CampusX Deep Learning Curriculum

A. Artificial Neural Network and how to improve them

1. Biological Inspiration

- Understanding the neuron structure
- Synapses and signal transmission
- How biological concepts translate to artificial neurons

2. History of Neural Networks

- Early models (Perceptron)
- Backpropagation and MLPs
- The "AI Winter" and resurgence of neural networks
- Emergence of deep learning

3. Perceptron and Multilayer Perceptrons (MLP)

- Single-layer perceptron limitations
- XOR problem and the need for hidden layers
- MLP architecture

4. Layers and Their Functions

- **Input Layer**
    - Accepting input data
- **Hidden Layers**
    - Feature extraction
- **Output Layer**
    - Producing final predictions

5. Activation Functions

- **Sigmoid Function**
  - Characteristics and limitations
- **Hyperbolic Tangent (tanh)**
  - Comparison with sigmoid
- **ReLU (Rectified Linear Unit)**
  - Advantages in mitigating vanishing gradients
- **Leaky ReLU and Parametric ReLU**
  - Addressing the dying ReLU problem
- **Softmax Function**
  - Multi-class classification outputs

6. Forward Propagation

- Mathematical computations at each neuron
- Passing inputs through the network to generate outputs

7. Loss Functions

- **Mean Squared Error (MSE)**
  - Used in regression tasks
- **Cross-Entropy Loss**
  - Used in classification tasks
- **Hinge Loss**
  - Used with SVMs
- Selecting appropriate loss functions based on tasks

8. Backpropagation

- Derivation using the chain rule
- Computing gradients for each layer
- Updating weights and biases
- Understanding computational graphs

9. Gradient Descent Variants

- **Batch Gradient Descent**
  - Pros and cons

- **Stochastic Gradient Descent (SGD)**
    - Advantages in large datasets
- **Mini-Batch Gradient Descent**
    - Balancing between batch and SGD

10. Optimization Algorithms

- **Momentum**
    - Accelerating SGD
- **Nesterov Accelerated Gradient**
    - Looking ahead to the future position
- **AdaGrad**
    - Adaptive learning rates
- **RMSProp**
    - Fixing AdaGrad's diminishing learning rates
- **Adam**
    - Combining momentum and RMSProp

11. Regularization Techniques

- **L1 and L2 Regularization**
    - Adding penalty terms to the loss function
- **Dropout**
    - Preventing overfitting by randomly dropping neurons
- **Early Stopping**
    - Halting training when validation loss increases

12. Hyperparameter Tuning

- **Learning Rate**
    - Impact on convergence
- **Batch Size**
    - Trade-offs between speed and stability
- **Number of Epochs**
    - Avoiding overfitting
- **Network Architecture**

    - Deciding depth and width
- Techniques:
    - Grid search
    - Random Search
    - Bayesian optimization

13. Vanishing and Exploding Gradients

- Problems in deep networks
- Solutions:
    - Proper weight initialization
    - Use of ReLU activation functions

14. Weight Initialization Strategies

- Xavier/Glorot Initialization
- He Initialization

15. Batch Normalization

- Normalizing inputs of each layer
- Accelerating training
- Reducing dependence on initialization

B. Convolution Neural Networks

1. Challenges with MLPs for Image Data

- High dimensionality
- Lack of spatial invariance

2. Advantages of CNNs

- Parameter sharing
- Local connectivity

3. Convolution Operation

- **Understanding Kernels/Filters**
  - Edge detection filters
  - Feature extraction
- **Mathematical Representation**
  - Convolution in 2D and 3D
- **Hyperparameters**
  - Kernel size, depth
- **Stride and Padding**
  - Controlling output dimensions
  - Types of padding: same vs. valid

4. Activation Functions

- **ReLU (Rectified Linear Unit)**
  - Advantages over sigmoid/tanh
- **Variants**
  - Leaky ReLU
  - ELU (Exponential Linear Unit)

5. Pooling Layers

- **Purpose**
  - Dimensionality reduction
  - Translation invariance
- **Types of Pooling**
  - Max pooling
  - Average pooling
- **Pooling Size and Stride**

6. Fully Connected Layers

- **Transition from Convolutional Layers**
- **Flattening**
  - Converting 2D features to 1D

7. Loss Functions

- Cross-Entropy Loss for Classification
- Mean Squared Error for Regression

8. CNN Architecture

Layer Stacking

- Convolutional -> Activation -> Pooling

Feature Maps

- Understanding depth and channels

Visualization

- Interpreting learned features

9. Data Preprocessing Techniques - Data Normalization

- **Scaling Pixel Values**
  - 0-1 normalization
  - Standardization (z-score)

10. Data Preprocessing Techniques -Data Augmentation

- **Techniques**
  - Rotation, flipping, cropping
  - Color jitter, noise addition
- **Purpose**
  - Reducing overfitting
  - Increasing dataset diversity

CNN Architectures and Innovations

11. LeNet-5

- **Architecture Details**

- Layers, activations

- **Contributions**
  - Handwritten digit recognition

12. AlexNet

- **Breakthroughs**
  - Deeper network
  - Use of ReLU
- **Impact on ImageNet Challenge**

13. VGG Networks

- **VGG-16 and VGG-19**
- **Design Philosophy**
  - Using small filters (3x3)
  - Deep but uniform architecture

14. Inception Networks (GoogLeNet)

- **Inception Modules**
  - Parallel convolutional layers
- **Motivation**
  - Efficient computation

15. ResNet (Residual Networks)

- **Residual Blocks**
  - Identity mappings
  - Shortcut connections
- **Solving Vanishing Gradient Problem**
- **Variants**
  - ResNet-50, ResNet-101

16. MobileNets

- Depthwise Separable Convolutions

- Optimizations for Mobile Devices

17. Pre-trained Models & Transfer Learning

- Using Models Trained on ImageNet
- Fine-Tuning vs. Feature Extraction

Object Detection and Localization [Optional]

18. Traditional Methods

- Sliding Window Approach

19. Modern Architecture

- **Region-Based CNNs (R-CNN)**
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- **You Only Look Once (YOLO)**
- **Single Shot MultiBox Detector (SSD)**
- **Mask R-CNN**
  - Instance segmentation

Semantic Segmentation

20. Fully Convolutional Networks (FCN)

- Replacing Fully Connected Layers

21. U-Net

- Encoder-Decoder Architecture
- Skip Connections

Generative Models with CNNs

22. Autoencoders

- **Convolutional Autoencoders**
    - Image reconstruction
- **Variational Autoencoders (VAE)**

23. Generative Adversarial Networks (GANs)

- **DCGAN**
    - Using CNNs in GANs
- **Applications**
    - Image generation
    - Super-resolution

C. Recurrent Neural Networks

1. Architecture of RNNs

- Sequential Data Challenges
- Basic RNN Structure
- Mathematical Formulation
- Activation Functions

2. Forward Propagation Through Time

- **Sequence Input Processing**
    - Handling variable-length sequences
- **Output Generation**
    - At each time step or after the entire sequence

3. Backpropagation Through Time (BPTT)

- **Unfolding the RNN**
    - Treating RNN as a deep network over time
- **Calculating Gradients**
    - Applying the chain rule through time steps
- **Computational Complexity**
    - Memory and computation considerations

4. Challenges in Training RNNs

- **Vanishing Gradients**
  - Gradients diminish over long sequences
- **Exploding Gradients**
  - Gradients grow exponentially
- **Solutions**
  - Gradient clipping
  - Advanced architectures (e.g., LSTMs, GRUs)

5. LSTM

- LSTM core components
- Gates in LSTM
- Intuition Behind LSTMs
- Backpropagation Through Time

6. GRU

- GRU core components
- Gates in GRU
- Intuition Behind GRU
- Backpropagation in GRUs
- GRU vs LSTM

6. Deep RNNs

- Stacking RNN layers
- Vanishing and Exploding Gradients in Deep RNNs
- Using LSTM and GRU
- Solution and techniques to overcome VGP and EGP
- Residual Connections
- Regularization

7. Bidirectional RNNs

- Motivation behind Bidirectional RNNs

- Bidirectional RNN architecture
- Forward and Backward pass
- Combining outputs
- Bidirectional LSTM

8. Applications of RNNs

- Language modeling - Next word prediction
- Sentiment Analysis
- POS Tagging
- Time series forecasting

Seq2Seq Networks

1. Encoder-Decoder Networks

A. Introduction to Encoder-Decoder Architecture

- **Purpose and Motivation**
  - Handling variable-length input and output sequences.
  - Essential for tasks like machine translation, text summarization, and speech recognition.

B. Components of Encoder-Decoder Networks

- **Encoder**
  - Processes the input sequence and encodes it into a fixed-length context vector.
  - **Architecture**: Typically uses Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), or Gated Recurrent Units (GRUs).

- **Decoder**
    - Generates the output sequence from the context vector.
    - **Architecture**: Similar to the encoder but focuses on producing outputs.

C. Mathematical Formulation

- **Encoder and Decoder Equations**

D. Implementation Details

- **Handling Variable-Length Sequences**
    - **Padding**: Adding zeros to sequences to ensure uniform length.
    - **Masking**: Ignoring padded elements during computation.
- **Loss Functions**
    - **Cross-Entropy Loss**: Commonly used for classification tasks at each time step.
- **Training Techniques**
    - **Teacher Forcing**: Using the actual output as the next input during training to speed up convergence.

E. Limitations of Basic Encoder-Decoder Models

- **Fixed-Length Context Vector Bottleneck**
    - Difficulty in capturing all necessary information from long input sequences.
- **Solution Overview**
    - Introduction of attention mechanisms to allow the model to focus on relevant parts of the input sequence.

---

2. Attention Mechanisms and Their Types

A. Motivation for Attention

- **Overcoming the Bottleneck**
    - Attention allows the model to access all encoder hidden states rather than compressing all information into a single context vector.
- **Benefits**
    - Improved performance on long sequences.
    - Enhanced ability to capture alignment between input and output sequences.

B. Types of Attention Mechanisms

1. Additive Attention (Bahdanau Attention)

- **Concept**
    - Calculates alignment scores using a feedforward network
- **Characteristics**
    - Considered more computationally intensive due to additional parameters.

2. Multiplicative Attention (Luong Attention)

- **Concept**
    - Calculates alignment scores using dot products.
    - **Scaled Dot Product**: Adjusts for dimensionality.
- **Characteristics**
    - More efficient than additive attention.

C. Attention Mechanism Steps

1. Calculate Alignment Scores
2. Compute Attention Weights
3. Compute Context Vector
4. Update Decoder State

D. Implementing Attention in Seq2Seq Models

- **Integration with Decoder**

    - Modify the decoder to incorporate the context vector at each time step.
- **Training Adjustments**
    - Backpropagate through the attention mechanism.

E. Visualization and Interpretation

- **Attention Weights Matrix**
    - Visualizing which input tokens the model attends to during each output generation step.
- **Applications**
    - Error analysis.
    - Model interpretability.

---

3. Transformer Architectures

A. Limitations of RNN-Based Seq2Seq Models

- **Sequential Processing**
    - RNNs process inputs sequentially, hindering parallelization.
- **Long-Term Dependencies**
    - Difficulty in capturing relationships between distant tokens.

B. Introduction to Transformers

- **Key Innovations**
    - **Self-Attention Mechanism**: Allows the model to relate different positions of a single sequence to compute representations.
    - **Positional Encoding**: Injects information about the position of the tokens in the sequence.
- **Advantages**
    - Improved parallelization.
    - Better at capturing global dependencies.

C. Components of Transformer Architecture

1. Multi-Head Self-Attention

- **Concept**
  - Multiple attention mechanisms (heads) operating in parallel.
- **Process**
  - **Query (Q)**, **Key (K)**, and **Value (V)** matrices are computed from input embeddings.
  - The attention mechanism calculates a weighted sum of the values, with weights derived from the queries and keys.

2. Positional Encoding

- **Purpose**
  - Since transformers do not have recurrence or convolution, positional encoding provides the model with information about the position of each token.
- **Techniques**
  - **Sinusoidal Functions**:
  - **Learned Embeddings**

3. Feedforward Networks

- **Architecture**
  - Position-wise fully connected layers applied independently to each position.
- **Activation Functions**
  - Typically ReLU or GELU.

4. Layer Normalization

- **Purpose**
  - Normalizes inputs across the features to stabilize and accelerate training.

5. Residual Connections

- **Purpose**
    - Helps in training deeper networks by mitigating the vanishing gradient problem.
- **Implementation**
    - Adding the input of a layer to its output before applying the activation function.

D. Transformer Encoder-Decoder Structure

- **Encoder Stack**
    - Composed of multiple identical layers, each containing:
        - Multi-head self-attention layer.
        - Feedforward network.
- **Decoder Stack**
    - Similar to the encoder but includes:
        - Masked multi-head self-attention layer to prevent positions from attending to subsequent positions.
        - Encoder-decoder attention layer.

E. Implementing Transformers

- **Key Steps**
    - **Embedding Layer**: Converts input tokens into dense vectors.
    - **Adding Positional Encoding**: Combines positional information with embeddings.
    - **Building Encoder and Decoder Layers**: Stack multiple layers as per the architecture.
    - **Output Layer**: Generates final predictions, often followed by a softmax function.

---

4. Types of Transformers

A. BERT (Bidirectional Encoder Representations from Transformers)

- **Purpose**
  - Pre-training deep bidirectional representations by jointly conditioning on both left and right context.
- **Architecture**
  - Uses only the encoder part of the transformer.
- **Pre-Training Objectives**
  - **Masked Language Modeling (MLM)**: Predicting masked tokens in the input.
  - **Next Sentence Prediction (NSP)**: Predicting if two sentences follow each other.

B. GPT (Generative Pre-trained Transformer)

- **Purpose**
  - Focused on language generation tasks.
- **Architecture**
  - Uses only the decoder part of the transformer with masked self-attention to prevent information flow from future tokens.
- **Training Objective**
  - **Causal Language Modeling (CLM)**: Predicting the next word in a sequence.

C. Other Notable Transformers

- **RoBERTa**
  - Improves on BERT by training with larger batches and more data.
- **ALBERT**
  - Reduces model size by sharing parameters and factorizing embeddings.
- **T5 (Text-to-Text Transfer Transformer)**
  - Treats every NLP task as a text-to-text problem.

---

5. Fine-Tuning Transformers

A. Concept of Fine-Tuning

- **Transfer Learning**
  - Adapting a pre-trained model to a downstream task with task-specific data.

B. Steps in Fine-Tuning

1. **Loading Pre-Trained Model**
   - Use pre-trained weights from models like BERT, GPT, etc.
2. **Modifying Output Layers**
   - Replace the final layer to suit the specific task (e.g., classification head).
3. **Adjusting Hyperparameters**
   - Learning rate, batch size, number of epochs.
4. **Training on Task-Specific Data**
   - Use labeled data relevant to the task.

C. Best Practices

- **Layer-Wise Learning Rates**
  - Apply different learning rates to different layers.
- **Avoiding Catastrophic Forgetting**
  - Use smaller learning rates to prevent the model from losing pre-trained knowledge.
- **Regularization Techniques**
  - Dropout, weight decay.

D. Common Fine-Tuning Tasks

- **Text Classification**
- **Named Entity Recognition**
- **Question Answering**
- **Text Summarization**

---

6. Pre-Training Transformers

A. Pre-Training Objectives

- **Masked Language Modeling (MLM)**
  - Predicting masked tokens in the input sequence.
- **Causal Language Modeling (CLM)**
  - Predicting the next token given the previous tokens.
- **Sequence-to-Sequence Pre-Training**
  - Used in models like T5.

B. Data Preparation

- **Corpus Selection**
  - Large and diverse datasets (e.g., Wikipedia, Common Crawl).
- **Tokenization Strategies**
  - **WordPiece**: Used by BERT.
  - **Byte-Pair Encoding (BPE)**: Used by GPT.

C. Training Strategies

- **Distributed Training**
  - Using multiple GPUs or TPUs.
- **Mixed Precision Training**
  - Reduces memory usage and increases speed.
- **Optimization Algorithms**
  - Adam optimizer with weight decay (AdamW).

D. Challenges in Pre-Training

- **Compute Resources**
  - Requires significant computational power.
- **Data Quality**
  - Noisy data can affect model performance.

E. Evaluation of Pre-Trained Models

- **Benchmarking**
  - Using datasets like GLUE, SQuAD to assess performance.
- **Ablation Studies**
  - Understanding the impact of different components.

---

7. Optimizing Transformers

A. Computational Challenges

- **High Memory Consumption**
  - Due to self-attention mechanisms.
- **Long Training Times**

B. Optimization Techniques

1. Efficient Attention Mechanisms

- **Sparse Attention**
  - Reduces the number of computations by focusing on local patterns.
- **Linearized Attention (Linformer)**
  - Approximates attention to reduce complexity.
- **Reformer**
  - Uses locality-sensitive hashing to reduce complexity.

2. Model Compression

- **Quantization**
  - Reducing the precision of weights (e.g., from 32-bit to 8-bit).
- **Pruning**
  - Removing less important weights or neurons.
- **Knowledge Distillation**

    - Training a smaller model (student) to replicate the behavior of a larger model (teacher).

C. Hardware Considerations

- **GPUs vs. TPUs**
    - TPUs can offer faster computation for tensor operations.
- **Parallelism Strategies**
    - **Data Parallelism**
        - Distributing data across multiple devices.
    - **Model Parallelism**
        - Distributing the model's layers across devices.

D. Software Tools

- **Optimized Libraries**
    - **Hugging Face Transformers**: Provides optimized implementations.
    - **DeepSpeed**: Optimizes memory and computation.
    - **NVIDIA Apex**: Enables mixed precision training.

---

8. NLP Applications Using Transformers

A. Text Classification

- **Sentiment Analysis**
    - Classifying text as positive, negative, or neutral.
- **Topic Classification**
    - Categorizing text into predefined topics.

B. Question Answering

- **Implementing QA Systems**
    - Using models like BERT to find answers within a context.

- **Datasets**
    - SQuAD, TriviaQA.

C. Machine Translation

- **Transformer Models**
    - Implementing translation systems without RNNs.
- **Datasets**
    - WMT datasets.

D. Text Summarization

- **Abstractive Summarization**
    - Generating concise summaries using models like T5.
- **Datasets**
    - CNN/Daily Mail, Gigaword.

E. Language Generation

- **Chatbots**
    - Creating conversational agents using GPT models.
- **Story Generation**
    - Generating coherent narratives.

F. Named Entity Recognition

- **Sequence Labeling**
    - Identifying entities like names, locations, dates.
- **Fine-Tuning**
    - Adapting pre-trained models for NER tasks.