Case Study VIII

Krushang Shah

Prof. Richard Khan

Senior Design I

ID 23821856

# Convolutional Neural Network

Convolutional Neural Networks are mainly used in the domain of Computer Vision. They are used to enable machines to view the worlds similar to humans, and even use the Intelligence and Knowledge for a multitude of tasks such as Video and Image recognition, Image classification, Image analysis, Media Recreation, Natural Language Processing, Recommendation systems and many more. The advancements in Computer Vision and Deep Learning have been perfected over time, mainly using one main algorithm, which is Convolutional Neural Networks.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm, which can take an input image, process it through multiple filters, biases and learnable weights. The pre-processing required in a CNN is way lower than that required in a general classification algorithms. While in older methods, filters were designed by humans with enough training, CNN's have an ability to learn these filters and characteristics overtime by themselves.

The architecture of a CNN is similar to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field, which is also known as receptive field. A collection of such distinct fields overlaps to cover the entire visual area.

A general Convolutional Neural Network has multiple hidden layers, and an input and output layer. The name convolutional neural network indicates that it uses a mathematical operation called convolution. Convolution is a special kind of linear operation. They use convolution instead of a general matrix multiplication in at least one of their layers.

**Convolutional Layer:** It is the core of the model, which consists of a set of learnable filters. During the pass, each layer is convolved along the width and height of the input volume.

**Local Connectivity:** Convolutional networks exploit spatial local correlation by enforcing a sparse local connectivity pattern between neurons and each neuron is connected to a small region of the input volume.

**Spatial arrangement:** The spatial arrangement is controlled by depth, stride and zero-padding of the convolutional layer, which determines the size of the output volume.

**Parameter sharing:** This scheme is used in convolutional layers, to control the number of free parameters, assuming that if a patch feature is useful to compute at some spatial position, then it can also be used to compute other positions.

**Pooling layer:** There are multiple types of pooling layers like max pooling, which takes the maximum in each X*X layer and takes the maximum from each block. The average pooling layer is similar which takes average instead of maximum of each blocks.

**ReLU layer:** ReLU stands for rectified linear unit, which applies the non-saturating activation function. That is it removes negative values from an activation map by setting them to zero.

**Fully connected layer:** After several convolutional and pooling layers, the high-level reasoning is done via fully connected layers.

**Loss layer:** This layer specifies how training penalizes the deviation between predicted output and true labels, usually this is the final layer of a neural network.

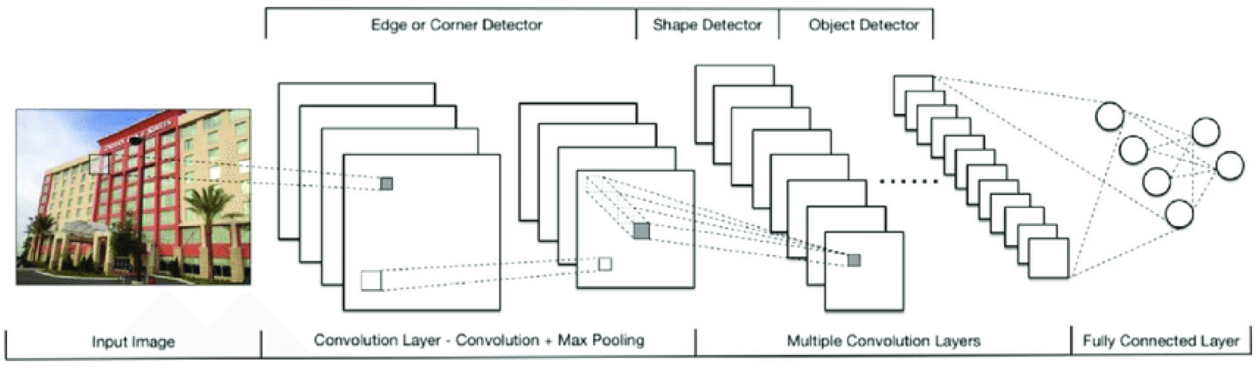


*Figure 1: General example of a Convolutional Neural Network*

The above shown diagram shows an example of a Convolutional Neural Network. Generally, a neural network can be separated into two main parts. One is Feature Learning where there are different types of mathematical and activation functions happening. And the second part is the classification, which combines the results from the previous layers, flattens the input into a tensor from a matrix and then classifies them based on different class descriptions. Here for example, an image of robot is used as an input, which then processes through Convolutional layer and ReLU layer, the convolutional layer convolutes the input image to make a feature map, using filters and filter biases. The ReLU layer removes all the negative values and only works with positive values, which improves the time footprint. Then it passes through pooling layer, which decreases the size of input by applying max pooling or other types of pooling. The neural network uses multiple instances of Convolutional layer and pooling layers to improve the accuracy of the model, but not overfit on the training data. The next part of a Convolutional Neural Network is the Classification part. After processing the input data from multiple layers, it finally passes to the flattening layer, where the data gets flattened from a matrix or grid form to a tensor. The next layer is the fully connected layer. In the fully connected layer, all the neurons are connected with the activation functions from the previous layers. The next step is passing the values through softmax function to take the maximum of one object and make others zero. The final step is the classification of the input image into the appropriate category based on the value obtained in the probability.

## Loss Layer

The loss layer determines how the model would be penalized while training, when there is a difference between the predicted output, that is output obtained from the model, and the correct output or the true label. This layer is usually the final layer of the network. The selection of the loss function for the loss layer depends on the task that the model is designed for.

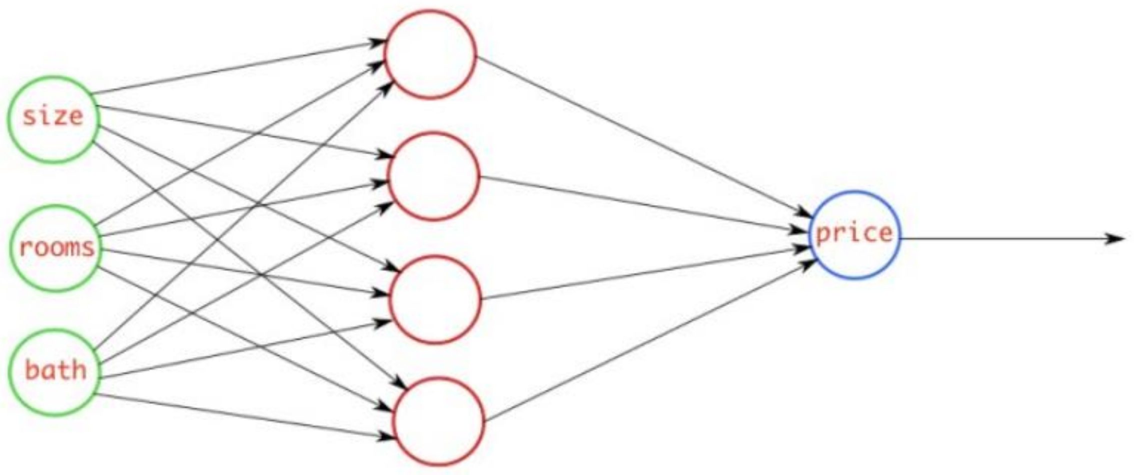
The loss is like the prediction of error of the Neural net. The loss is used to calculate the gradients, and the gradients are used to update the weights of the neural network, so that the next cycle would have an improved accuracy. This is the usual process of training of a neural net, that it learns from its mistakes and errors through feedback.

There are different types of loss functions, which can be used for most of the objectives, below mentioned are some essential loss functions.

## Mean Squared Error (MSE)

Mean Squared Error loss is usually used for regression tasks. Implied from the name, this loss function calculates the loss by taking the mean of the squared differences between the predicted values and the target values.

Ex.



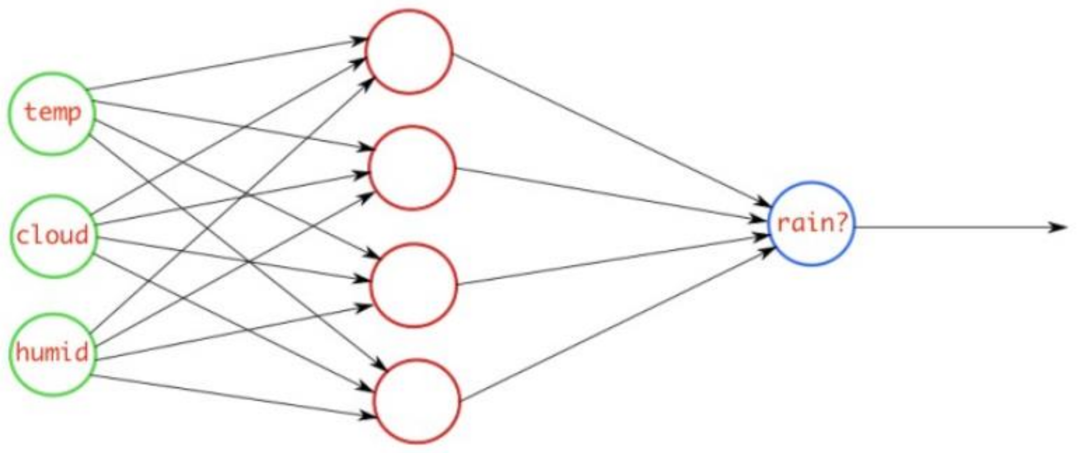
Mean Squared Error

This is a neural network that takes in the collected house data as the input and predicts the price of the house. Here, MSE loss function can be used to calculate the loss of each network while training. Usually, the MSE loss function can be used when the output is a real number.

## Binary Cross Entropy (BCE)

The Binary Cross Entropy is used for the binary classifications. For BCE loss function, there is only one output node needed, which classifies the data into two classes. This can be done by passing the output through a sigmoid activation function, which gives the range of output between 0 and 1.

Ex.



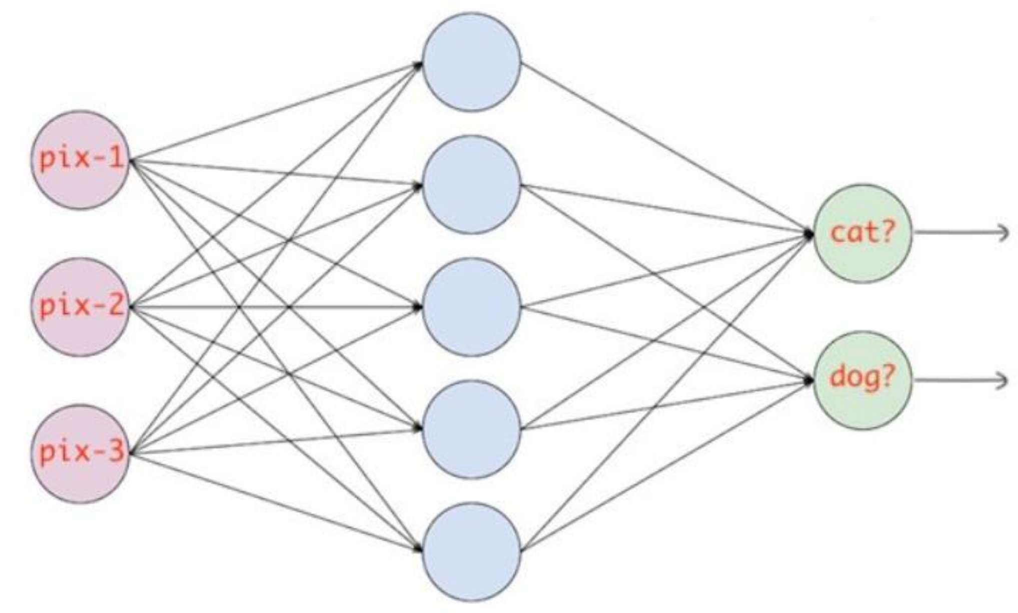
Binary Crossentropy

This network takes in the parameters and predicts if it will rain or not. If the output is greater than 0.5, it classifies as it will rain, while output less than 0.5 will mean that it will not rain. Hence, the BCE loss function can be used when there are only two possibilities of output.

## Categorical Cross Entropy (CCE)

This type of Categorical Cross Entropy loss function can be used when we have a multi-class classification as the output of our network. To use this type of loss function, you need to have the same number of classes as the output nodes. The final layer output needs to be passed through the softmax function, in order to get the probability of each node output between 0 and 1, and the sum of probabilities to 1.

Ex.



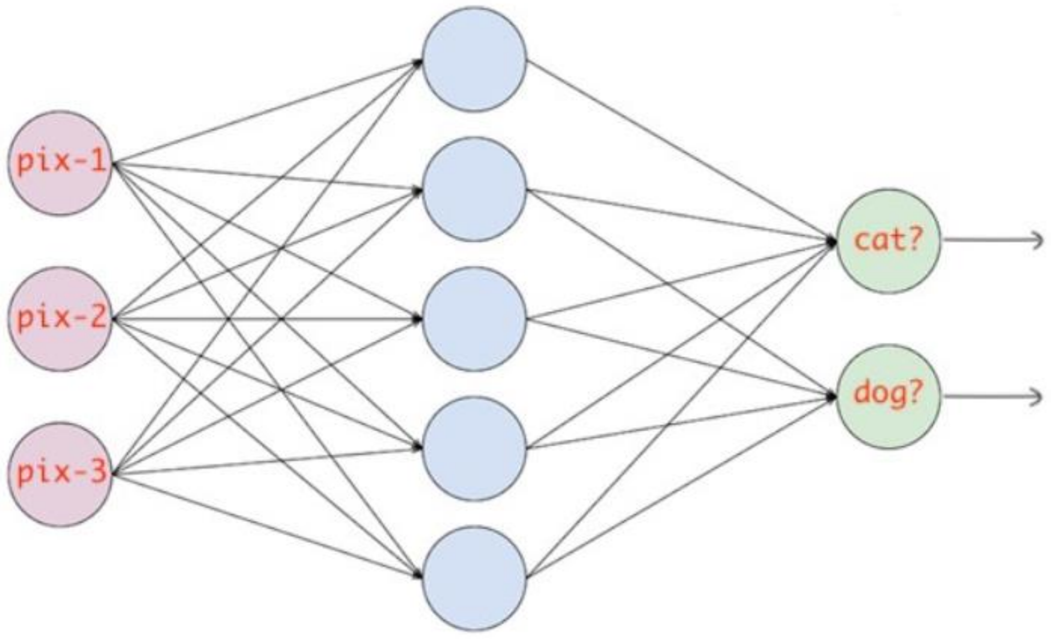
Categorical Crossentropy

This network takes in the parameters and predicts the type of animal. Here, if the cat node has higher probability from the two, then image will be classified as cat, and if the dog node has probability higher than cat, then the image will be classified as dog. For using this type of loss function, the target vector of the output needs to be of the same size as the number of classes, where each index of target corresponds to actual class. One of the elements in the target output vector will be 1 and other elements will be 0.

## Sparse Categorical Cross Entropy (SCCE)

It is almost similar to CCE except that for the output, the one hot encode on the target vector is not needed. That is only single output will be passed, which is the index of the class that the output belongs to.

Ex.



Sparse Categorical Crossentropy

Here, if the target image is of a cat, then you pass 1, if not then the output is a dog, hence you pass 2. Essentially you only need to pass the index of the class that the predicted output belongs to.