* Deep learning is versatile.
* Calorific rays are beyond red light.
* 780nm-3000nm is near infrared.
* 3000nm-6000nm is mid infrared.
* 0.6 um – 300um is far infrared.
* Label encoding gives labels. However, one hot encoding gives matrix which gives 1 only for one column.
* So as to not compare different types, you need standardization or normalization.
* Our algorithm memorize everything in the training set. The distinction between validation and training is, algorithm is able to see the labels of training set but for the validation set, it does not have direct access to the labels. The validation labels are used to check the accuracy (how well our algorithm is doing)
* We have some input images that input image will go through some deep convolution network that network will give us some feature vector that has different dimensions from that final feature vector we will have some fully connected layer that gives us 2 numbers for the different class scores.
* With a bunch of convolutional layers to make predictions for pixels all at once.
* Convolutional layers with huge number of channels such as 64,128 and 256, applying bunch of convolutions on this high resolution input image over a sequence of layers would be extremely computationally expensive.
* So rather than doing all the convolutions of the full spatial resolution of the image using small number of convolutional layers at the original resolution than downsample that feature map using maxpooling or strided convolutions.
* To reduce spatial size, we can use pooling or strided convolution mechanisms.
* Upsampling (skip connection) helps us handle fine details better and helps us preserve some of lost spatial information during pooling.
* The width and height dimensions tend to shrink as you go deeper in the network.
* The number of channels is controlled by the first argument passed to the conv2D.
* (3,3,64) = 64 kanallı ve 3x3 lük tensör demektir.
* Flatten ile 192x3 =576 lık olur. Dense layer onu çıkışta beklenen sınıf sayısına çeker.
* Flatten, Dense with Relu, dense with softmax
* Dense layer learns global (patterns involving all patterns) patterns but convolution layers learn local patterns (small 2d windows)
* A first convolution layer will learn small local patterns second convolution layer will learn larger patterns made of the features of the first layers and so on. This allows convnets to efficiently learn increasingly complex and abstract visual concept.
* Convolutions operate over 3D tensors called feature maps, height,width, channel.
* Height,width = (two spatial)
* Channel = (depth)
* 28x28x1 => 26x26x32 It means that 32 different response maps. 32 different response of the filter pattern at different locations.
* 26x26x32 = feature map
* 26x26x1 = each one is feature.
* The patches extracte from the input are 3x3 or 5x5
* 32 is the depth of output feature map 26x26x32
* By sliding windows of 3x3 or 5x5 over 3D input images, we transform 3D patches into 1D vectors and reassemble them into 3D output feature map corresponds to the same location in the input feature map.
* Output width abd height may differ from the input width and height since we use border effects and strides.
* Border effects can be countered by padding the input feature map
* For 5x5 you add 2 pixels and for 3x3 you add 1 pixel for each directions
* It is possible strided convolutions. Strided convolution means that convolution with a stride higher than 1.
* Using stride 2 means the width and height of the feature map are downsampled by a factor 2.
* Strided convolutions are rarely used
* Instead of strided convolutions we use max pooling.
* The role of max pooling is aggresively downsampling feature maps
* Max pooling is like convolution however instead of linear transformations like convolving it assigns / finds the max pattern via hardcoded tensor operation
* Max pooling uses 2x2 windows convolution uses 3x3
* Max pooling uses strides 2 and convolution uses stride 1
* Why do not we keep feature maps all the way up? 1) it is not conducive to learning a spatial hierarchy of features 2) to reduce parameter numbers
* We use increasngly large layers to induce/provide spatial filter hiearchy
* You can use strided convolution in the prior convolution layers
* Data augmentation is powerful technique for mitigating overfitting in computer vision
* Deep learniing can find interesting features in the training data on its own in computer vision.
* Since convolution networks learn local, translation-invariant features they are used in.
* Train-test-validation = 1000-500-500
* This is balanced binary classification problem which means classification accuracy will be appropriate measure of success.
* Deep learning are black boxes but we can visualize filters, models, intermediate outputs and class activations.
* Smoothing can make us get rid of noisy accuracy/loss graphs.
* The activations of higher layers carry less information about the specific input being seen.
* Densely connected networks are commonly used for categorical data
* They are also used for final decisiob/classification/regression stage
* To reduce the size of feature maps we use more convolution layers and max poolings.
* This serves to further reduce the size of the feature maps
* Since we deal with binary classification problem, we use dense layer of size 1 and sigmoid activation
* RMSProp(lr=0.0001) is good for the compilation step
* Binary classification-sigmoid-binary crossentropy
* Multiclass single label-softmax-categorical crossentropy
* Multiclass multi label -sigmoid-binary crossentropy
* Regression-sigmoid-mse or binary crossentropy
* Before fed into the network, data should be formatted into appropriately preprocessed floating tensors.
* Dropout and wieght decay (L2 regularization) can decrease/mitigate overfitting
* Overfitting is caused by having too few samples to learn from, rendering you unable to train a model that generalize to new data
* With infinite data, your model would be exposed to every possible aspect of the data distribution and you would never overfit
* After flattening you can use dropout to decrease/mitigate overfitting
* Pretrained network is a saved network that was previously trained on a large dataset whose spatial hierarchy of features learned very well
* For problems where object location matters, densely connected features are largely useless
* By frozen we used fine tuning
* Earlier layers in the convolutional base encode more generic resuable features whereas layers higher up encode more specialized features
* It is useful to fine tune the more specialized features
* The more parameters you are training the more you are at risk of overfitting
* By using low learning rate, you avoid/limit the magnitude of the modifications you make to the representations of layers. Large updates may learn these representations
* Sequential model assumes networks has one input and one output
* Sequential model consists of a linear stack of layers
* If 2 attributes arenot statistically independent you could build a better model by learning to jointly predict instead of training 2 seperate models
* İf sequential model is used you use add to add
* İf functional api is used, you use outputs as input to the next layer
* Convolution 1x1 is used in inception model
* Although it seems useless 1x1 is used in inception model
* İnception network includes independent parallel branches to be connected or concatenated after a while.
* Residual network uses early layers as input to upcoming layers. It is a kind of shortcut
* Residual network uses adding instead of concatenation
* Callback monitors validation loss and validation accuracy so while calling model.fit, you need validation data.
* You can reduce learning rate by a factor when validation loss stops improbing
* Reucelronplateu is effective to get rid of local minima problems during training
* Batch normalization adaptively normalize data even mean and variance changes over time
* Normalization helps us to generalize well to new data
* Normalization centers the data on 0 by subtracting the mean from data and giving data a unit standard deviation by dividing data by its standard deviation
* İf you have small model with limited data, you can use seperable convolution which uses less data parameters with same accuracy
* While hyperparameter optimization is done, initial decisions are suboptimal even if you have good intiution
* For hyperparameter choices, randomness is the best choive despite being the most naive one.
* Model ensembling means equal classifiers can be averaged to find final prediction
* Elephant hose, elephant tail elephant leg come together and we pool them and find the object as elephant
* When building high performance deep conv nets, you need batch normalization, seperable convolutions and residual connections.