Mid term report

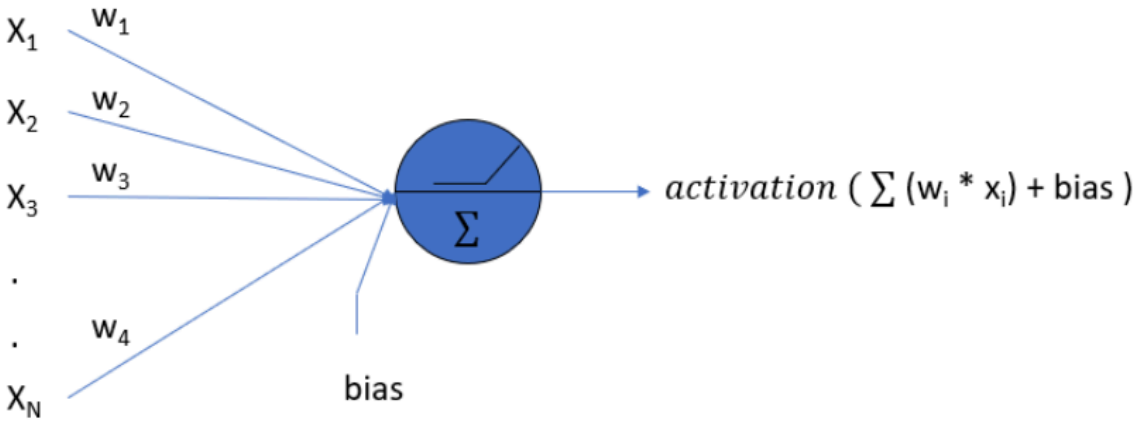
Basics of neural Network

Neurons and Layers:

Neurons and Layers are the basic/fundamental building blocks of neural networks. A single neuron is a basic unit that receives input, processes it through an activation function, and passes the output to the next layer. Layers in a neural network are input layer, hidden layer and output layer:

- **Input layer:** The first layer that receives the input data.
- **Hidden Layer:** Intermediate layers where computations are performed, enabling the network to learn complex patterns.
- **Output layer:** The final layer that produces the prediction or classification result.

Each neuron in a layer is connected to neurons in the next layer through weights. The learning process is all about adjusting these weights to minimise the error between predicted and actual output.

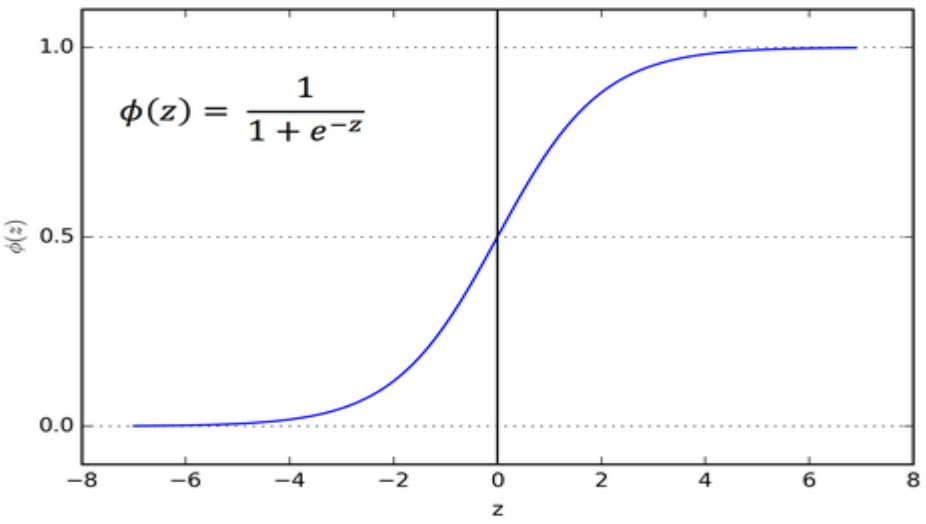


A single neuron shown with X_i inputs with their respective weights W_i and a bias term and applied activation function

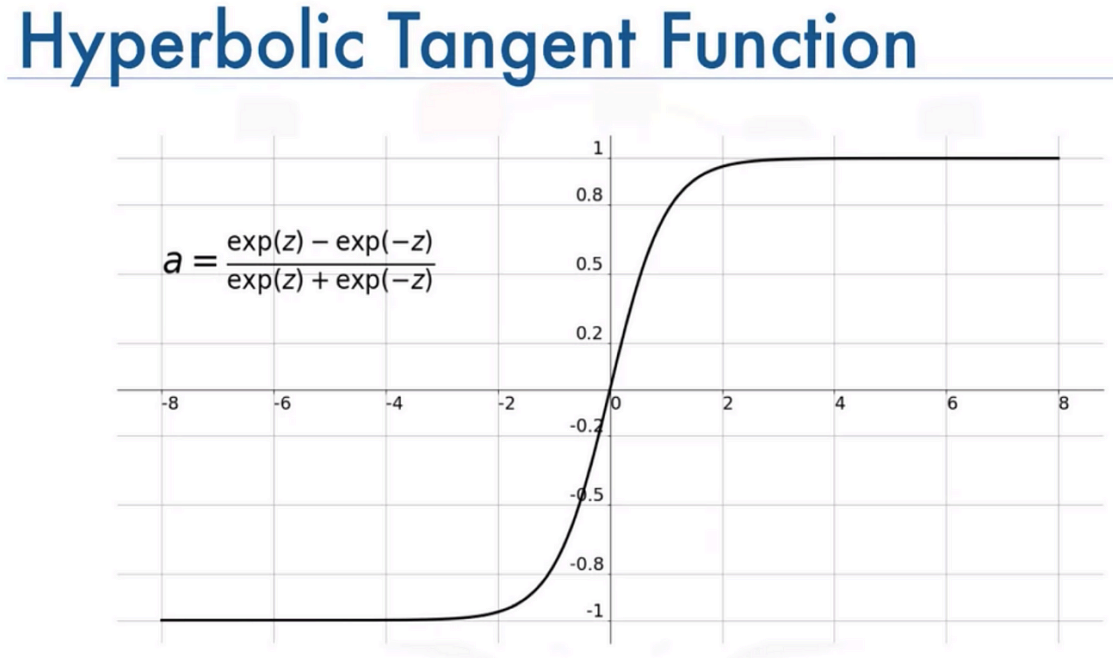
Activation function

Activation functions introduce non-linearity into network, allowing it to lern complex relationships. If activation function are removed from the layers than complete neural network can be compressed into a single layer. Common activation functions include:

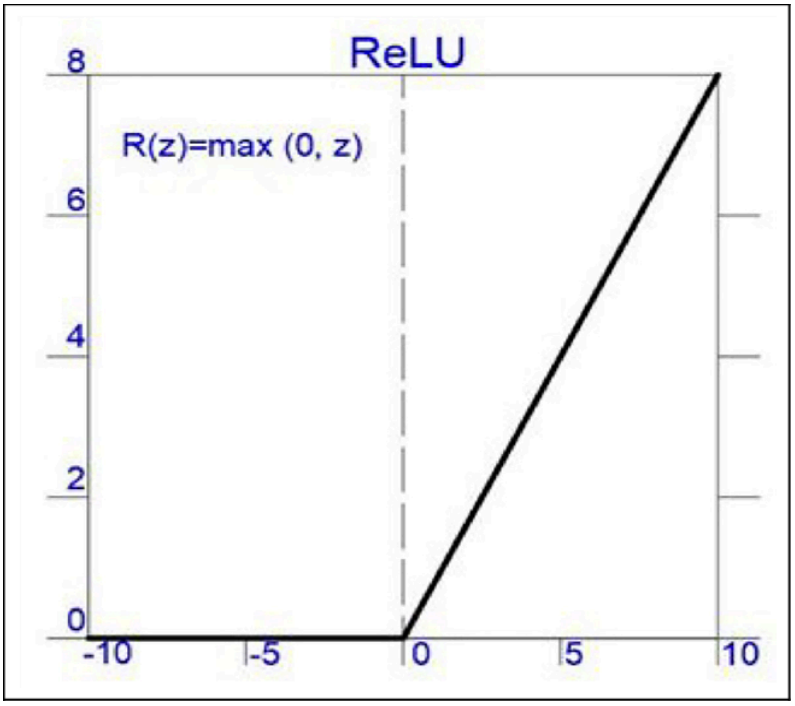
- **Sigmoid**
    1. Squashes the neurons pre-activation between 0 and 1
    2. Always positive
    3. Bounded
    4. Strictly increasing



- **Hyperbolic tangent**
    1. Squash the neuron's preactivation between -1 and 1
    2. Can be positive or negative
    3. Bounded
    4. Strictly increasing



- **Rectified linear activation**
    1. Bounded below by zero
    2. Not upper bound
    3. Strictly increasing
    4. Tends to give neuron with sparse activation



Feedforward Process

The feed forward process is the mechanism by which data moves through the network:

- **Input:** Raw data is fed into the input layer.
- **Forward propagation:** Data moves through each layer, where neurons perform weighted sums and apply activation functions.
- **Output:** The final prediction is generated at the output layer.

During forward propagation, the network transforms input data through multi layers, extracting features and patterns.

Training Neural network/ Backpropagation

Backpropagation is the key algorithm for training neural networks. It involves:

- **Forward pass**: Compute the output and compare it to the actual value using a loss function.
- **Loss calculation:** Compute the error between predicted and actual outputs

- **Backward pass**: Compute the gradient of the loss function with respect to each weight using the chain rule
- **Weight update**: Adjust weights to minimise the loss using optimization algorithms.

Empirical risk minimization

The process of choosing a function from a hypothesis space that minimises the empirical risk, which is the average loss on the training data.

- Framework to design the learning algorithm:

$$\hat{\mathbf{w}} \in \underset{\alpha \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} \underbrace{\left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)}\right)^2}_{\text{empirical risk}} + \lambda \underbrace{\left(\mathbf{w}^T \mathbf{x}^{(i)} - \alpha u^{(i)}\right)^2}_{\text{subjective explainability}}$$

- Subjective explainability also called as regularizer (here) penalises certain value of weights.

Stochastic gradient descent

Algorithm that performs update after each example

$$\underbrace{\mathbf{w}^{(t+1)}}_{\substack{\text{position of next} \\ \text{iteration}}} = \underbrace{\mathbf{w}}_{\substack{\text{position of} \\ \text{previous step}}} - \underbrace{\alpha \nabla f_i(\mathbf{w}^{(t)})}_{\text{step}}$$

learning rate

observation *i*

Regularisation

- L2 regularisation:

$$\text{L2:} \qquad R(\theta) = \|\theta\|_2^2 = \sum_{i=1}^{n} \theta_i^2$$

    1. Only applied on weights not on bais
    2. Can be interpreted as having gaussian prior over the weights

- L1 regularisation:
    1. This can also be applied on weights
    2. Unlike L2, L1 will push certain weights to be exactly zero.
    3. Can be interpreted to have laplacian prior over the weights

$$\text{L1:} \qquad R(\theta) = \|\theta\|_1 = \sum_{i=1}^{n} |\theta_i|$$

Optimization Algorithms

Optimization algorithms adjust the network's weights to minimize the loss function. Common algorithms include:

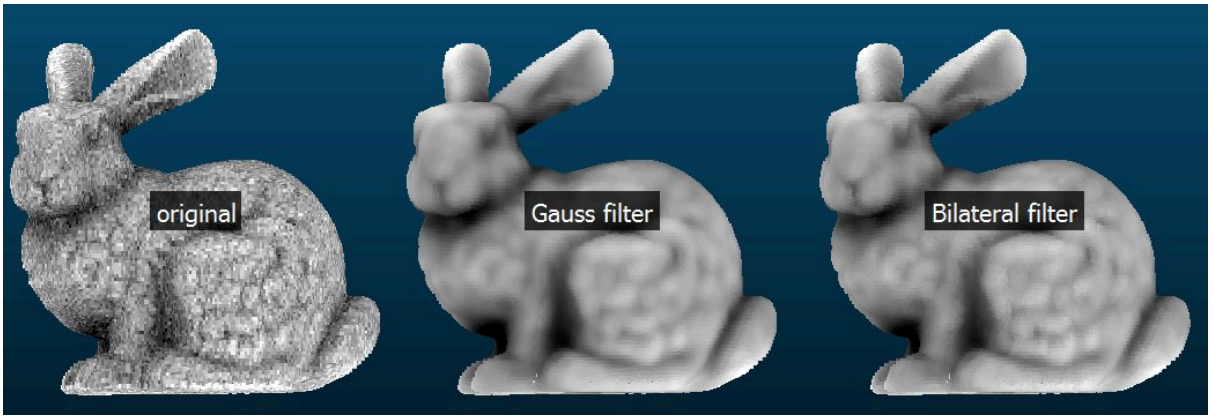
- **Gradient Descent:** Iteratively updates weights in the direction of the negative gradient.
- **Adam (Adaptive Moment Estimation):** Combines the advantages of two other extensions of stochastic gradient descent, AdaGrad and RMSProp, and is well-suited for large datasets and high-dimensional parameter spaces.
- **RMSprop (Root Mean Square Propagation):** Adjusts the learning rate for each parameter, improving convergence for non-stationary problems.

Classical computer vision algorithm (OpenCV) based Image processing

Basic Image Processing Techniques

Image Filtering: Image filtering is a process used to enhance images or extract useful information. Common filters include:

- **Gaussian Filter:** Smoothens the image by averaging the pixels within a kernel, reducing noise.
- **Median Filter:** Reduces noise by replacing each pixel with the median value in its neighborhood.



Edge Detection: Edge detection identifies boundaries within an image. Common techniques include:

- **Sobel Operator:** Computes gradients in the image to detect edges.
- **Canny Edge Detector:** Uses a multi-stage algorithm to detect edges robustly.

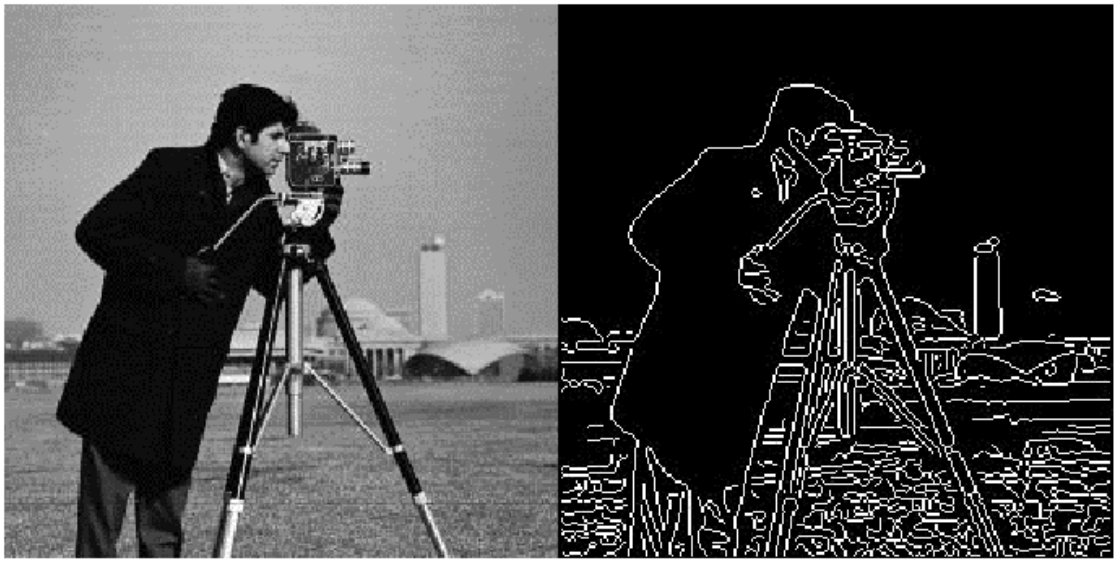


Figure 1: The cameraman image and its edges extracted

Histogram Equalization: Histogram equalisation improves the contrast of an image by spreading out the most frequent intensity values. This technique is useful for enhancing the details in images with poor contrast.

Convolutional Neural Networks

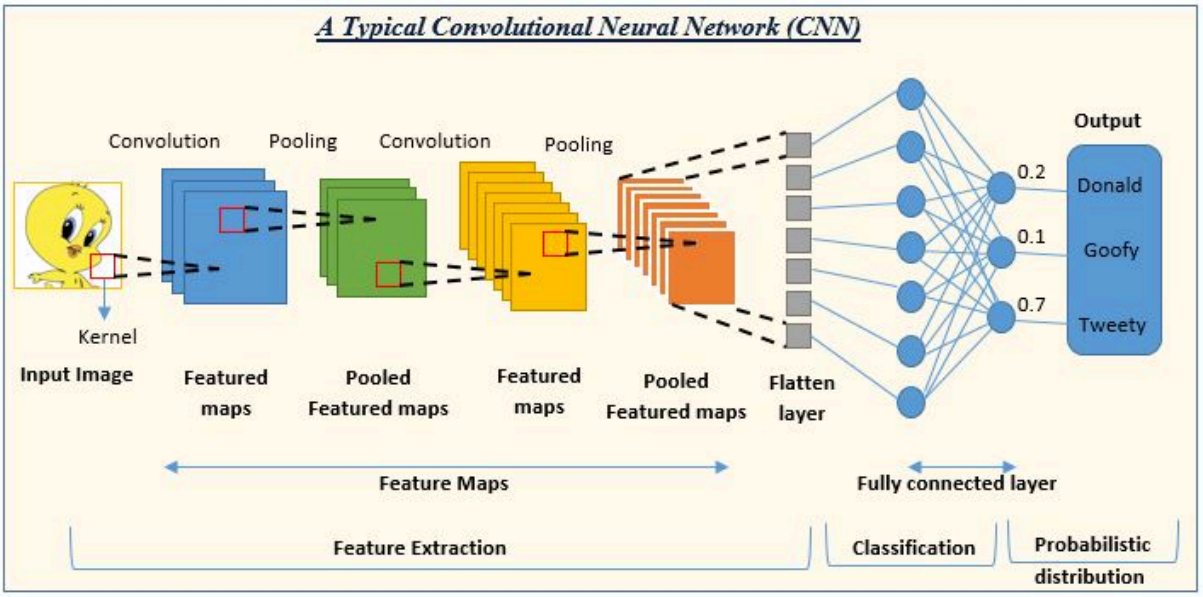
1. Basics of CNNs

Convolutional Layers: Convolutional layers apply convolution operations to the input, using filters (kernels) to extract features like edges, textures, and patterns. Each filter slides over the input, performing element-wise multiplication and summing the results to produce feature maps.

Pooling Layers: Pooling layers reduce the spatial dimensions of the feature maps, retaining the most important information. Common pooling operations include:

- **Max Pooling:** Selects the maximum value from each region.
- **Average Pooling:** Computes the average value of each region.

Fully Connected Layers: Fully connected layers are traditional neural network layers where each neuron is connected to every neuron in the previous layer. They are used at the end of the CNN to produce the final output.



2. Training CNNs

Backpropagation in CNNs: The backpropagation algorithm updates the weights in CNNs by calculating gradients through the convolutional and fully connected layers. The process involves:

1. **Forward Pass:** Compute the output of the CNN and calculate the loss.

2. **Backward Pass:** Compute the gradient of the loss with respect to each weight using the chain rule.
3. **Weight Update:** Adjust weights using optimization algorithms.

Regularization Techniques: Regularization techniques prevent overfitting by adding constraints to the model. Common techniques include:

- **Dropout:** Randomly sets a fraction of the neurons to zero during training to prevent co-adaptation.
- **Weight Decay:** Adds a penalty to the loss function based on the magnitude of the weights.

Hyperparameter Tuning: Hyperparameter tuning involves adjusting parameters like learning rate, batch size, and number of epochs to improve the performance of the CNN