Week 1: Introduction to Deep Learning

Task 1: Setup

- Install Python, Jupyter Notebook, TensorFlow, and Keras on your machine.
- Ensure all libraries are properly configured to run deep learning models.
- Resource: TensorFlow Installation Guide

Task 2: Basic

- Write a simple neural network from scratch using NumPy to understand basic concepts like neurons, layers, and activation functions.
- Resource: Neural Networks from Scratch

Task 3: Data Handling

- Load and preprocess the MNIST dataset using TensorFlow/Keras.
- Focus on normalizing data and converting labels to one-hot encoding.
- Resource: MNIST Dataset with Keras

Task 4: Model Building

- Build a basic feedforward neural network (FNN) for classification using Keras.
- Include input, hidden, and output layers.
- Resource: Building Your First Neural Network

Task 5: Training and Evaluation

- Train the FNN, evaluate its performance using metrics like accuracy, and visualize the results using Matplotlib.
- Resource: Evaluating Keras Models

Week 2: Convolutional Neural Networks (CNNs)

Task 1: Introduction to CNNs

- Implement a simple convolutional layer and pooling layer from scratch to understand their functions.
- Resource: Understanding Convolutional Layers

Task 2: Building CNNs

- Create a CNN using Keras for image classification on the CIFAR-10 dataset.
- Include convolutional, pooling, and dense layers.
- Resource: Keras CNN Tutorial

Task 3: Data Augmentation

- Implement data augmentation techniques (e.g., rotation, flipping, scaling) to improve the CNN's performance.

- Resource: Image Data Augmentation with Keras

Task 4: Model Optimization

- Experiment with different optimizers (e.g., SGD, Adam), learning rates, and regularization techniques.
- Resource: Optimizers in Keras

Task 5: Visualization

- Visualize the filters and feature maps learned by the CNN to understand its inner workings.
- Resource: Visualizing Intermediate Activations

Week 3: Recurrent Neural Networks (RNNs)

Task 1: Introduction to RNNs

- **Objective**: Understand the basics of RNNs.
- **Steps**:
  1. Implement a simple RNN from scratch.
  2. Learn how RNNs handle sequential data.
- **Resources**: Understanding RNNs

Task 2: Building RNNs

- **Objective**: Build an RNN for sequence modeling.
- **Steps**:
  1. Create an RNN using Keras for time series prediction or text generation.
- **Resources**: Keras RNN Tutorial

Task 3: Long Short-Term Memory (LSTM)

- **Objective**: Use LSTM networks for better sequence modeling.
- **Steps**:
  1. Implement an LSTM network for tasks like language modeling or stock price prediction.
- **Resources**: Understanding LSTMs

Task 4: Gated Recurrent Units (GRU)

- **Objective**: Compare LSTM and GRU networks.
- **Steps**:
  1. Implement a GRU network and compare its performance with LSTM.
- **Resources**: GRU in Keras

Task 5: Evaluation and Tuning

- **Objective**: Evaluate and tune RNN models.
- **Steps**:
    1. Use metrics like accuracy, precision, and recall.
    2. Tune hyperparameters like learning rate and batch size.
- **Resources**: Hyperparameter Tuning

Week 4: Advanced Neural Network Architectures

Task 1: Transfer Learning

- **Objective**: Use pre-trained models for new tasks.
- **Steps**:
    1. Implement transfer learning using a pre-trained model like VGG16 or ResNet.
- **Resources**: Transfer Learning with Keras

Task 2: Generative Adversarial Networks (GANs)

- **Objective**: Create synthetic data with GANs.
- **Steps**:
    1. Build a simple GAN to generate synthetic images.
- **Resources**: Introduction to GANs

Task 3: Autoencoders

- **Objective**: Use autoencoders for tasks like anomaly detection.
- **Steps**:
    1. Implement an autoencoder for dimensionality reduction or anomaly detection.
- **Resources**: Autoencoders in Keras

Task 4: Sequence-to-Sequence (Seq2Seq) Models

- **Objective**: Handle sequence-to-sequence tasks.
- **Steps**:
    1. Create a Seq2Seq model for tasks like machine translation or text summarization.
- **Resources**: Seq2Seq in Keras

Task 5: Attention Mechanisms

- **Objective**: Enhance Seq2Seq models with attention mechanisms.
- **Steps**:

1. Integrate attention mechanisms into your Seq2Seq models.

- **Resources**: Attention Mechanisms

Week 5: Specialized Deep Learning Techniques

Task 1: Object Detection

- **Objective**: Learn to detect objects in images.
- **Steps**:
  1. Implement an object detection model using a deep learning framework like TensorFlow or Keras.
  2. Explore models such as YOLO (You Only Look Once) or Faster R-CNN.
  3. Use a dataset suitable for object detection tasks (e.g., COCO dataset).
- **Resources**:

  1. TensorFlow Object Detection API
  2. YOLOv3 Implementation
  3. Faster R-CNN

Task 2: Semantic Segmentation

- **Objective**: Segment images into meaningful regions.
- **Steps**:

  1. Implement a semantic segmentation model using deep learning techniques.
  2. Explore architectures like U-Net, SegNet, or DeepLab.
  3. Use datasets such as Pascal VOC or ADE20K for training and evaluation.

- **Resources**:

  - U-Net Tutorial
  - SegNet Paper
  - DeepLab Paper

Task 3: Reinforcement Learning

- **Objective**: Explore learning through interaction and rewards.
- **Steps**:

  1. Implement a reinforcement learning agent using deep Q-learning or policy gradient methods.
  2. Choose an environment suitable for reinforcement learning tasks (e.g., OpenAI Gym).
  3. Train the agent to perform tasks like game playing or robotic control.

- **Resources**:

  - OpenAI Gym
  - Deep Reinforcement Learning with TensorFlow
  - Deep Q-Learning Paper

Task 4: Graph Neural Networks (GNNs)

- **Objective**: Apply deep learning to graph-structured data.
- **Steps**:

    1. Implement a Graph Neural Network (GNN) for tasks such as node classification or link prediction.
    2. Explore architectures like Graph Convolutional Networks (GCNs) or Graph Attention Networks (GATs).
    3. Use datasets with graph structures (e.g., Cora, Reddit) for training and evaluation.

- **Resources**:

    - Introduction to GNNs
    - Graph Convolutional Networks (GCNs)
    - Graph Attention Networks (GATs)

Task 5: Hyperparameter Optimization

- **Objective**: Optimize model performance with hyperparameter tuning.
- **Steps**:

    1. Use techniques like grid search or Bayesian optimization to find optimal hyperparameters.
    2. Explore tools like Keras Tuner or scikit-optimize for automated hyperparameter tuning.
    3. Apply the best hyperparameters to your deep learning models from previous tasks.

- **Resources**:

    - Hyperparameter Tuning with Keras Tuner
    - Bayesian Optimization with scikit-optimize

Week 6: Final Project and Presentation (continued)

Task 1: Project Planning

- **Objective**: Plan a comprehensive deep learning project.
- **Steps**:
    1. Define the scope and objectives of your project.
    2. Outline the tasks, timelines, and resources needed.
    3. Set milestones to track progress.
- **Resources**: Project Planning Guide

Task 2: Data Preparation

- **Objective**: Collect and preprocess data for your project.
- **Steps**:

    1. Gather and clean the dataset.
    2. Perform data augmentation if necessary.
    3. Split the data into training, validation, and test sets.

- **Resources**: Data Preprocessing Techniques

Task 3: Model Development

- **Objective**: Develop a deep learning model tailored to your project.
- **Steps**:

    1. Select the appropriate architecture (CNN, RNN, GAN, etc.).
    2. Implement the model using TensorFlow/Keras.
    3. Train the model on your dataset.

- **Resources**: Model Training Best Practices

Task 4: Evaluation and Refinement

- **Objective**: Evaluate and refine your model for optimal performance.
- **Steps**:

    1. Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.
    2. Fine-tune hyperparameters such as learning rate and batch size.
    3. Consider regularization techniques if needed.

- **Resources**: Model Evaluation Techniques

Task 5: Presentation

- **Objective**: Prepare and deliver a clear presentation of your project.
- **Steps**:

    1. Structure your presentation with an introduction, methodology, results, and conclusions.
    2. Use visuals (charts, graphs) to illustrate key findings.
    3. Practice presenting to ensure clarity and confidence.

- **Resources**: Effective Presentation Skills