**Weekly Assessment**

1. **Components and functionality of CNN layers:**

Convolutional layers:

* Purpose: Extract features from input data
* Functionality: Apply filters (kernels) to input, performing convolution operations
* Parameters: Number of filters, filter size, stride, padding

Pooling layers:

* Purpose: Reduce spatial dimensions, retain important features
* Functionality: Typically use max pooling or average pooling
* Parameters: Pool size, stride

Fully connected layers:

* Purpose: Combine features for final classification/regression
* Functionality: Connect all neurons from previous layer to current layer
* Parameters: Number of neurons, activation function

Example application in computer vision: A CNN for image classification might have this structure:

1. Convolutional layer (extract low-level features)
2. Pooling layer (reduce dimensions)
3. Convolutional layer (extract higher-level features)
4. Pooling layer
5. Fully connected layer (combine features)
6. Output layer (classification)

This structure could be used for tasks like classifying images into categories (e.g., cats vs. dogs).

1. **Differences between traditional machine learning and deep learning**:

Feature engineering:

* Traditional ML: Often requires manual feature engineering
* Deep learning: Automatically learns relevant features

Model complexity:

* Traditional ML: Generally simpler models (e.g., linear regression, decision trees)
* Deep learning: More complex models with many layers and parameters

Data requirements:

* Traditional ML: Can work with smaller datasets
* Deep learning: Generally requires large amounts of data for effective training

Interpretability:

* Traditional ML: Often more interpretable
* Deep learning: Often considered a "black box" approach

Computational resources:

* Traditional ML: Generally requires less computational power
* Deep learning: Often requires significant computational resources (GPUs)

1. **Gradient Descent, SGD, and Adam optimizer:**

Gradient Descent:

* Updates weights using the entire dataset in each iteration
* Advantage: Stable convergence
* Disadvantage: Slow for large datasets

Stochastic Gradient Descent (SGD):

* Updates weights using a single random sample in each iteration
* Advantage: Faster, can escape local minima
* Disadvantage: Higher variance in updates

Adam optimizer:

* Adaptive learning rate optimization algorithm
* Combines ideas from momentum and RMSprop
* Advantage: Efficient, handles sparse gradients well
* Disadvantage: May converge to suboptimal solutions in some cases

1. **Forward propagation and backpropagation:**

Forward propagation:

* Process of moving from input layer to output layer
* At each neuron: Compute weighted sum of inputs, apply activation function
* Weights and biases determine the strength of connections between neurons

Backpropagation:

* Process of computing gradients and updating weights
* Steps:
  1. Compute loss at output layer
  2. Compute gradients of loss with respect to weights and biases
  3. Propagate gradients backwards through the network
  4. Update weights and biases using an optimization algorithm (e.g., SGD)

Weight and bias updates:

* Typically use gradient descent: new\_weight = old\_weight - learning\_rate \* gradient
* Learning rate determines the size of updates