MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Vth Semester B. Tech Data Science & Engineering

DSE 3141 Deep Learning Lab [0 0 3 1]

LABORATORY MANUAL

Instructors:

Dr. Rohini R Rao,

Dr. Abhilash K Pai,

Dr. Gangothri

Department of Data Science & Computer Applications

Manipal Institute of Technology, Manipal, India

JULY 2025

COURSE OUTCOMES (COS)

At the end of this course, the student should be able to:		No. of Contact Hours	Marks
CO1	Apply the tools, on different dataset types, do performance evaluation methods, and fine-tuning strategies to build and optimize vanilla deep neural network models for performing classification and regression on structured data.	6	15
CO2	Design, develop, fine-tune, evaluate simple and advanced CNN models for Image classification.	6	35
CO3	Design, develop, fine-tune, evaluate simple and advanced RNN models for sequence modelling tasks like Time series prediction and NLP.	12	26
CO4	Design, develop, fine-tune, and evaluate Deep Learning models for a problem statement of their choice	12	24
	Total	**36**	**100**

ASSESSMENT PLAN

Components	**Continuous Evaluation**	**End semester Examination**
Duration	2.5 Hours per week	180 Minutes
Weightage	60%	40%
Pattern	• 1 evaluation of 20 marks: 1. Record : 6M, 2. Program execution : 7M, 3. Quiz : 3M • 1 Mid-Sem Examination: 20 marks **Mini Project** : 24 marks **Phase1**: 2M **Phase 2**: 3M **Phase 3**: 10M **Phase 4**: 4M **Peer Review**: 5M	Model Performance Analysis: 15 marks, Program execution : 25 marks.

LESSON PLAN

Week No	TOPICS	Course Outcome Addressed
Week 1	Tensorflow & Keras Tutotial, Getting Started with Building Fully Connected Neural Networks In Keras	CO1
Week 2	Experimenting with Deep Neural Networks	CO1
Week 3	Convolutional Neural Networks (CNN) Vs Fully Connected Neural Networks for Image Classification	CO2
Week 4	Advanced CNN Architectures and Transfer Learning for Image Classification	CO2
Week 5	Recurrent Neural Networks for Time Series Forecasting	CO3
Week 6	Mid-Semester Examination, Mini Project	CO2
Week 7	LSTM and GRU for Sentiment Analysis Mini Project Phase 1 evaluation	CO3
Week 8	Neural Machine Translation using Encoder-Decoder Architecture, Mini Project Phase 2 evaluation	CO3
Week 9	Image Reconstruction and Image Denoising Using Autoencoders	CO3
Week 10	Image Generation Using Generative Adversarial Networks, Phase 3 Evaluation	CO4
Week 11	Mini Project Phase 4 Evaluation	CO1
Week 12	End-term lab examination	

References:

SL.No	References
1	Aurelien Geron, "Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow, OReilly Publications
2	Francois Chollet, "Deep Learning with Python", Manning Publications Co, 2nd edition
3	Introduction to Tensorflow, https://www.tensorflow.org/learn
4	Keras Documentation, https://keras.io/
5	Ahmed Menshawy, Md. Rezaul Karim, Giancarlo Zaccone, " Deep Learning with TensorFlow", Packt Publishing

TENSORFLOW & KERAS TUTORIAL

1.1 What is TensorFlow?

TensorFlow is an open-source deep learning framework developed by the Google Brain team. It allows users to create, train, and deploy machine learning models, especially deep neural networks. TensorFlow provides a flexible architecture to work with numerical data using multi-dimensional arrays called **tensors**. It supports both CPU and GPU computations, making it suitable for running on a variety of hardware.

1.2 What are Tensors?

In TensorFlow, tensors are the fundamental data structures used for representing data. They are similar to multi-dimensional arrays and can hold data of any number of dimensions. Tensors are the building blocks of neural networks, as they store the input data, weights, biases, and intermediate outputs during the computation.

Examples of Tensors:

1. Scalar (0-D tensor): A single value is a 0-D tensor.

   Eg: `scalar_tensor = 5  #rank-0 tensor`

2. Vector (1-D tensor): A 1-D tensor contains a sequence of values.

   Eg: `vector_tensor = [1, 2, 3, 4, 5]   #rank-1 tensor`

3. Matrix (2-D tensor): A 2-D tensor is an array of arrays.

   Eg: `matrix_tensor = [[1, 2, 3], [4, 5, 6], [7, 8, 9]] #rank-2 tensor`

4. Higher-dimensional tensor (e.g., 3-D tensor):

   Eg: `tensor_3d = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]] #rank-3 tensor`

Note: For a detailed explanation, visit the TensorFlow | Tensor documentation: https://www.tensorflow.org/guide/tensor

1. 3 Graph Computation:

TensorFlow follows a symbolic approach for computation using graphs. A graph is a computational graph that represents the flow of data through a series of operations (nodes) to produce output (tensors). The nodes in the graph represent operations, and the edges represent tensors flowing between these operations.

Example of Graph Computation:

```
import tensorflow as tf

# Define input variables (placeholders)
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

# Define operations
x_squared = tf.square(x)    # Square operation
x_squared_times_y = tf.multiply(x_squared, y)    # Multiply operation
result = tf.add(x_squared_times_y, tf.add(y, 2))    # Add operation

# Create a session to run the computation graph
with tf.Session() as sess:
    # Provide input values and run the graph
    output = sess.run(result, feed_dict={x: 3.0, y: 4.0})
    print("Output:", output)
```

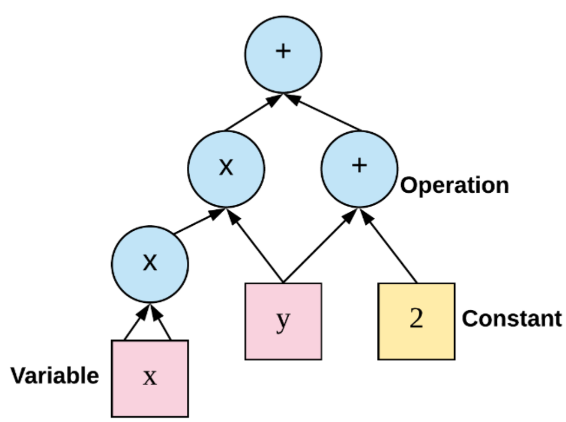


Fig1: Computation graph in tensorflow for **f(x, y) =** x^2**y + y + 2**
[Image Source: https://iq.opengenus.org]

1.4 What is Keras?

Keras is an open-source high-level neural networks API written in Python and capable of running on top of TensorFlow, among other backends. It was designed with a focus on enabling fast experimentation and easy-to-use syntax for building deep learning models. Keras provides a user-friendly interface for constructing complex neural networks, making it an ideal choice for beginners in deep learning.

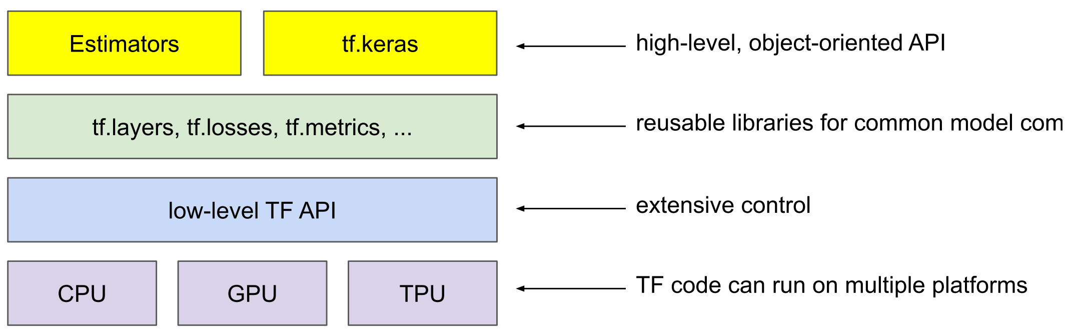


Fig 2. Tensorflow and Keras as API
Image Source: https://developers.google.com/

Note: For a detailed explanation, visit the TensorFlow | Keras documentation: https://www.tensorflow.org/guide/keras

In Keras, there are two primary ways to create deep learning models: the **Sequential API** and the **Functional API**. Each approach serves a different purpose and offers distinct advantages.

1.5 Sequential API:

The Sequential API is the simplest and most straightforward way to build deep learning models in Keras. It allows you to create a linear stack of layers, where each layer has exactly one input tensor and one output tensor. This means that the data flows sequentially through each layer in the order they are added to the model. The Sequential API is well-suited for simple feedforward neural networks and other models that have a clear linear flow of data.

Example of Sequential API:

```python
from keras.models import Sequential
from keras.layers import Dense, Input

# Create a sequential model
model = Sequential()

# Add layers to the model
model.add(Input(shape=(input_dim,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
```

```
model.add(Dense(10, activation='softmax'))

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

# Print the model summary
model.summary()
```

1.6 Functional API:

The Functional API in Keras allows you to create more complex models with multiple input and output tensors, as well as models with shared layers. It provides greater flexibility and is particularly useful when building models with branching or merging architectures.

Example of Functional API:

```
from keras.models import Model
from keras.layers import Input, Dense


# Define input tensor
input_tensor = Input(shape=(input_dim,))

# Create layers and connect them
hidden_layer1 = Dense(64, activation='relu')(input_tensor)
hidden_layer2 = Dense(32, activation='relu')(hidden_layer1)
output_tensor = Dense(10, activation='softmax')(hidden_layer2)

# Create the model
model = Model(inputs=input_tensor, outputs=output_tensor)

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

# Print the model summary
model.summary()
```

1.7 Deep Learning Model Life-Cycle

The deep learning model life cycle typically involves the following steps: Define the model, Compile the model, Fit the model, Evaluate the model, and Make predictions.

I. Define the Model:

In this step, you specify the architecture of your deep learning model. You define the layers, their configurations, activation functions, and any other required settings. The architecture depends on the problem you are trying to solve, and it may include fully connected layers, convolutional layers, recurrent layers, etc.

```
from keras.models import Sequential
from keras.layers import Dense

# Define the model
model = Sequential()
model.add(Dense(64, activation='relu', input_shape=(input_dim,)))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

II. Compile the Model:

After defining the model, you need to compile it. During this step, you specify the loss function, optimizer, and evaluation metrics. The loss function is used to measure how well the model is performing on the training data. The optimizer determines how the model's weights are updated during training, and the evaluation metrics provide additional performance metrics during training.

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

III. Fit the Model:

In this step, you train the model on your training data. You provide the input features (X) and their corresponding target labels (y) to the model. The model then adjusts its internal parameters (weights) through an optimization process (usually gradient descent) to minimize the defined loss function.

```
# Fit the model
model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_data=(X_val, y_val))
```

IV. Evaluate the Model:

After the model is trained, you need to evaluate its performance on a separate set of data that it has never seen before (e.g., a validation set or a test set). This step gives you an indication of how well the model generalizes to unseen data.

```
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test loss: {loss}, Test accuracy: {accuracy}")
```

V. Make Predictions:

Once the model is trained and evaluated, you can use it to make predictions on new, unseen data. You pass the new data to the model, and it will provide predictions based on what it has learned during training.

```
# Make predictions
predictions = model.predict(X_new_data)
```

Example: Building a Simple Neural Network with Keras

```
#1) Import the necessray libraries

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input

#2) For the tutorial, lets experiment with random data

# Generate random input data (features)
X = np.random.rand(num_samples, num_features)

# Generate random output labels (classes)
y = np.random.randint(0, num_classes, size=num_samples)

# Split the data into training and testing sets
split_ratio = 0.8
split_index = int(num_samples * split_ratio)

X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]

#3) Define the model

# Build the neural network model using Sequential API
model = Sequential([
    Input(shape=(num_features,)),
    Dense(6, activation='relu'),  # Hidden layer with 6 neurons
```

```
    Dense(num_classes, activation='softmax')  # Output layer with
num_classes neurons and softmax activation for classification
])

# Display a summary of the model architecture
model.summary()

#4) Compile the model

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

#5) Fit/train the model

# Train the model using the training data
epochs = 50
batch_size = 32
model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,
validation_split=0.1)


#6) Evaluate/test the model

# Evaluate the model on the testing data
loss, accuracy = model.evaluate(X_test, y_test, batch_size=batch_size)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

WEEK-1: GETTING STARTED WITH BUILDING FULLY CONNECTED NEURAL NETWORKS IN KERAS

Q1. Using the **Iris Flowers Dataset**, build and Neural Network with the following specifications to perform multi-class classification.

- Split the Data into Training: Validation: Testing = 80:10:10
- Number of Hidden Layers = 2, containing 8 Neurons and 4 Neurons
- Use RELU activation function in the hidden layers, choose the optimizer as ADAM and set learning rate to be equal to 0.1.

WEEK-2: EXPERIMENTING WITH DEEP NEURAL NETWORKS

Q1. Consider the following dataset 'Churn_Modelling.csv':
https://www.kaggle.com/datasets/aakash50897/churn-modellingcsv

The dataset has 14 features which are as follows :

- RowNumber:- Represents the number of rows
- CustomerId:- Represents customerId
- Surname:- Represents surname of the customer
- CreditScore:- Represents credit score of the customer
- Geography:- Represents the city to which customers belongs to
- Gender:- Represents Gender of the customer
- Age:- Represents age of the customer
- Tenure:- Represents tenure of the customer with a bank
- Balance:- Represents balance hold by the customer
- NumOfProducts:- Represents the number of bank services used by the customer
- HasCrCard:- Represents if a customer has a credit card or not
- IsActiveMember:- Represents if a customer is an active member or not
- EstimatedSalary:- Represents estimated salary of the customer
- Exited:- Represents if a customer is going to exit the bank or not.

1. Perform the required pre-processing and write comment lines to explain the pre-processing steps.

2. Perform experiments using (80,10,10) split and tabulate the performance in terms of Accuracy, Precision & Recall for the following experimental setup:
   a) Number of Hidden Layers and Number of Units per Layer

Number of Hidden Layers	Number of Units
1	128, 0 ,0
2	128, 64, 0
3	128, 64, 32

b) Epochs (10,20,30)
c) Activation function (Sigmoid, ReLU)
d) Without Regularization, with Regularization (L1/L2)
e) Learning rate (0.1, 0.01,0.001)

Visualize the training and validation loss against the epochs and comment on optimal hyperparameters.

Q2. Accurate measurement of body fat is inconvenient/costly, and it is desirable to have easy methods of predicting Body Fat. Using the given Body Fat dataset, build a Neural Network to predict body fat. Plot the training and validation performance curves and analyze the performance of the proposed neural network.

The attributes of the dataset are as follows:

1. Density determined from underwater weighing
2. Percent body fat from Siri's (1956) equation
3. Age (years)
4. Weight (lbs)
5. Height (inches)
6. Neck circumference (cm)
7. Chest circumference (cm)
8. Abdomen 2 circumference (cm)
9. Hip circumference (cm)
10. Thigh circumference (cm)
11. Knee circumference (cm)
12. Ankle circumference (cm)
13. Biceps (extended) circumference (cm)
14. Forearm circumference (cm)
15. Wrist circumference (cm)

Use the following hyperparameters/design choices for your neural network:

a. Split the data in the ratio Training: Validation: Testing = 80:10:10.
b. Perform Normalization using Standard Scalar.
c. Number of Hidden layers = 3 and number of units for each hidden layers are 128,64,32, respectively.

d. Use RELU activation function in the hidden layers, choose the optimizer as ADAM and set learning rate to be equal to 0.1.

Visualize the training and validation loss against the epochs and comment on optimal hyperparameters.

WEEK-3: CONVOLUTIONAL NEURAL NETWORKS (CNN) VS FULLY CONNECTED NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Consider the following datasets:

A. **Fashion MNIST dataset** [Fashion MNIST dataset, an alternative to MNIST (keras.io)],
B. **CIFAR-10 dataset** [CIFAR10 small images classification dataset (keras.io)]

For each of the Datasets A and B, do the following:

Q1. Understanding the Dataset and Pre-processing: Implement the following:

a. Compute and display the number of classes.
b. Compute and display the dimensions of each image.
c. Display one image from each class.
d. Perform normalization.

Q2. Performing experiments on Fully Connected Neural Networks (FCNN):

a. Design a FCNN which is most suitable for the given dataset: Experimentally choose the best network (the intuitions and learnings from the experiments you have performed in Week-1 and Week-2 will help you choose the hyperparameters for the network).
b. Train and test the network (choose the best epoch size so that there is no overfitting).
c. Plot the performance curves.

Q3. Performing experiments on a Convolutional Neural Networks (CNNs):

a. Design **CNN-1** which contains:
   - One Convolution layer which uses 32 kernels each of size 5x5, stride = 1 and, padding =0.
   - One Pooling layer which uses MAXPOOLING with stride =2.
   - One hidden layer having number of neurons = 100
b. Design **CNN-2** which contains:
   - Two back-to-back Convolution layers which uses 32 kernels each of size 3x3, stride = 1, and padding =0.
   - One Pooling layer which uses MAXPOOLING with stride =2.
   - One hidden layer having number of neurons = 100

Note: use ReLU activation function after each convolution layer.

c. Train and test the networks (choose the best epoch size so that there is no overfitting).
d. Plot the performance curves for CNN-1 and CNN-2.
e. Compare the performances of CNN-1 and CNN-2.

Q4. Compare the performances of FCNN and CNN.

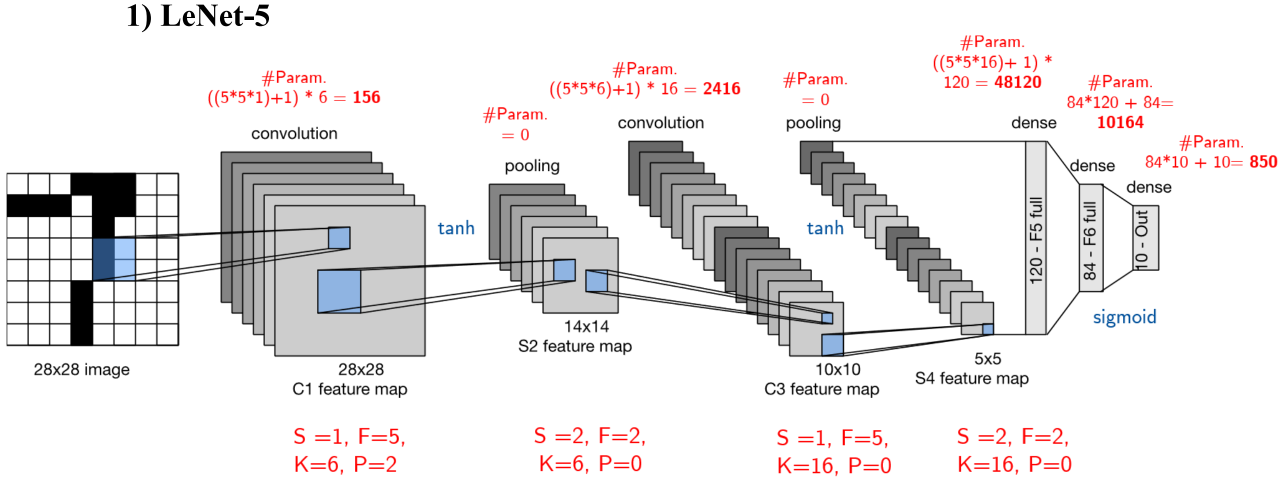
Q5. Compare the number of parameters in the FCNN and the CNN. Which layer/s in CNN contribute most to the total number of parameters.

WEEK-4: ADVANCED CNN ARCHITECTURES AND TRANSFER LEARNING FOR IMAGE CLASSIFICATION

Q1. Implementation and Comparison of CNN Architectures

A) Implement the following CNN architectures:

1) LeNet-5



2) AlexNet

B) Train, test, and compare the performance of these models on the following datasets:

Cats and Dogs Dataset:

Download from: https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip

Face Mask Detection Dataset:

Download from:
https://drive.google.com/file/d/1ejyQe12TIHjOHj6oT5dxHVwcwap41TuV/view?usp=sharing

Q2. Transfer Learning and Model Performance Analysis

Train, test, and report the performance of the following PRE-TRAINED IMAGENET models on the Cats and Dogs Dataset` and Face Mask Detection Dataset.

A) VGG-16
B) GoogleNet (InceptionV3)
C) ResNet50
D) EfficientNetB0
E) MobileNetV2

WEEK 5 – RECURRENT NEURAL NETWORKS

Q1. Use the following code to generate a time series:

```
def generate_time_series(sample_size, n_steps):
freq1, freq2, offsets1, offsets2 = np.random.rand(4, sample_size, 1)
time = np.linspace(0, 1, n_steps)
series = 0.5 * np.sin((time - offsets1) * (freq1 * 10 + 10)) #wave1+
series += 0.2 * np.sin((time - offsets2) * (freq2 * 20 + 20)) #wave2+
series += 0.1 * (np.random.rand(sample_size, n_steps) - 0.5) #noise
return series[..., np.newaxis].astype(np.float32)
```

The above code generates as many time series as requested, which can be specified using the "sample_size" argument. Each time series is of length "n_steps" and there is just one value per time step in each series.

Use the above code to do the following:

A) Create a dataset of 10,000 samples with 51 time steps each (Note: the 51st time step should be used as the label)

B) Split the dataset in the ratio training: validation: testing = 70:20:10.

C) Design, train, test and compare the performances of the following on the prediction of the value of 51st time step in the generated time series.

i. Simple RNN with one hidden layer and one output layer.

ii. Simple RNN with two hidden layers and one output layer.

Q2: Baltimore is a city is significantly known for high crime rate which ranks higher than the national average. Each crime record comes with both spatial (latitude and longitude) and temporal (date and time of occurrence) information along with the specific type of crime. This includes eleven different categories of crimes such as homicide, robbery, larceny etc.

https://www.kaggle.com/datasets/sohier/crime-in-baltimore

Consider the crimes of LARCENY (Crime code starting with 6), BURGLARY (Crime code starting with 5). Create two timeseries datasets LarcenyTs, BurglaryTs to represent the total number of crimes , day-wise. Put data from 2014, 2015 into training and predict the total number of LARCENY and BURGLARY crimes for the year 2016.

A)Build a Simple RNN model vs a LSTM model, both with 4 layers to predict the total number of LARCENY and BURGLARY crimes for the year 2016.

B) Compare and comment on their accuracy using MAPE, RMSE.

C) Comment on how many epochs are required for adequate learning.

D) Plot the actual vs predicted values using the test data for the year 2016.

WEEK 6 – SENTIMENT ANALYSIS & MINI PROJECT INITIATION

Q1. Consider the following data set:

https://www.kaggle.com/datasets/columbine/imdb-dataset-sentiment-analysis-in-csv-format

Keras Documentation : https://www.tensorflow.org/guide/keras/preprocessing_layers

1. Perform required text pre-processing – lowering text, removing URLs, punctuation, stop words and correct spelling.
2. Perform tokenization and lemmatization on cleaned data.
3. Visualize the most frequent words and bigrams.
4. Visualize the practical words that represent positive and negative sentiment in the dataset.
5. Create an embedding layer and build the following models:
   a. 3-layer LSTM,
   b. 3-Layer GRU and,
   c. 5-layer Bidirectional RNN for predicting the sentiment.
6. Train, Evaluate, Test the models. Plot the performance curves and tabulate accuracy. Which model performed the best?

Q2. DLT MINI PROJECT – Phase 1 Submission

Discuss the mini project idea, and complete the steps of Phase 1. Submit the soft copy of a report with the following contents:

1. Define the problem statement.
2. Study the metadata of your dataset
3. Perform Exploratory Analysis & Visualization
4. Define the Preprocessing pipeline specific to the data
5. Define Project Objectives

WEEK 7- MID TERM EXAMINATION

WEEK 8 – NEURAL MACHINE TRANSLATION

Q1. Using the following NMT repo:

[Tab-delimited Bilingual Sentence Pairs from the Tatoeba Project (Good for Anki and Similar Flashcard Applications) (manythings.org)]

1. Perform required text pre-processing.
2. Train and test the encode-decoder model with attention mechanism.
3. Use any pre-trained transformer model , fine tune it on given dataset for language translation tasks.
4. Compare the performance of model defined in (2) and (3).

Q2. DLT MINI PROJECT – Phase 2 Submission

1. Discuss the mini project, and complete the steps of Phase 2.

2. Submit Python Notebook with EDA & Preprocessing

3. Submit the soft copy of a report with the following contents:

- Literature review to identify models to be implemented
- Pros, Cons of each model
- Shortlist models for implementation
- Define baseline model
  - If No structure -> fully connected
  - If Spatial structure -> convolutional
  - If Sequential structure -> recurrent

WEEK-9: IMAGE RECONSTRUCTION AND IMAGE DENOISING USING AUTOENCODERS

Q1. Use the pins-face-recognition dataset (https://www.kaggle.com/datasets/hereisburak/pins-face-recognition) and perform the following tasks"

1. Preprocess the data, ensuring it is suitable for training an autoencoder.
2. Design an autoencoder architecture for image reconstruction. Train the model on the preprocessed dataset and evaluate the model's performance in terms of image reconstruction.
3. Visualize original images and their reconstructed counterparts to assess the quality of reconstruction.
4. Introduce noise to the images to create a noisy dataset.
5. Train the Autoencoder model using the noisy images as input and the clean images as target output.

6. Evaluate the model's performance in terms of noise reduction.
7. Visualize noisy images, their denoised counterparts, and the original clean images to observe the denoising effect of the model.

Q2. DLT MINI PROJECT – Phase 3 Discussion

- Define working end-to-end pipeline
- Determine your goals—what error metric to use, and target value.
  Ex: Accuracy, Coverage, Precision , Recall etc
- Diagnose performance and optimization curves
- Based on findings gather new data, adjust hyperparameters (learning rate, number of layers etc), or change architecture

WEEK-10: IMAGE GENERATION USING GENERATIVE ADVERSARIAL NETWORKS (GANs)

Q1. Consider the Fashion MNIST dataset as an input perform the following tasks:

1. Build the GAN model to generate a new dataset.
2. Experiment with 100, 150, 200 epochs, latent dimensions as 20, 50, 100 and visualize the impact of changing these hyperparameters on the quality of generated image.
3. Plot the generated image and save them as a new dataset.

Q2. DLT MINI PROJECT – Phase 3 Demonstration & Submission

1.Submit Python Notebook with running models

2. Submit the soft copy of a report with the following contents:

- Tabulation and visualization of results in terms of performance and accuracy , roc /prc etc
- Result analysis
- Comment on accuracy, performance
- Reasoning about hyperparameters
- Conclusion

WEEK-11: PHASE 4 DEMONSTRATION

- Deployment in app/cloud
- Peer Review

WEEK 12- MID TERM EXAMINATION