Neural networks: Deep neural networks

A four layer DNN will look like

Input layer

Hidden layer

Hidden layer

Output Layer

For our basics we are going to start Fully connected layer where every node on each layer is connected to every other node in next layer: fully connected neural network.

Hidden Layer is like functions, the code inside the black box is the hidden layer. Input layer of 3 layers, output layer of 4 nodes, total 12 connections. Every of these connections have weights/parameters. More the weights, more complex the model is.

DNN/CNN are feed forward neural networks. In these we pass input as parameters (input layer), the function performs a sequenced set of actions based on the inputs (hidden layer) and outputs as the results. two styles to code these:

**Sequential API method:** We are going to import sequential from keras. In this we add one layer at a time, until the output layer.

**Functional API model**: We build layers first and then we tie it all together. Using this we are taking the advantage of polymorphism. we are going to import model class, from keras.layers we are going to import layers class.

Finally we are going to instantiate the model parameters.

Input shape vs input layers

Every element in the input vector will be passed to every node in the input layer.

Input layers in the number of nodes in the input layers.

Dense layer

It is defined as having “n” number of nodes.

Activations

These are the squashing functions. We can understand Activation function as the signal and this signal is how strongly do I believe that something is true.

Rectified Linear unit (reLU): For any value less than zero it will be zero (no signal)

Compile

Compile() adds in the backward propagation during training. Each time we send data forward through the neural network, the neural network calculates the loss(predicted results) from the actual values (labels) and uses the information to adjust the weights and bias at the nodes. This is how the model is “learning”. We can mention what method we wanna use to calculate the loss.

DNN binary classifier: Logistic classifier

Instead of using ReLU, on the output node, we use the sigmoid activation function, It maps all the values in the range from [0,1]

DNN Multi-class Classifier

We are going to predict more than one class. Eg, we want to classify baby/toddler/teen/adult from height/weight/gender/surface-area (body measurements) as parameters.

TO do this we will have a node for every class in the output layer. Every node at the output will do the prediction on what it is. Every output node will independently learn and predict its confidence on whether the input is the corresponding class or no.

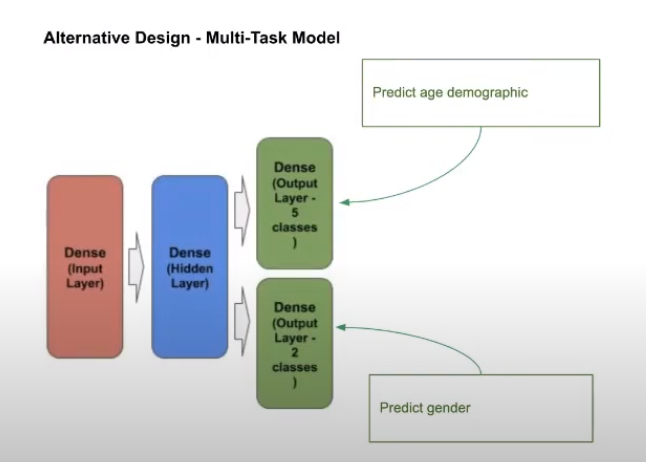
For this we are going to use Softmax activation function and the loss function will be ‘categorical\_crossentropy’

Multilabel multiclass classifier

To predict two or more classes(labels) per input. In the above example if we wanna:

Input - > [height, weight, nose surface area]

Output/prediction -> [age group, gender]



Source: Google clouds presentation

At the output level we want two different outputs, we will have two different output dense layers

Overfitting

Overfitting can lead model to memorise data not generalize it. When we will use that model on test data then accuracy will be very less.

Regularization is the method to address overfitting while training neural networks. To avoid overfitting we use regularization and one of the best methods is “dropout”, dropout is like forgetting.

Model

Model.add(Dropout(0.5))

CNN

Convolution (frontend) + Deep neural network(backend)

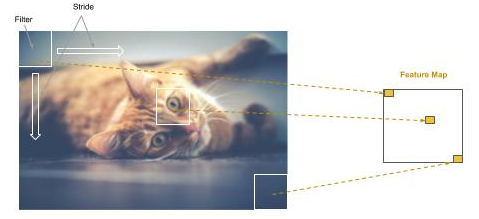
Why do we need convolution layers?

Convolution layers transforms the image from higher dimensional pixel based image to lower dimensionality features based image. In short feature detection is convolution, Performs the task of feature detection within an image.

Resize/Downsampling… to have fewer pixels to start with.

**Filters**: the sliding of filters over the image. Mostly the filters are square, 3X3 is most common

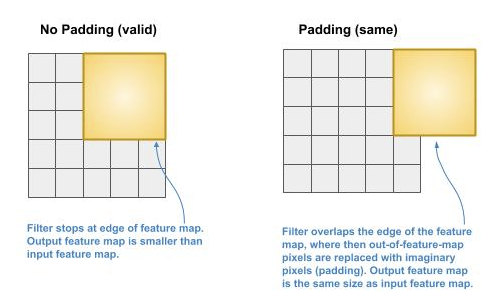
Typical number of filters are multiples of 16



Feature map maintains the relationship back to image. Here stride is the rate that filter is slid across the image. Stride=1, then filter advances one pixel at a time, thus filter will partially overlap the previous step.

Padding:

What will I do when I get to the edge when I’m striding in the image above? When padding is on, we are creating the same data for filter when we are at the corner.



Thumb rule:

MORE FILTER 🡪 MORE DATA

BIGGER STRIDES 🡪 LESS DATA

Down sampling types:

**Pooling**: Feature maps are resized to a smaller dimension.

max pooling

mean pooling

The common practice is a 2x2 pool size with a stride of 2. This will result in a 75% reduction in pixel data.

**Feature pooling** -> we are learning the best sampling algorithm

Flattening

DNN takes 1D vectors as input but at this point we have 2D matrices, so we need to transform them to 1D. We start with the first row of the first pooled map as the beginning of the 1D vector. We then take the 2nd row and append it to the end, and then the 3rd row, and so forth. We then proceed to the second pooled map and do the same process, continuously appending each row, until we’ve completed the last pooled map.

Basic sequential CNN is Conv2D

In Conv2D, 2D means that the input to the convolution layer will be matrix (2D array). 2D convolutions are the common practice in computer vision.