**Course 3: Types of Neural Network Arhitectures**

**Convolutional Neural Network (CNN):**

* Convolutional neural networks are primarily designed for processing grid-like data, such as images or videos.
* They employ specialized layers called convolutional layers that automatically learn local patterns and features from the input data.
* CNNs are known for their ability to capture spatial and hierarchical representations, making them highly effective in tasks like image classification, object detection, and image generation.

**Recurrent Neural Network (RNN):**

* Recurrent neural networks are designed to process sequential data, where the order of the data points matters.
* They have feedback connections, allowing information to flow in loops within the network, making them capable of capturing temporal dependencies.
* RNNs are commonly used in tasks such as natural language processing, speech recognition, and time series analysis.
* However, traditional RNNs can struggle with long-term dependencies due to the vanishing or exploding gradient problem.

**Long Short-Term Memory (LSTM):**

* Long Short-Term Memory is a type of recurrent neural network architecture that addresses the vanishing gradient problem.
* LSTMs are designed to capture long-term dependencies in sequential data by using a memory cell, which can selectively retain or forget information.
* They have proven to be effective in tasks that involve long sequences, such as language modeling, machine translation, and speech recognition.
* LSTMs have become a widely used variant of recurrent neural networks due to their ability to retain important information over longer time periods.

**Feedforward Neural Network:**

* Feedforward neural networks are the simplest form of artificial neural networks. They are composed of interconnected layers of perceptrons or artificial neurons.
* In a feedforward neural network, the data flows in one direction, from the input layer through one or more hidden layers to the output layer.
* Here is an example of a feedforward neural network with an input layer, multiple hidden layers, and an output layer:
* Input Layer (features) -> Hidden Layer 1 -> Hidden Layer 2 -> ... -> Hidden Layer N -> Output Layer (predictions)
* As the number of hidden layers increases, the network gains the ability to learn more complex patterns and representations.

**Design Choices for Feedforward Neural Networks:**

* Activation Function: The activation function determines the output of a neuron given its input. It introduces non-linearity into the network and allows it to learn complex relationships.
* Loss Function: The loss function measures the difference between the predicted output of the network and the true output. It guides the training process by quantifying the network's performance.
* Output Units: The choice of output units depends on the nature of the problem. For binary classification, a single output unit with a suitable activation function is used. For multi-class classification, multiple output units (equal to the number of classes) with softmax activation are often employed.
* Architecture: The architecture of a feedforward neural network refers to the arrangement and connectivity of its layers and neurons. This includes the number of hidden layers and the number of neurons in each layer.

**Activation Functions:**

Activation functions introduce non-linearity to the network, allowing it to learn complex patterns and make non-linear transformations. Some commonly used activation functions include:

* Sigmoid function
* Hyperbolic tangent (tanh) function
* Rectified Linear Unit (ReLU) function
* Leaky ReLU function
* Generalized ReLU (ReLU variants with different slopes)
* Softplus function
* Swish function

**Loss Functions for ANN's:**

Loss functions quantify the difference between predicted outputs and true outputs. Different loss functions are used depending on the nature of the problem. Some common loss functions for ANN's include:

* Mean Absolute Error Loss
* Mean Squared Error Loss
* Negative Log-Likelihood Loss
* Cross-Entropy Loss
* Hinge Embedding Loss

**Output Functions:**

The choice of output functions depends on the problem at hand:

* For binary classification, a common choice is the sigmoid function, which maps the output to a probability between 0 and 1.
* For multi-class classification, the softmax function is often used to obtain a probability distribution across different classes.

**Universal Approximation Theorem:**

The Universal Approximation Theorem states that a feedforward neural network with a single hidden layer and a finite number of neurons can approximate any continuous function to arbitrary precision, given a sufficiently large number of neurons in the hidden layer.

**Training a Feedforward Neural Network:**

* The training process of a feedforward neural network involves iteratively adjusting the weights and biases of the neurons based on the training data and the specified loss function.
* Backpropagation is a commonly used algorithm for training feedforward neural networks. It calculates the derivatives of the error with respect to the weights and biases, allowing for the adjustment of these parameters through gradient descent.

**Computing the Derivatives of Error:**

* When training an ANN, the goal is to minimize the error between the predicted outputs and the actual outputs.
* The derivatives of the error with respect to the network's parameters (weights and biases) are computed using a technique called gradient descent.
* Gradient descent calculates the gradient of the error function with respect to each parameter, indicating the direction and magnitude of the steepest descent.
* By iteratively updating the parameters in the opposite direction of the gradient, the network gradually moves towards the optimal set of parameters that minimize the error.

**Back-propagation of Errors:**

* Back-propagation is an algorithm used to calculate the gradients of the error with respect to the parameters in a multi-layer neural network.
* It involves two phases: forward propagation and backward propagation.
* In the forward propagation phase, the input data is fed through the network, and the activations of each neuron are computed layer by layer, eventually resulting in the predicted output.
* In the backward propagation phase (back-propagation), the error is propagated backward from the output layer to the input layer, while calculating the gradients of the error with respect to the network's parameters.
* These gradients are then used to update the parameters in the direction that minimizes the error, using the gradient descent optimization algorithm.

Back-propagation allows the network to adjust its parameters based on the errors made during the forward propagation phase, gradually improving its performance over multiple iterations of training.