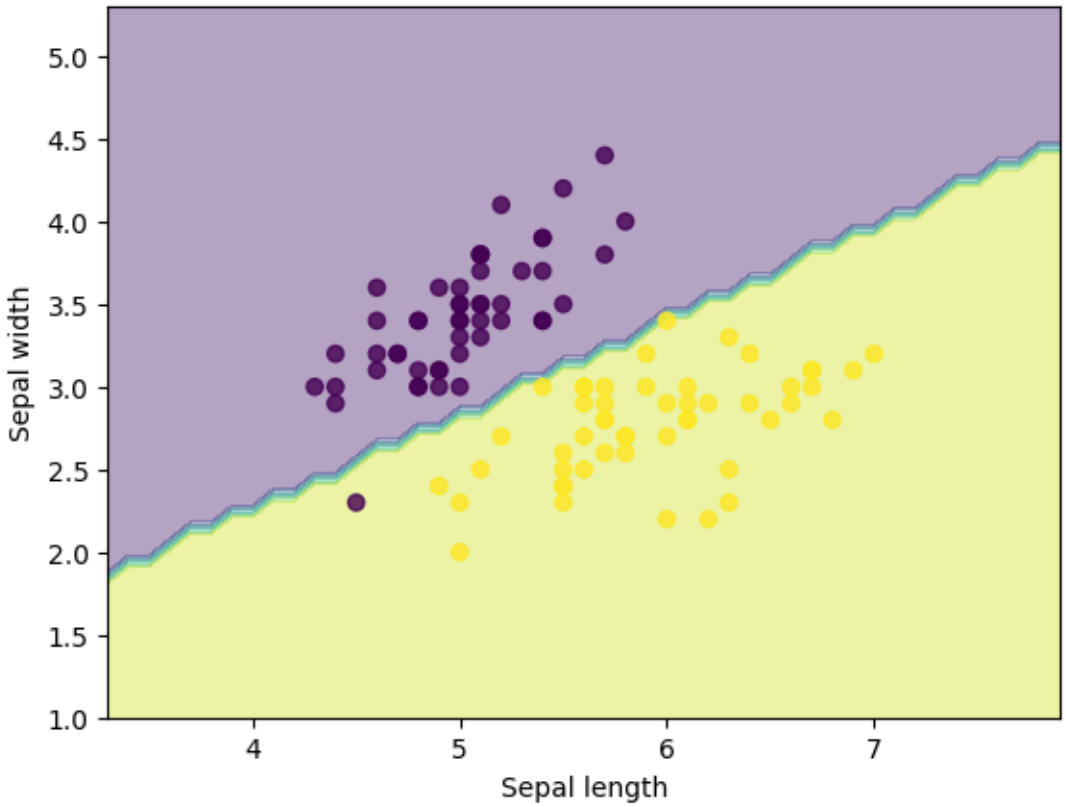
```
Import numpy as np
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()

# Select two classes of flowers for binary classification
X = iris.data[:100, :2]
y = iris.target[:100]

# Define the learning rate and number of iterations
learning_rate = 0.1
num_iterations = 100

# Initialize the weights randomly
weights = np.random.rand(2)

# Define the activation function (here we use a simple threshold function)
def activation(x):
    return np.where(x >= 0, 1, 0)

# Train the perceptron
for i in range(num_iterations):
```

```
    # Initialize the gradient and cost
    gradient = np.zeros(2)
    cost = 0

    # Loop over the training examples
    for j in range(len(X)):
        # Compute the output and error for the current example
        output = activation(np.dot(X[j], weights))
        error = y[j] - output

        # Update the gradient and cost
        gradient += error * X[j]
        cost += error ** 2

    # Update the weights based on the gradient and learning rate
    weights += learning_rate * gradient

    # Print the cost for this iteration
    print("Iteration:", i, "Cost:", cost)

# Plot the decision boundary
import matplotlib.pyplot as plt
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),np.arange(y_min,
y_max, 0.1))
Z = activation(np.dot(np.c_[xx.ravel(), yy.ravel()], weights))
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, alpha=0.8)
plt.xlabel("Sepal length")
plt.ylabel("Sepal width")
plt.show()

Iteration: 0 Cost: 50
Iteration: 1 Cost: 50
Iteration: 2 Cost: 50
Iteration: 3 Cost: 50
Iteration: 4 Cost: 50
Iteration: 5 Cost: 50
```

Iteration: 6 Cost: 50
Iteration: 7 Cost: 50
Iteration: 8 Cost: 50
Iteration: 9 Cost: 50
Iteration: 10 Cost: 50
Iteration: 11 Cost: 50
Iteration: 12 Cost: 50
Iteration: 13 Cost: 50
Iteration: 14 Cost: 50
Iteration: 15 Cost: 50
Iteration: 16 Cost: 50
Iteration: 17 Cost: 50
Iteration: 18 Cost: 50
Iteration: 19 Cost: 47
Iteration: 20 Cost: 50
Iteration: 21 Cost: 47
Iteration: 22 Cost: 50
Iteration: 23 Cost: 44
Iteration: 24 Cost: 50
Iteration: 25 Cost: 45
Iteration: 26 Cost: 50
Iteration: 27 Cost: 46
Iteration: 28 Cost: 50
Iteration: 29 Cost: 45
Iteration: 30 Cost: 50
Iteration: 31 Cost: 45
Iteration: 32 Cost: 50
Iteration: 33 Cost: 45
Iteration: 34 Cost: 50
Iteration: 35 Cost: 45
Iteration: 36 Cost: 50
Iteration: 37 Cost: 45
Iteration: 38 Cost: 50
Iteration: 39 Cost: 45
Iteration: 40 Cost: 50
Iteration: 41 Cost: 46
Iteration: 42 Cost: 50
Iteration: 43 Cost: 45
Iteration: 44 Cost: 48
Iteration: 45 Cost: 44

Iteration: 46 Cost: 46
Iteration: 47 Cost: 42
Iteration: 48 Cost: 46
Iteration: 49 Cost: 42
Iteration: 50 Cost: 46
Iteration: 51 Cost: 40
Iteration: 52 Cost: 39
Iteration: 53 Cost: 21
Iteration: 54 Cost: 2
Iteration: 55 Cost: 1
Iteration: 56 Cost: 1
Iteration: 57 Cost: 1
Iteration: 58 Cost: 1
Iteration: 59 Cost: 1
Iteration: 60 Cost: 1
Iteration: 61 Cost: 2
Iteration: 62 Cost: 2
Iteration: 63 Cost: 1
Iteration: 64 Cost: 2
Iteration: 65 Cost: 2
Iteration: 66 Cost: 1
Iteration: 67 Cost: 2
Iteration: 68 Cost: 2
Iteration: 69 Cost: 2
Iteration: 70 Cost: 1
Iteration: 71 Cost: 2
Iteration: 72 Cost: 2
Iteration: 73 Cost: 2
Iteration: 74 Cost: 1
Iteration: 75 Cost: 2
Iteration: 76 Cost: 2
Iteration: 77 Cost: 2
Iteration: 78 Cost: 1
Iteration: 79 Cost: 2
Iteration: 80 Cost: 2
Iteration: 81 Cost: 1
Iteration: 82 Cost: 2
Iteration: 83 Cost: 2
Iteration: 84 Cost: 2
Iteration: 85 Cost: 1

Iteration: 86 Cost: 2
Iteration: 87 Cost: 2
Iteration: 88 Cost: 2
Iteration: 89 Cost: 1
Iteration: 90 Cost: 2
Iteration: 91 Cost: 2
Iteration: 92 Cost: 2
Iteration: 93 Cost: 1
Iteration: 94 Cost: 2
Iteration: 95 Cost: 2
Iteration: 96 Cost: 1
Iteration: 97 Cost: 2
Iteration: 98 Cost: 2
Iteration: 99 Cost: 2



MLP stands for Multi-Layer Perceptron.

It is a type of artificial neural network that consists of multiple layers of interconnected nodes, or neurons. Each neuron in an MLP takes a set of inputs, performs a weighted sum of those inputs, applies an activation function, and produces an output. The outputs from one layer of neurons are used as inputs to the next layer until the final output is produced.

An MLP is a feedforward neural network, which means that information flows in one direction, from the input layer to the output layer. MLPs are commonly used for classification and regression tasks, such as image recognition, natural language processing, and financial forecasting.

RNN stands for Recurrent Neural Network.

It is another type of artificial neural network that is designed to handle sequential data, such as time series data or natural language text. Unlike MLPs, RNNs have connections that loop back on themselves, allowing the network to maintain a memory of previous inputs. This makes RNNs particularly effective at tasks such as language modeling, machine translation, and speech recognition.

The key feature of RNNs is the use of a hidden state that is updated at each time step, based on the current input and the previous hidden state. The hidden state is then used to produce an output and to update the hidden state at the next time step. This allows the network to capture temporal dependencies in the input data, making it well-suited for tasks that involve predicting future values or generating new sequences of data.

Regenerate response

5). Design ,train and test the MLP for tabular data and verify various activation functions and optimizers tensor flow

Sol)

In this example, we load the breast cancer dataset and split it into training and test sets. We define a function **create_model** that takes an activation function and optimizer as arguments and returns an MLP with two hidden layers and dropout regularization. We then define a list of activation functions and optimizers to try, and loop over them to train and test models with different combinations of activation functions and optimizers. We print the test loss and accuracy for each model.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import SGD, Adam, RMSprop
from tensorflow.keras.activations import relu, sigmoid, tanh

# load the data
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
data = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,
test_size=0.2, random_state=42)

# define the model
def create_model(activation_func, optimizer):
    model = Sequential([
        Dense(64, input_dim=X_train.shape[1], activation=activation_func),
        Dropout(0.5),
        Dense(32, activation=activation_func),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(loss='binary_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    return model

# define the activation functions and optimizers to try
activation_funcs = [relu, sigmoid, tanh]
```

```
optimizers = [SGD(lr=0.01), Adam(lr=0.001), RMSprop(lr=0.001)]

# train and test the models with different activation functions and
optimizers
for activation_func in activation_funcs:
    for optimizer in optimizers:
        model = create_model(activation_func, optimizer)
        print(f'Training model with activation function
{activation_func.__name__} and optimizer
{optimizer.__class__.__name__}...')
        model.fit(X_train, y_train, epochs=50, batch_size=16, verbose=0)
        loss, accuracy = model.evaluate(X_test, y_test)
        print(f'Test loss: {loss:.3f}, Test accuracy: {accuracy:.3f}\n')
```

OUTPUT:
Test loss: 0.202, Test accuracy: 0.939

NOTE: https://www.youtube.com/watch?v=iajq0xQZ2cQ

6.) Design and implement a simple RNN model with tensorflow and check accuracy

Ans)

Implementation of a Recurrent Neural Network (RNN) using TensorFlow. We'll be using the MNIST dataset for training and testing the model. Let's get started!

```
        //First, let's import the necessary libraries:
import tensorflow as tf
from tensorflow.keras.datasets import mnist
        //Next, let's load the MNIST dataset:
(x_train, y_train), (x_test, y_test) = mnist.load_data()
        //Now, we need to normalize the input data and convert the labels to
one-hot encoding:
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
      //Let's define the hyperparameters:
input_shape = (28, 28)
```

```
num_classes = 10
hidden_size = 128
batch_size = 128
epochs = 10

//Now, let's define the RNN model:

model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=input_shape),
    tf.keras.layers.Reshape(target_shape=(input_shape[0],
input_shape[1]*1)),
    tf.keras.layers.LSTM(units=hidden_size, activation='tanh'),
    tf.keras.layers.Dense(num_classes, activation='softmax')
])

/* The model consists of an LSTM layer followed by a dense layer with a
softmax activation function.
Now, let's compile the model:
*/
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(),
              metrics=['accuracy'])


// Finally, let's train the model:
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))
```

//After training, we can evaluate the accuracy of the model on the test data:

```
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

OUTPUT:
Test loss: 0.049647483974695206
Test accuracy: 0.9848999977111816

// Working RNN model for the MNIST dataset in TensorFlow.

7). Design and implement a simple LSTM model with tensorflow and check accuracy

A ns) .

We defined here a simple LSTM model with one layer of 32 units and a dense output layer with a sigmoid activation function. We compile the model using the Adam optimizer and binary cross-entropy loss. We then generate some random data and train the model for 10 epochs using a batch size of 32. Finally, we evaluate the model on the same data and print the test loss and accuracy.

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

# define the model
model = Sequential([
    LSTM(32, input_shape=(10, 1)),
    Dense(1, activation='sigmoid')
])
# compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# generate some random data
import numpy as np
X = np.random.rand(100, 10, 1)
y = np.random.randint(0, 2, (100, 1))

# train the model
model.fit(X, y, epochs=10, batch_size=32)

# evaluate the model
loss, accuracy = model.evaluate(X, y)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

Output:

Test loss: 0.6935951709747314, Test accuracy: 0.5

GRU stands for Gated Recurrent Unit,

Which is a type of recurrent neural network (RNN) that was introduced to address some of the limitations of traditional RNNs, such as the vanishing gradient problem and difficulty in capturing long-term dependencies in sequential data.

GRUs use a gating mechanism to control the flow of information through the network, allowing it to selectively remember or forget information from previous time steps. Specifically, a GRU has two types of gates: an update gate and a reset gate. The update gate determines how much of the previous hidden state should be retained, while the reset gate determines how much of the new input should be combined with the previous hidden state.

The equations that govern the behavior of a GRU are as follows:

- r(t) = σ(Wr[x(t), h(t-1)] + br)
- z(t) = σ(Wz[x(t), h(t-1)] + bz)
- h_hat(t) = tanh(Wh[x(t), r(t) * h(t-1)] + bh)
- h(t) = (1 - z(t)) * h(t-1) + z(t) * h_hat(t)

where:

- x(t) is the input at time step t
- h(t) is the hidden state at time step t
- r(t) is the reset gate at time step t
- z(t) is the update gate at time step t
- h_hat(t) is the proposed activation at time step t
- σ is the sigmoid activation function
-
- is the element-wise multiplication operation
- W and b are the weight matrix and bias vector, respectively, for each gate and the proposed activation

In summary, the GRU model is a type of RNN that uses gating mechanisms to selectively retain or forget information from previous time steps, allowing it to better capture long-term dependencies in sequential data.

8). Design and implement a simple GRU model with tensorflow and check accuracy.

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

# define the model
model = Sequential([
    GRU(32, input_shape=(10, 1)),
    Dense(1, activation='sigmoid')
])
# compile the model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
# generate some random data
import numpy as np
X = np.random.rand(100, 10, 1)
y = np.random.randint(0, 2, (100, 1))

# train the model
model.fit(X, y, epochs=10, batch_size=32)

# evaluate the model
loss, accuracy = model.evaluate(X, y)
print(f'Test loss: {loss}, Test accuracy: {accuracy}')
```

output:

```
Test loss: 0.6837542653083801, Test accuracy: 0.5699999928474426
```

1.Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.

Expected Output:

Original List: [12.23, 13.32, 100, 36.32]

One-dimensional NumPy array: [12.23 13.32 100. 36.32]

Sample Solution:-

Python Code:

```
import numpy as np
l = [12.23, 13.32, 100, 36.32]
print("Original List:",l)
a = np.array(l)
print("One-dimensional NumPy array: ",a)
```

Sample Output:

```
Original List: [12.23, 13.32, 100, 36.32]
One-dimensional NumPy array:  [  12.23   13.32  100.
36.32]
```

2.**Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.**

Expected Output:

[[2 3 4]
[5 6 7]
[8 9 10]]

Sample Solution:-

Python Code:

```
import numpy as np
x =  np.arange(2, 11).reshape(3,3)
print(x)
```

Sample Output:

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

3.Write a NumPy program to create a null vector of size 10 and update sixth value to 11.

Expected Output:

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Update sixth value to 11
[0. 0. 0. 0. 0. 0. 11. 0. 0. 0.]

Sample Solution:-

Python Code:

```
import numpy as np
```

```
x = np.zeros(10)

 print(x)

print("Update sixth value to 11")

x[6] = 11

print(x)
```

Sample Output:

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
Update sixth value to 11
[  0.   0.   0.   0.   0.   0.  11.   0.   0.   0.]
```

4.Write a NumPy program to add, subtract, multiply, divide arguments element-wise.

Sample input: 1.0, 4.0
Expected Output:
Add:
5.0
Subtract:
-3.0
Multiply:
4.0
Divide:
0.25

Sample Solution:-

Python Code:

```
import numpy as np
print("Add:")
print(np.add(1.0, 4.0))
print("Subtract:")
print(np.subtract(1.0, 4.0))
```

```
print("Multiply:")
print(np.multiply(1.0, 4.0))
print("Divide:")
print(np.divide(1.0, 4.0))
```

Sample Output:

```
Add:
5.0
Subtract:
-3.0
Multiply:
4.0
Divide:
0.25
```

5.Write a NumPy program to compute the multiplication of two given matrices.

Sample input Matrix:

[[1, 0], [0, 1]]
[[1, 2], [3, 4]]

Sample Output:

```
original matrix:
[[1, 0], [0, 1]]
[[1, 2], [3, 4]]
Result of the said matrix multiplication:
[[1 2]
 [3 4]]
```

Sample Solution :

```
import numpy as np
p = [[1, 0], [0, 1]]
q = [[1, 2], [3, 4]]
print("original matrix:")
print(p)
print(q)
result1 = np.dot(p, q)
print("Result of the said matrix multiplication:")
print(result1)
```

6 .Write a NumPy program to reverse an array (first element becomes last).

Original array:
[12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37]
Reverse array:
[37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12]

Sample Solution:-

```
import numpy as np
import numpy as np
x = np.arange(12, 38)
print("Original array:")
print(x)
print("Reverse array:")
x = x[::-1]
print(x)
```

Sample Output:
Original array:
[12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36
37]
Reverse array:
[37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14
13
12]

7.Write a NumPy program to convert an array to a float type.
Sample output:

Original array
[1, 2, 3, 4]
Array converted to a float type:
[1. 2. 3. 4.]

Sample Solution:-

Python Code:

```
import numpy as np
import numpy as np
a = [1, 2, 3, 4]
print("Original array")
print(a)
x = np.asfarray(a)
print("Array converted to a float type:")
print(x)
```

Sample Output:

```
Original array
[1, 2, 3, 4]
Array converted to a float type:
[ 1.  2.  3.  4.]
```

8. Write a NumPy program to find the number of elements of an array, length of one array element in bytes and total bytes consumed by the elements.

Expected Output:

Size of the array: 3

Length of one array element in bytes: 8

Total bytes consumed by the elements of the array: 24

Sample Solution:-

NumPy Code:

```
import numpy as np
x = np.array([1,2,3], dtype=np.float64)
print("Size of the array: ", x.size)
```

```
print("Length of one array element in bytes: ",
x.itemsize)
print("Total bytes consumed by the elements of the array:
",
x.nbytes)
```

9. Write a NumPy program to change the dimension of an array.

Expected Output:

6 rows and 0 columns

(6,)

(3, 3) -> 3 rows and 3 columns

[[1 2 3]

[4 5 6]

[7 8 9]]

Change array shape to (3, 3) -> 3 rows and 3 columns

[[1 2 3]

[4 5 6]

[7 8 9]]

Sample Solution:-

NumPy Code:

```
import numpy as np

x = np.array([1, 2, 3, 4, 5, 6])

print("6 rows and 0 columns")

print(x.shape)



y = np.array([[1, 2, 3],[4, 5, 6],[7,8,9]])

print("(3, 3) -> 3 rows and 3 columns ")

print(y)



x = np.array([1,2,3,4,5,6,7,8,9])

print("Change array shape to (3, 3) -> 3 rows and 3
columns ")

x.shape = (3, 3)

print(x)
```

Sample Output:

```
6 rows and 0 columns

(6,)

(3, 3) -> 3 rows and 3 columns

[[1 2 3]
```

[4 5 6]

[7 8 9]]

Change array shape to (3, 3) -> 3 rows and 3 columns

[[1 2 3]

[4 5 6]

[7 8 9]]

10: #Read problem statement carefully and write your python program here

```
L=[3,5,7,10,22,100,90.80,40,33,45,7,5,20,200,17,25,55,39,
50,300,12,54,78,89,9,11,18,23,34,45,67,60,51,65,38,49,19,
24,111,121,23,145,256,34,1,43,88,57,32,27,112,150,145,234
,209,49,79,102]
    2 L is predefined  List available in the current
environment
    3 Create a numpy array  "n" with the elements of L
    4 Display elements of  array  "n"
    5 Display the elements of  n  in revers order.
    6

1 case = 1
    2 output = [3, 5, 7, 10, 22, 100, 90.8, 40, 33, 45,
7, 5, 20, 200, 17, 25, 55, 39, 50, 300, 12, 54, 78, 89,
9, 11, 18, 23, 34, 45, 67, 60, 51, 65, 38, 49, 19, 24,
111, 121, 23, 145, 256, 34, 1, 43, 88, 57, 32, 27, 112,
150, 145, 234, 209, 49, 79, 102]
    3 [102, 79, 49, 209, 234, 145, 150, 112, 27, 32, 57,
88, 43, 1, 34, 256, 145, 23, 121, 111, 24, 19, 49, 38,
65, 51, 60, 67, 45, 34, 23, 18, 11, 9, 89, 78, 54, 12,
300, 50, 39, 55, 25, 17, 200, 20, 5, 7, 45, 33, 40, 90.8,
100, 22, 10, 7, 5, 3]
    4
```

import numpy as np

```
L=[3,5,7,10,22,100,90.80,40,33,45,7,5,20,200,17,25,55,39,50,300,12,54,78,8
9,9,11,18,23,34,45,67,60,51,65,38,49,19,24,111,121,23,145,256,34,1,43,88,5
7,32,27,112,150,145,234,209,49,79,102]
arr=np.array(L)
print(L)
print(L[::-1])
```

11: Create a numpy array list with Indian rivers L and print the list, L is predefined.

```
2 L=[2900,2900,2510,1450,1290,1290,870,760]
3 Display the Length of the rivers
4 Display the revers of a List
5 Display the first two rivers

1 case = 1
2 output =
8
3 [ 760  870 1290 1290 1450 2510 2900 2900]
4 [2900 2900]
```

```
import numpy as np
L2=[2900,2900,2510,1450,1290,1290,870,760]
Larr1=np.array(L2)
print(len(L2))
b=Larr1[::-1]
print(b)
b=Larr1[0:2]
print(b)
```

12:

```
1 Create a numpy array with the dimension 4X4X4 using
arange()
2 Create a View with second row of each element of
0th dimension
```

```
    3 Display the View and display the shape of the array
    4
1 case = 1
    2 output = [[[ 4  5  6  7]]
    3
    4 [[20 21 22 23]]
    5
    6 [[36 37 38 39]]

7
    8 [[52 53 54 55]]]
    9 (4, 4, 4)
```

```
import numpy as np
arr = np.arange(64).reshape(4,4,4)
a=arr[:,1:2,:]
print(a)
print(arr.shape)
```

13 :

```
1 Write a NumPy program to create a vector with values
ranging from 15 to 55
    2 print all values except the first and last.
    3 Insert  a number  55 at 11th position and print
array
    4 Find out the minimum element from the Vector and
print it
    5 Find out the maximum element from the Vector and
print it
1 case = 1
    2 output = [16 17 18 19 20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
    3 40 41 42 43 44 45 46 47 48 49 50 51 52 53]
    4 [15 16 17 18 19 20 21 22 23 24 25 55 26 27 28 29 30
31 32 33 34 35 36 37
```

```
 5 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54]
 6 15
 7 54
```

```
import numpy as np
v = np.arange(15,55)
print(v[1:-1])
print(np.insert(v,11,55))
print(np.min(v))
print(np.max(v))
```

11 :

```
#read the problem statement carefully and write a pytho
program
 2 import numpy as np
 3 m=4 #dont edit this
 4 n=201 #dont edit this

Create a numpy array with the following
 2 1. Elements from 'm' to 'n'-------m&n is already
defined
 3 2. Data type is float m=4 n=201
 4 3. Print Only Even numbers

case = 1
 2 output = [  4.   6.   8.  10.  12.  14.  16.  18.
20.  22.  24.  26.  28.  30.
 3  32.  34.  36.  38.  40.  42.  44.  46.  48.  50.
52.  54.  56.  58.
 4  60.  62.  64.  66.  68.  70.  72.  74.  76.  78.
80.  82.  84.  86.
 5  88.  90.  92.  94.  96.  98. 100. 102. 104. 106.
108. 110. 112. 114.
```

```
    6 116. 118. 120. 122. 124. 126. 128. 130. 132. 134.
136. 138. 140. 142.
    7 144. 146. 148. 150. 152. 154. 156. 158. 160. 162.
164. 166. 168. 170.
    8 172. 174. 176. 178. 180. 182. 184. 186. 188. 190.
192. 194. 196. 198.
    9 200.]
```

```
import numpy as np
m=4
n=201
a=np.arange(m,n,dtype=float)
even=a[a%2==0]
print(even)
```

12 :

```
1 #read the problem statement carefully and write a
python program
```

```
1 Create 1 Dimension array A
    2 2. Reshape A with 4 dimensions 1D = 2 Elements 2D =
3 Elements 3D = 4 Elements 4D = 5 Elements 2. Data type
is int32
    3 3. Display A
1 case = 1
    2 output =  [  5   6   7   8   9]
    3   [ 10  11  12  13  14]
    4   [ 15  16  17  18  19]]
    5
    6  [[ 20  21  22  23  24]
    7   [ 25  26  27  28  29]
    8   [ 30  31  32  33  34]
    9   [ 35  36  37  38  39]]
   10
   11  [[ 40  41  42  43  44]
   12   [ 45  46  47  48  49]
   13   [ 50  51  52  53  54]
   14   [ 55  56  57  58  59]]]
   15
   17 [[[ 60  61  62  63  64]
```

```
18   [ 65  66  67  68  69]
19   [ 70  71  72  73  74]
20   [ 75  76  77  78  79]]
21
22  [[ 80  81  82  83  84]
23   [ 85  86  87  88  89]
24   [ 90  91  92  93  94]
25   [ 95  96  97  98  99]]
26
27  [[100 101 102 103 104]
28   [105 106 107 108 109]
29   [110 111 112 113 114]
30   [115 116 117 118 119]]]]
```

```
import numpy as np
A=np.arange(120,dtype=int)
A=A.reshape(2,3,4,5)
print(A)
```

13 :

```
 1 #read the problem statement carefully and write a
python program

1 A and B are 2 pre-defined lists. Consider the given
lists and create arrays a, b of numpy
    2 Apply the arithmetic operators like +,-,*,/,% on
them Display the output as given in the sample output.
    3
A=[4,3,6,8,2,9,1,45,34,87,22,98,34,62,71,23,67,37,82,45,1
1,23,37,47,98]
    4
B=[5,8,67,43,22,54,33,12,36,73,89,32,12,67,44,87,33,65,22
,89,22,39,22,44,33]
    5 SAMPLE OUTPUT:
    6 Array a = [ 4 3 6 …]
    7 Array b = [ 5 8 67 43 …]
    8 a + b = [ 9 11 73 …]
    9 a - b = [ -1 -5 -61 …]
   10 a * b = [ 20 24 402 …]
   11 a / b = [0.8 0.375 0.08955224 0.18604651 …]
```

```
   12 a % b = [ 4 3 6 …]

1 case = 1
    2 output = Array a = [ 4  3  6  8  2  9  1 45 34 87
22 98 34 62 71 23 67 37 82 45 11 23 37 47
    3 98]
    4 Array b = [ 5  8 67 43 22 54 33 12 36 73 89 32 12
67 44 87 33 65 22 89 22 39 22 44
    5 33]
    6 a + b = [  9  11  73  51  24  63  34  57  70 160
111 130  46 129 115 110 100 102
    7 104 134  33  62  59  91 131]
    8 a - b = [ -1  -5 -61 -35 -20 -45 -32  33  -2  14 -
67  66  22  -5  27 -64  34 -28
    9  60 -44 -11 -16  15   3  65]
   10 a * b = [  20   24  402  344   44  486   33  540
1224 6351 1958 3136  408 4154
   11 3124 2001 2211 2405 1804 4005  242  897  814 2068
3234]
   12 a / b = [0.8        0.375      0.08955224
0.18604651 0.09090909 0.16666667
   13 0.03030303 3.75       0.94444444 1.19178082
0.24719101 3.0625
   14 2.83333333 0.92537313 1.61363636 0.26436782
2.03030303 0.56923077
   15 3.72727273 0.50561798 0.5        0.58974359
1.68181818 1.06818182
   16 2.96969697]
   17 a % b = [ 4  3  6  8  2  9  1  9 34 14 22  2 10 62
27 23  1 37 16 45 11 23 15  3
   18 32]
```

```
import numpy as np
A=[4,3,6,8,2,9,1,45,34,87,22,98,34,62,71,23,67,37,82,45,11,23,37,47,98]
B=[5,8,67,43,22,54,33,12,36,73,89,32,12,67,44,87,33,65,22,89,22,39,22,44,3
3]
a=np.array(A,dtype=int)
b=np.array(B,dtype=int)
print("Array a = {0}".format(a))
print("Array b = {0}".format(b))
c=a+b
```

```
d=a-b
e=a*b
f=a/b
g=a%b
print("a + b = {0}".format(c))
print("a - b = {0}".format(d))
print("a * b = {0}".format(e))
print("a / b = {0}".format(f))
print("a % b = {0}".format(g))
```

14 :

1 #read the problem statement carefully and write a python program

```
1 Create a numpy array convert it into 3X3X3 dimension
    2 Create a View "v1" with the elements present in second row of each element of 0th dimension
    3 Create a View "v2" with the elements present in second col of each element of 0th dimension
    4 Add v1 and v2 and store it in v3
    5 Display v,1v2, v3
    6
```

```
1 case = 1
    2 output = [[[ 3  4  5]]
    3
    4 [[12 13 14]]
    5
    6 [[21 22 23]]]
    7 [[[ 1]
    8  [ 4]
    9  [ 7]]
   10
   11 [[10]
   12  [13]
```

```
 [16]]

[[19]
 [22]
 [25]]]
[[[ 4  5  6]
 [ 7  8  9]
 [10 11 12]]

[[22 23 24]
 [25 26 27]
 [28 29 30]]

[[40 41 42]
 [43 44 45]
 [46 47 48]]]
```

```
import numpy as np
a = np.arange(27).reshape(3,3,3)
b = a[:,1:2,:]
c = a[:,:,1:2]
print(b)
print(c)
print(b+c)
```

14 :

```
#read the problem statement carefully and write a python program

The table below provides the population of daily order volumes for a recent week.
    Calculate the mean, variance, and standard deviation of this population and display them as expected
    Day                Order Volume
    1                     16
    2                     10
    3                     15
    4                     12
    5                      11
    EXPECTED OUTPUT:
    mean = xx
```

```
11 median = yy
12 variance = zz
13 standard deviation = ss
14
```

```
1 case = 1
2 output = mean =  12.8
3 median =  12.0
4 variance =  5.359999999999999
5 standard deviation =  2.315167380558045
```

```
import numpy as np
ordervolume=[16,10,15,12,11]
arr=np.array(ordervolume)
print("mean = ",np.mean(arr))
print("median = ",np.median(arr))
print("variance = ",np.var(arr))
print("standard deviation = ",np.std(arr))
```

15 :

```
1 # read the problem statement carefully and write the python program

A well-known manufacturer of sugarless food products has invested a great deal of time and money in developing
2 the formula for a new kind of sweetener. Although costly to develop, this sweetener is significantly less
3 expensive to produce than the sweeteners the manufacturer had been using. The manufacturer would like to know
4 if the new sweetener is as good as the traditional product. The manufacturer knows that when consumers are asked
5 to indicate their level of satisfaction with the traditional sweeteners, they respond that on average their level
```

```
    6 of satisfaction is 5.5.The manufacturer conducts
market research to determine the level of acceptance of
this
    7 new product. Consumer taste acceptance data are
collected from 25 consumers of sugarless products.
    8 The data collected can be seen in the LIST below
    9
list_sati=[5,6,7,5,6,5,7,4,5,5,6,6,7,5,5,7,5,6,6,7,7,7,6,
5,7]note:all values in list are float values
   10 Display  the following as expected:
   11 Average satisfaction of all consumers
   12 The value in the middle of the satisfaction.
   13 The average of total squared differences of all
elements with mean.
   14 How far the elements from mean?
   15 expected output:
   16 Mean = 5.84
   17 Median = 6.0
   18 Variance = 0.7744
   19 Standard deviation = 0.88

case = 1
    2 output = Mean =  5.88
    3 Median =  6.0
    4 Variance =  0.8256
    5 Standard deviation =  0.9086253353280438
```

```
import numpy as np
list_sati=[5,6,7,5,6,5,7,4,5,5,6,6,7,5,5,7,5,6,6,7,7,7,6,5,7]
list_satisifaction=np.array(list_sati,dtype=float)
print("Mean = ",np.mean(list_satisifaction))
print("Median = ",np.median(list_satisifaction))
print("Variance = ",np.var(list_satisifaction))
print("Standard deviation = ",np.std(list_satisifaction)
```

8. **Write a python code for Design and implement a CNN model to classify multi category JPG images with tensorflow / keras and check accuracy.Predict labels for new images.**

Python code that demonstrates how to design and implement a convolutional neural network (CNN) model using TensorFlow and Keras for multi-category image classification. It also includes the steps to train the model, evaluate its accuracy, and make predictions for new images.

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set the parameters
batch_size = 32
image_height = 128
image_width = 128
num_classes = 3
num_epochs = 10
```

```
# Create the CNN model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(image_height, image_width, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Load and preprocess the image data
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_data_dir = 'path/to/training/directory'

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
# Train the model
model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=num_epochs
)

# Evaluate the model
test_data_dir = 'path/to/test/directory'
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical'
)

accuracy = model.evaluate(test_generator)
print("Test accuracy:", accuracy[1])

# Predict labels for new images
new_image_path = 'path/to/new/image.jpg'
new_image = tf.keras.preprocessing.image.load_img(new_image_path,
target_size=(image_height, image_width))
new_image = tf.keras.preprocessing.image.img_to_array(new_image)
new_image = new_image / 255.0
new_image = tf.expand_dims(new_image, axis=0)

predictions = model.predict(new_image)
predicted_label = tf.argmax(predictions, axis=1)[0]
print("Predicted label:", predicted_label)
```

Make sure to replace the `'path/to/training/directory'`, `'path/to/test/directory'`, and `'path/to/new/image.jpg'` with the actual paths to your training data directory, test data directory, and the new image you want to predict, respectively. Note that this code assumes you have a directory structure where each class/category of images is stored in a separate subdirectory within the training and test directories.Please ensure that you have TensorFlow and Keras installed in your Python environment before running this code.

9. Writ e a python program for Design and implement a CNN model to classify multi category tiff images with tensorflow/keras and check the accuracy. Check whether your model is overfit/underfit/perfect fit and apply the techniques to avoid overfit and underfit like regulizers, dropouts etc.

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Set the parameters
batch_size = 32
image_height = 128
image_width = 128
num_classes = 3
num_epochs = 10

# Create the CNN model
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(image_height, image_width, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu',
kernel_regularizer=regularizers.l2(0.001)))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer='adam',
```

```
        loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
        metrics=['accuracy'])

# Load and preprocess the image data
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_data_dir = 'path/to/training/directory'

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)
validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
# Train the model
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=num_epochs
)
# Evaluate the model
test_data_dir = 'path/to/test/directory'
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical'
)
accuracy = model.evaluate(test_generator)
print("Test accuracy:", accuracy[1])
```

```
# Check for overfitting or underfitting
import matplotlib.pyplot as plt

train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

epochs_range = range(num_epochs)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, train_loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, train_acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.show()
```

Make sure to replace the `'path/to/training/directory'` and `'path/to/test/directory'` with the actual paths to your training data directory and test data directory, respectively.

Note that TIFF images are not natively supported by the Keras ImageDataGenerator. You would need to preprocess your TIFF images and convert them to a compatible format (e.g., JPEG) before using this code.

This code includes a dropout layer with a dropout rate of 0.5 and a dense layer with L2 regularization to help prevent overfitting. The model is trained using the `fit()` function, and the evaluation is performed using the `evaluate()` function.

After training, the code plots the training and validation loss as well as the training and validation accuracy over the epochs to help you analyze whether the model is overfitting, underfitting, or achieving a good fit.

Please ensure that you have TensorFlow and Keras installed in your Python environment before running this code.

10. Implement a CNN architectures (LeNet, Alexnet, VGG, etc) model to classify multi category Satellite images withtensorflow / keras and check the accuracy. Check whether your model is overfit / underfit / perfect fit and apply the techniques to avoid overfit and underfit.

```
import tensorflow as tf
from tensorflow.keras import layers, models, optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping

# Set the parameters
batch_size = 32
image_height = 128
image_width = 128
num_classes = 3
num_epochs = 50

# Create the LeNet model
def build_lenet_model():
    model = models.Sequential()
    model.add(layers.Conv2D(32, (5, 5), activation='relu', input_shape=(image_height, image_width, 3)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (5, 5), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(64, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model

# Create the AlexNet model
def build_alexnet_model():
    model = models.Sequential()
    model.add(layers.Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=(image_height, image_width, 3)))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))
```

```
    model.add(layers.Conv2D(256, (5, 5), activation='relu', padding="same"))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))
    model.add(layers.Conv2D(384, (3, 3), activation='relu', padding="same"))
    model.add(layers.Conv2D(384, (3, 3), activation='relu', padding="same"))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding="same"))
    model.add(layers.MaxPooling2D((3, 3), strides=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model

# Create the VGG model
def build_vgg_model():
    model = models.Sequential()
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same',
input_shape=(image_height, image_width, 3)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dense(num_classes, activation='softmax'))
    return model
```

```
# Select the model architecture
model = build_vgg_model()  # Change the function name to choose a different architecture

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Load and preprocess the image data
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_data_dir = 'path/to/training/directory'

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Apply data augmentation to avoid overfitting
train_datagen_augmented = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

```
train_generator_augmented = train_datagen_augmented.flow_from_directory(
    train_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

# Early stopping to avoid overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=5)

# Train the model
history = model.fit(
    train_generator_augmented,
    validation_data=validation_generator,
    epochs=num_epochs,
    callbacks=[early_stopping]
)

# Evaluate the model
test_data_dir = 'path/to/test/directory'
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='categorical'
)

accuracy = model.evaluate(test_generator)
print("Test accuracy:", accuracy[1])

# Check for overfitting or underfitting
import matplotlib.pyplot as plt

train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history['accuracy']
```

```
val_acc = history.history['val_accuracy']

epochs_range = range(len(train_loss))

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, train_loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, train_acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.show()
```

Make sure to replace the `'path/to/training/directory'` and `'path/to/test/directory'` with the actual paths to your training data directory and test data directory, respectively.In this code, you can choose between three different CNN architectures: LeNet, AlexNet, and VGG. Uncomment the desired model function to select the architecture. The models are compiled with the Adam optimizer and categorical cross-entropy loss.Data augmentation is applied to the training images using the `ImageDataGenerator` class to increase the diversity of the training data and help avoid overfitting. Early stopping is used as a callback to monitor the validation loss and stop the training if the loss does not improve for a specified number of epochs.After training, the code evaluates the model on the test data and prints the test accuracy. Additionally, it plots the training and validation loss as well as the training and validation accuracy over the epochs to help analyze whether the model is overfitting, underfitting, or achieving a good fit.Please ensure that you have TensorFlow and Keras installed in your Python environment before running this code.

11. Implement an Auto encoder to de-noise image.

To implement an autoencoder to denoise images using TensorFlow and Keras.

Autoencoders are a type of neural network that can learn to reconstruct input data by encoding it into a lower-dimensional
representation and then decoding it back to the original shape. Denoising autoencoders are specifically designed to remove
noise from the input data. Let's go through the steps:

Step 1: Import the required libraries

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

Step 2: Define the autoencoder architecture

```
input_img = keras.Input(shape=(28, 28, 1))

# Encoder
x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = layers.MaxPooling2D((2, 2), padding='same')(x)
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = layers.MaxPooling2D((2, 2), padding='same')(x)

# Decoder
x = layers.Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
x = layers.Conv2D(16, (3, 3), activation='relu', padding='same')(x)
x = layers.UpSampling2D((2, 2))(x)
decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = keras.Model(input_img, decoded)
```

In the above code, we define an autoencoder architecture using convolutional and upsampling layers.
The encoder part reduces the dimensionality of the input image, while the decoder part upsamples the encoded
representation to reconstruct the original image.

Step 3: Compile the model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

We use the Adam optimizer and binary cross-entropy loss since we are treating the problem as a pixel-wise binary classification task.

Step 4: Preprocess the data
Load the dataset (example: MNIST)
(x_train, _), (x_test, _) = keras.datasets.mnist.load_data()

# Normalize and reshape the input images
x_train = x_train.astype('float32') / 255.0
x_train = tf.expand_dims(x_train, axis=-1)

x_test = x_test.astype('float32') / 255.0
x_test = tf.expand_dims(x_test, axis=-1)

Step 5: Add noise to the input images
# Add random noise to the training and test images
noise_factor = 0.5
x_train_noisy = x_train + noise_factor *
tf.random.normal(shape=x_train.shape)
x_test_noisy = x_test + noise_factor * tf.random.normal(shape=x_test.shape)

# Clip the values to [0, 1]
x_train_noisy = tf.clip_by_value(x_train_noisy, clip_value_min=0.0,
clip_value_max=1.0)
x_test_noisy = tf.clip_by_value(x_test_noisy, clip_value_min=0.0,
clip_value_max=1.0)

Step 6: Train the autoencoder
autoencoder.fit(x_train_noisy, x_train,
epochs=10,

```
        batch_size=128,
        shuffle=True,
        validation_data=(x_test_noisy, x_test))
```

Adjust the number of epochs and batch size according to your needs.

Step 7: Evaluate the autoencode

```
decoded_imgs = autoencoder.predict(x_test_noisy)

# Calculate the reconstruction loss (MSE)
mse = tf.keras.losses.mean_squared_error(x_test, decoded_imgs)
reconstruction_loss = tf.reduce_mean(mse)

print(f"Reconstruction Loss: {reconstruction_loss:.4f}")
```

Step 8: Visualize the results

```
import matplotlib.pyplot as plt

n = 10  # Number of images to display
plt.figure(figsize=(20, 4))

for i in range(n):
    # Original images
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(tf.squeeze(x_test_noisy[i]))
    plt.title("Original + Noise")
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    # Decoded images
    ax = plt.subplot(2, n, i + n + 1)
    plt.imshow(tf.squeeze(decoded_imgs[i]))
    plt.title("Denoised")
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

plt.show()
```

This code will display a comparison between the original images with added noise and the denoised images generated by the autoencoder.

That's it! You have now implemented an autoencoder to denoise images using TensorFlow and Keras.