Neural Networks (econnerty/NN-Study (github.com))

Architectures:

- **Feed-Forward Neural Networks**
  - A simple neural network where inputs only travel in one direction.
  - Can be broken down into the **input layer, hidden layers, and output layer**
  - Layers can be understood as filters or mathematical operations that are performed on the input
  - Input Layers map features from the input data to weights and biases in the hidden layer. These weights and biases shape and modify the data until it is passed to the output layer, where it can be used
- **Convolutional Neural Networks (CNNs)**
  - A special type of Feed-Forward Neural Network that uses at least one convolutional layer to filter and extract features from an input tensor.
  - Usually used with multi-dimensional tensors such as images for image classification
- **Autoencoder**
  - Another type of Feed-Forward Neural network used to **reduce dimensionality** of the input, and then reconstruct that data using a trained decoder
  - Composed of two parts:
    - Encoder - Trained to reduce the dimensionality of an input and to create a unique encoding for each input
    - Decoder - Trained to reconstruct the encoded data back into its original format
- **Generative Adversarial Network (GAN)**
  - Similar to an autoencoder in structure, a GAN is used for image generation, synthetic data generation, and drug discovery amongst other things.
  - Composed of two parts:
    - Generator- Trained to create some sort of output, such as a picture
    - Discriminator - Trained to classify an output from the generator as either real and fake
  - Each component is trained
- **Recurrent Neural Networks (RNNs)**
  - A type of neural network that is used to find patterns in a sequence of data
  - Contains the concept of “state” within its neural network architecture. An RNN is able to remember previous inputs and update its “state” to more effectively make judgements based on what it has seen.
  - In the context of NLP, this allows RNNs to keep track of each word in a sentence, even though the data is passed in sequence
- **Graph Neural Networks (GNNs)**
  - A type of neural network designed to handle graph like unstructured data composed of “nodes” and “edges”
  - Graph neural networks are trained in such a way that they are not dependent on the order of the inputs they receive to get a given output. The order of edges can be changed in the input data, and the same predictions will be observed after passing through the network
- **Transformers**
  - A specialized type of feed-forward neural network that is used to process text data for various different applications such as chatbots, sentiment analysis, text-completion, and more recently: music generation
  - Uses a **self-attention mechanism** to learn the weights and importance of each word in a given input.
  - A tokenizer is used to assign unique IDs to each word in a given input, and these IDs are used to map input words to their “weights” or “importance”
  - Throughout the training process, the neural network learns how much emphasis to place on certain words, and how important they are to the sentences meaning
  - Can process text in a vectorized format, instead of sequentially (think word by word), offering dramatic performance improvements over RNNs

Components:

- **Layers**
  - Layers contain neurons which are trained during the training process to get desirable outcomes from the neural network
  - Some different types are **Linear Layers (aka fully connected layers), Convolutional Layers, Pooling Layers, and Dropout Layers.**
- **Activation Functions**

    - Specialized functions in a neural network that introduce non-linearity and help mitigate “vanishing gradients”
- **Loss Functions**
    - Provide a way to calculate the difference between the “expected output” and the “actual output” of a neural network
    - Loss is fed to the optimizer to be improved upon in a process known as “training”
- **Optimizers**
    - Specialized algorithms for training and optimizing a neural network
    - Use a technique known as **backpropagation** that takes advantage of a feature known as **automatic differentiation** to update parameters and reduce loss at each training step

Process:

1. **Data pre-processing**
    a. Usually entails manual labeling of training data, noise-reduction, normalization, and various other methods of data preparation
    b. In the case of a transformer model, a tokenizer is used to pre-process text data before being fed to the network. This process gives unique ID’s to each word, so that they can be processed and assigned weights
2. **Model initialization**
    a. A model architecture is defined, and its various parameters and transformations are initialized and sent to the GPU
3. **Optimizer and Loss Function selection**
    a. An optimizer and Loss function are selected for a specific task and initialized with their various hyperparameters
4. **Training Loop**
    a. The model is trained on the input data, and repeatedly updated in a process known as **backpropagation**.
    b. These parameter updates are made possible by a technique known as **automatic differentiation:** a process that keeps track of all changes made to the input tensor in a dynamic computational graph
    c. This computational graph is used to update the parameters of the model with help from the loss function and optimizer.
5. **Model Evaluation**
    a. The model is evaluated against a test data set for accuracy and performance