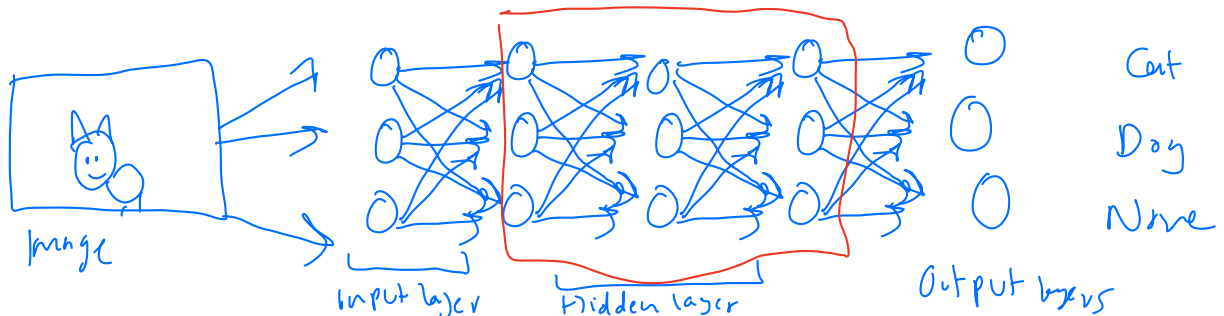
1.1) Gaussian mixture technique is a technique used in machine learning modeling where the model is used for representing subpopulations within a larger population, with the subpopulations being modeled as a Gaussian distribution. The mixture model assumes that data is a part of many different distributions, and in this case it is Gaussian. Each component in this mixture is a gaussian distribution. The components of Gaussian mixture are the component means (μ), the mean of each Gaussian distribution, covariances (Σ), the covariance matrices that describe the shape and orientation of each Gaussian distribution, and the mixing coefficients (π) which are the weights that determine how much each component contributes to the overall mixture in the model. It uses the expectation-maximization algorithm. It is used for anomaly detection by using the responsibility, or the probability that a point x_i belongs to a specific component. If a data point has a low probability of falling in the many gaussian distributions of the GMM, it is classified as an anomaly.

1.2)



Layers

Convolutional → ReLU → Pooling → Fully connected

Input layer: Raw pixel values of input image

Convolutional layer: A collection of kernels applied to input image to produce feature maps. It captures features like texture, pattern, and edges from an input picture.

ReLU Activation layer: Introduces non-linearity to model by turning negative values from feature map to 0s.

Pooling Layer: for dimensionality reduction. This layer reduces image dimensions while keeping the most important features. Different types of algorithms like max pooling & average pooling exist.

Fully connected layer: Flattens feature maps, ultimately using features to make predictions, outputs vector of class scores

Output layer: Uses softmax function to convert class scores into probabilities.

A couple of the latest CNN algorithms are ResNet, DenseNet, and VGGNet.

1.3) Vanishing gradients problems in backpropagation is when the gradient of the loss function diminishes as it is propagated. This causes weights near the input layer to not be updated properly. In turn, deep networks do

not learn long-range dependencies. Exploding gradients is when the gradient of the loss function grows exponentially. This leads to overly large weight updates, causing weights to not be optimal. Some techniques to combat these problems are Xavier initialization, leaky ReLU, gradient clipping, and batch normalization.

2) Observed error rate $\hat{p} = 20/100 = 0.2$, $z = 1.96$

$$\hat{p} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

$$0.2 \pm z\sqrt{\frac{.2(1-.2)}{100}} = (.1216, .2784)$$