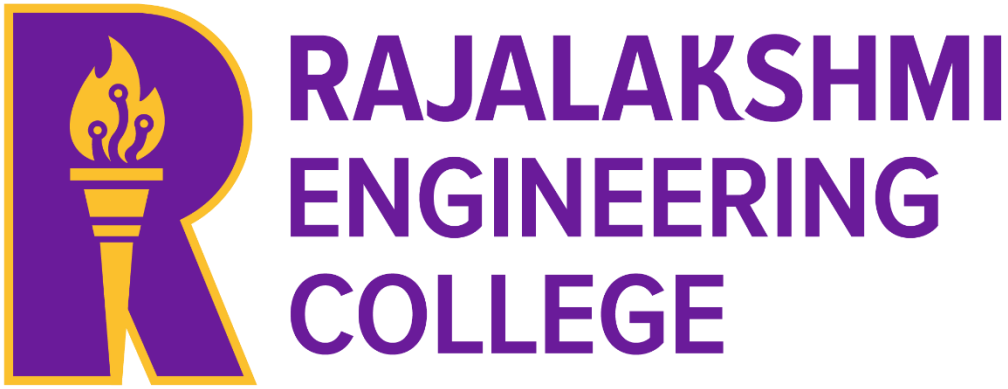
RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105



CS19P18 - DEEP LEARNING CONCEPTS

LABORATORY RECORD NOTEBOOK

NAME: ALDRIN ROGER S

YEAR/SEMESTER: 4TH YEAR / VII

BRANCH: CSE

REGISTER NO: 220701023

COLLEGE ROLL NO: 211622070102

ACADEMIC YEAR: 2025. -2026.

RAJALAKSHMI ENGINEERING COLLEGE

(An Autonomous Institution)

RAJALAKSHMI NAGAR, THANDALAM- 602 105

BONAFIDE CERTIFICATE

NAME: ALDRIN ROGER S **BRANCH/SECTION:** CSE

ACADEMIC YEAR: 2025. -2026. **SEMESTER:** VII

REGISTER NO: 220701023

Certified that this is a Bonafide record of work done by the above student in the **CS19P18 - DEEP LEARNING CONCEPTS**

during the year 2025- 2026

Signature of Faculty In-charge

Submitted for the Practical Examination Held on: 24/11/2025

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	PAGE NO	STAFF SIGN
1	20/08/2025	Create a neural network to recognize handwritten digits using MNIST dataset	14	
2	27/08/2025	Build a Convolutional Neural Network with Keras/TensorFlow	18	
3	03/09/2025	Image Classification on CIFAR-10 Dataset using CNN	23	
4	10/09/202	Transfer learning with CNN and Visualization		
5	17/09/2025	Build a Recurrent Neural Network using Keras/Tensorflow		
6	24/09/2025	Sentiment Classification of Text using RNN		
7	24/09/2025	Build autoencoders with keras/tensorflow		
8	08/10/2025	Object detection with yolo3		
9	08/10/2025	Build GAN with Keras/TensorFlow		
10	08/10/2025	Mini Project		

INSTALLATION AND CONFIGURATION OF TENSORFLOW

Aim:

To install and configure TensorFlow in anaconda environment in Windows 10.

Procedure:

1. Download Anaconda Navigator and install.

2. Open Anaconda prompt

3. Create a new environment dlc with python 3.7 using the following command:
   conda create -n dlc python=3.7

4. Activate newly created environment dlc using the following command:
   conda activate dlc

5. In dlc prompt, install tensorflow using the following command:
   pip install tensorflow

6. Next install Tensorflow-datasets using the following command:
   pip install tensorflow-datasets

7. Install scikit-learn package using the following command:
   pip install scikit-learn

8. Install pandas package using the following command:
   pip install pandas

9. Lastly, install jupyter notebook
   pip install jupyter notebook

10. Open jupyter notebook by typing the following in dlc prompt:
    jupyter notebook

11. Click create new and then choose python 3 (ipykernel)

12. Give the name to the file

13. Type the code and click Run button to execute (eg. Type import tensorflow and then run)

EX NO: 1 CREATE A NEURAL NETWORK TO RECOGNIZE HANDWRITTEN DIGITS USING MNIST DATASET

Aim:

To build a handwritten digit's recognition with MNIST dataset.

Procedure:

1. Download and load the MNIST dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

```
import numpy as np

import tensorflow as tf

from tensorflow import keras

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy_score

# Generate a synthetic dataset

X, y = make_classification(n_samples=1000, n_features=20, random_state=42)

# 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)

# Standardize features (optional but often beneficial)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test

# Define the model
model = keras.Sequential([
    keras.layers.Input(shape=(X_train.shape[1],)),  # Input layer
    keras.layers.Dense(64, activation='relu'),     # Hidden layer with 64 neurons and ReLU activation
    keras.layers.Dense(1, activation='sigmoid')   # Output layer with 1 neuron and sigmoid activation
)
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1)
# Evaluate the model on the test set
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int)
# Calculate accuracy on the test set
accuracy = accuracy_score(y_test, y_pred_classes)
# Calculate test loss
test_loss = model.evaluate(X_test, y_test)
print(f"Test accuracy: {accuracy * 100:.2f}%")
print(f"Test loss: {test_loss[0]:.4f}")
```

Output:

```
Epoch 1/10
23/23 [==============================] - 1s 11ms/step - loss: 0.9133 - accuracy: 0.5083 - val_loss: 0.7724 - val_accuracy: 0.5125
Epoch 2/10
23/23 [==============================] - 0s 4ms/step - loss: 0.6909 - accuracy: 0.5639 - val_loss: 0.6046 - val_accuracy: 0.6375
Epoch 3/10
23/23 [==============================] - 0s 4ms/step - loss: 0.5628 - accuracy: 0.7319 - val_loss: 0.5171 - val_accuracy: 0.7250
Epoch 4/10
23/23 [==============================] - 0s 3ms/step - loss: 0.4905 - accuracy: 0.8139 - val_loss: 0.4598 - val_accuracy: 0.7875
Epoch 5/10
23/23 [==============================] - 0s 5ms/step - loss: 0.4422 - accuracy: 0.8458 - val_loss: 0.4175 - val_accuracy: 0.8250
Epoch 6/10
23/23 [==============================] - 0s 5ms/step - loss: 0.4080 - accuracy: 0.8653 - val_loss: 0.3875 - val_accuracy: 0.8375
Epoch 7/10
23/23 [==============================] - 0s 5ms/step - loss: 0.3816 - accuracy: 0.8708 - val_loss: 0.3664 - val_accuracy: 0.8500
Epoch 8/10
23/23 [==============================] - 0s 5ms/step - loss: 0.3613 - accuracy: 0.8806 - val_loss: 0.3448 - val_accuracy: 0.8500
Epoch 9/10
23/23 [==============================] - 0s 4ms/step - loss: 0.3454 - accuracy: 0.8847 - val_loss: 0.3300 - val_accuracy: 0.8500
Epoch 10/10
23/23 [==============================] - 0s 4ms/step - loss: 0.3328 - accuracy: 0.8847 - val_loss: 0.3200 - val_accuracy: 0.8500
7/7 [==============================] - 0s 2ms/step
7/7 [==============================] - 0s 2ms/step - loss: 0.3999 - accuracy: 0.8250
Test accuracy: 82.50%
Test loss: 0.3999
```

Result:

Thus, the implementation to build a simple neural network using Keras/TensorFlow has been successfully executed.

EX NO:2 BUILD A CONVOLUTIONAL NEURAL NETWORK USING KERAS/TENSORFLOW

Aim:

To implement a Convolutional Neural Network (CNN) using Keras/TensorFlow to recognize and classify handwritten digits from the MNIST dataset with high accuracy.

Procedure:

1. Import required libraries (TensorFlow/Keras, NumPy, etc.).
2. Load the MNIST dataset from Keras.
3. Normalize and reshape the image data.
4. Convert labels to one-hot encoded vectors.
5. Build a CNN model with Conv2D, MaxPooling, Flatten, and Dense layers.
6. Compile the model using categorical crossentropy and Adam optimizer.
7. Train the model on training data.
8. Evaluate the model on test data.
9. Display accuracy and predictions.

Code:

Output:

Result

EX NO: 3 IMAGE CLASSIFICATION ON CIFAR-10 DATASET USING CNN

Aim:

To build a Convolutional Neural Network (CNN) model for classifying images from the CIFAR-10 dataset into one of the ten categories such as airplanes, cars, birds, cats, etc.

Procedure:

1. Download and load the CIFAR-10 dataset using Keras/TensorFlow.
2. Visualize and analyze sample images from the dataset.

3, Preprocess the data:

- Normalize the pixel values (divide by 255)
- Convert class labels to one-hot encoded format

4. Build a CNN model using Keras/TensorFlow:
    - Include convolutional, pooling, flatten, and dense layers.
5. Compile the model with suitable loss function and optimizer.
6. Train the model using training data and validate using test data.
7. Evaluate the model using accuracy and loss on test dataset.
8. Perform predictions on new/unseen CIFAR-10 images.

9 Visualize prediction results with sample images and predicted labels.

Code:

Output:

Result

Ex No: 4 TRANSFER LEARNING WITH CNN AND VISUALIZATION

Aim:

To build a convolutional neural network with transfer learning and perform visualization

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

Output:

Result

EX NO: 5 BUILD A RECURRENT NEURAL NETWORK (RNN) USING KERAS/TENSORFLOW

Aim:

To build a recurrent neural network with Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

Output:

Result

EX NO: 6 SENTIMENT CLASSIFICATION OF TEXT USING RNN

Aim:

To implement a Recurrent Neural Network (RNN) using Keras/TensorFlow for classifying the sentiment of text data (e.g., movie reviews) as positive or negative.

Procedure:

1. Import necessary libraries.
2. Load and preprocess the text dataset (e.g., IMDb).
3. Pad sequences and prepare labels.
4. Build an RNN model with Embedding and SimpleRNN layers.
5. Compile the model with loss and optimizer.
6. Train the model on training data.
7. Evaluate the model on test data.
8. Predict sentiment for new inputs

Code:

Output:

Result

Ex No: 7 BUILD AUTOENCODERS WITH KERAS/TENSORFLOW

Aim:

To build autoencoders with Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

Output:

Result

Ex No: 8 **OBJECT DETECTION WITH YOLO3**

Aim:

To build an object detection model with YOLO3 using Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

Output:

Result

Ex No: 9 BUILD GENERATIVE ADVERSARIAL NEURAL NETWORK

Aim:

To build a generative adversarial neural network using Keras/TensorFlow.

Procedure:

1. Download and load the dataset.
2. Perform analysis and preprocessing of the dataset.
3. Build a simple neural network model using Keras/TensorFlow.
4. Compile and fit the model.
5. Perform prediction with the test dataset.
6. Calculate performance metrics.

Code:

Output:

Result

Ex No: 10 **MINI PROJECT**

Aim:

Code:

Output:

Result

Ex No: 11 **MINI PROJECT**

Result: