DL Lab 2024 - Live

Instructors: Dr. K. Giribabu | M. Pardha Saradhi | K. Gnanendra |
| Syllabus | MCQ-Practice | Viva Q&A | Term Project |

UPDATES

- 21-01-2025: Lab # 03 posted
- 03-01-2025: Lab # 02 posted
- 19-12-2024: Installation details and Lab # 01 posted

Installations

| Anaconda installation | TensorFlow Installation |

Experiments

#Lab	Title	Resources
1	Installation and working on Python, Juypter, and its different libraries for deep learning (Tensor Flow, NumPy, Keras, Pandas, Matplotlib, etc.) Submission deadlines: | CSM-A: **04-01-2025** | CSM-B: **02-01-2025** | | CSM-C: **02-01-2025** | AIM: **30-12-2024** |	• Practice, • Exercises & deliverables • Refer to: | 1 | 2 |
2	To implement a Multilayer Perceptron (MLP) using Keras with TensorFlow, and	• Datasets: Iris

	fine-tune neural network hyperparameters for classification problems (Iris and other datasets). Submission deadlines: | CSM-A: **11-01-2025** | CSM-B: **11-01-2025** | | CSM-C: **11-01-2025** | AIM: **06-01-2025** | **Note**: Refer to the practice lab report, and prepare your report for the exercises & deliverables similarly.	● Practice: ○ Read ○ Notebook ○ Report ● Exercises & deliverables
3	To implement an MLP using Keras with TensorFlow for regression problems (housing price prediction). Submission deadlines: | CSM-A: **xx-01-2025** | CSM-B: **xx-01-2025** | | CSM-C: **30-01-2025** | AIM: **02-02-2025** |	● Datasets: housing from sklearn ● Practice: ○ Read ○ Notebook ○ Report ● Exercises & deliverables
4	To implement image recognition (on MNIST dataset) without using CCNs Submission deadlines: | CSM-A: **xx-02-2025** | CSM-B: **xx-02-2025** | | CSM-C: **xx-02-2025** | AIM: **xx-02-2025** |	● Dataset: MNIST from TF ● Practice ○ Notebook
5	a) To understand the basic building blocks of CNN model b) To implement a Convolution Neural Network (CNN) for dog/cat classification problems using Keras (both Sequential and Functional API) Submission deadlines: | CSM-A: **xx-01-2025** | CSM-B: **xx-01-2025** | | CSM-C: **xx-01-2025** | AIM: **xx-01-2025** |	● Dataset: cats_dogs.zip ● Notebooks ○ Building_blocks.ipynb ○ cats_vs_dogs.ipynb ● Exercises & deliverables
6	To Implement a CNN for semantic segmentation in the given image. Submission deadlines: | CSM-A: **xx-01-2025** | CSM-B: **xx-01-2025** | | CSM-C: **xx-01-2025** | AIM: **xx-01-2025** |	● Dataset: ● Practice ● Notebook ● Exercises & deliverables
7	Implement a Long Short-Term Memory (LSTM) to predict the next word. Submission deadlines: | CSM-A: **xx-01-2025** | CSM-B: **xx-01-2025** | | CSM-C: **xx-01-2025** | AIM: **xx-01-2025** |	● Dataset: IndiaUS.txt ● Practice ● Notebook ● Exercises & deliverables

8	Implement a Recurrent Neural Network (RNN) to predict time series data. Submission deadlines: | CSM-A: **xx-01-2025** | CSM-B: **xx-01-2025** | | CSM-C: **xx-01-2025** | AIM: **xx-01-2025** |	● Dataset: yfinance ● Practice ● Notebook ● Exercises & deliverables
9	Design and implement a CNN model (with 4+ layers of convolutions) to classify multi category image datasets. Use the concept of regularization and dropout while designing the CNN model. Use the Fashion MNIST datasets. Record the Training accuracy and Test accuracy corresponding to the following architectures: a. Model with L1 Regularization b. Model with L2 Regularization c. Model with Dropout d. Model with both L2 (or L1) and Dropout	● Dataset: fashion_MNIST ● Practice ● Notebooks 09a_L1Regularization 09b_L2Regularization 09c_DropoutRegularization 09b_L2Dropout_Regularization ● Exercises & deliverables
10	Implement Auto encoders for image denoising on MNIST dataset.	● Dataset: MNIST ● Practice ● Notebook ● Exercises & deliverables

Sequential Vs Function API

In TensorFlow/Keras, the Sequential and Functional APIs are two different ways to define neural network models. Let us see detailed comparison of their differences:

1. Structure and Flexibility

- **Sequential Model**:

    - Linear stack of layers where each layer has exactly one input and one output
    - Layers are added sequentially in the order they are defined
    - Simpler and more straightforward to use
    - Limited to single-input, single-output architectures
- **Functional Model**:
    - More flexible architecture definition using a computational graph
    - Allows multiple inputs and/or multiple outputs
    - Supports complex architectures (skip connections, branching, shared layers)
    - Layers are connected explicitly by calling them as functions

2. Use Cases

- **Sequential Model**:
    - Ideal for simple, straightforward architectures
    - Good for beginners or when your model is a linear stack
    - Example: Basic CNNs, simple feedforward networks
- python

```
model = Sequential([
  Dense(64, activation='relu'),
  Dense(10, activation='softmax')
      ])
```

- **Functional Model**:
    - Better for complex models
    - Used for multi-input/output models, residual connections, etc.
    - Example: Siamese networks, U-Net, models with shared layers

- `python`

```
inputs = Input(shape=(32,))
x = Dense(64, activation='relu')(inputs)
outputs = Dense(10, activation='softmax')(x)
```

- model = Model(inputs, outputs)

3. Syntax and Definition

- **Sequential Model**:
  - Uses a list-like or `.add()` method approach
  - No explicit input/output definition needed
  - Layers are implicitly connected in order
- **Functional Model**:
  - Requires explicit Input layer definition
  - Layers are connected by calling them as functions on previous layers
  - Model is created by specifying inputs and outputs in `Model()`

4. Capabilities

- **Sequential Model**:
  - Cannot handle:
    - Multiple inputs/outputs
    - Layer sharing
    - Non-linear topologies
    - Skip connections

- **Functional Model**:
  - Can handle:
    - Multiple inputs (e.g., image + metadata)
    - Multiple outputs (e.g., classification + regression)
    - Layer sharing (same layer used multiple times)
    - Complex architectures (ResNet, Inception)

5. Visualization and Debugging

- Both support `model.summary()` to view architecture
- Functional models might be easier to debug complex flows since connections are explicit
- Sequential models are more compact but less transparent about data flow

Example Comparison

Sequential:

`python`

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

model = Sequential([
    Dense(64, activation='relu', input_shape=(32,)),
    Dense(10, activation='softmax')
])
```

Functional:

python

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

inputs = Input(shape=(32,))
x = Dense(64, activation='relu')(inputs)
outputs = Dense(10, activation='softmax')(x)
model = Model(inputs=inputs, outputs=outputs)
```

When to Use Each

- Use **Sequential** when:
  - Building simple, linear architectures
  - Quick prototyping
  - No need for complex connections
- Use **Functional** when:
  - Need multiple inputs/outputs
  - Implementing advanced architectures
  - Requiring layer sharing or skip connections
  - Building models that might need to be extended later

Practical Implications with Your CNN

Your original CNN could be implemented with either API because it's a linear stack. However:

- Sequential was sufficient for the simple flow
- Functional would be necessary if you later wanted to:
  - Add skip connections between convolutional layers

- Incorporate multiple inputs (e.g., image + metadata)
- Produce multiple outputs (e.g., classification + bounding box)

Both APIs produce identical computational graphs for linear architectures, but the Functional API gives you more control and options for future modifications.