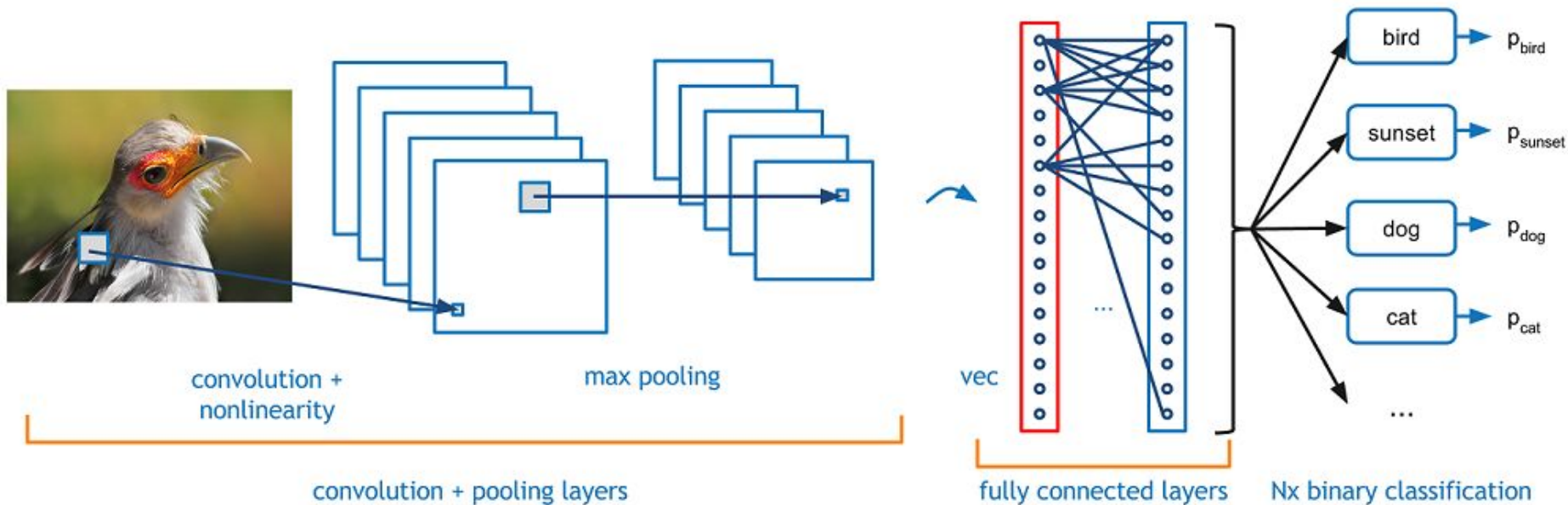


Convolutional Neural Networks



Lots of Break through to Computer Vision Industry

Convolution → Image processing of Data

Understanding pattern from unstructured data

Ex: image

Yann Lecun

Lenet arc base for conv which takes about Image processing

How does a Machine look at an image?

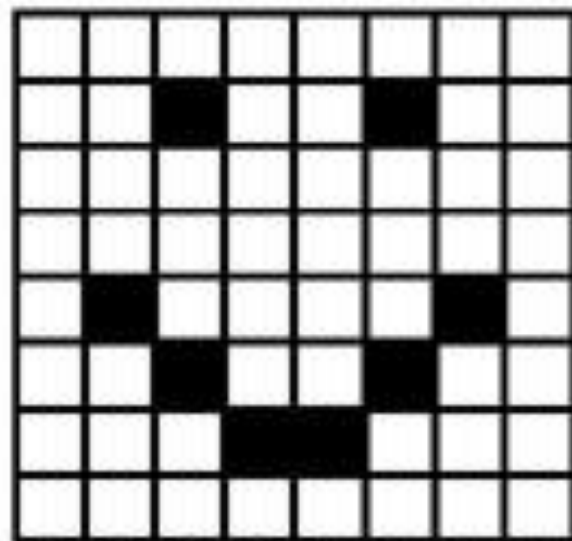
The machine will basically break this image into a matrix of Pixels and store the color code for each pixel at the representative location

Image ---> array of data points

Pixels values ranges from 1-256(0-255)

1-white(low)


256-dark(high)



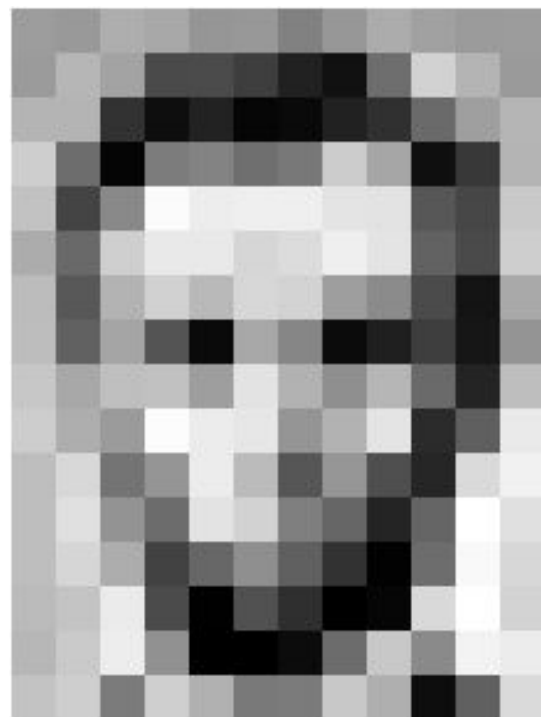
Picture

0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0

Bitmap

 = 0

 = 1



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	105	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Convolutional Neural Networks

Convolutional : It is the process of extracting features from image

Neural Networks: Building a architecture which is useful for feature extraction from image

Convolutional Neural Networks

Front end

Convolution

It tries to understand the
features from data

Back end

ANN(DNN)

Classify the activity

Convolutional Neural Networks

To extract the features from input

Filters : It is used to extract the information also known as kernel

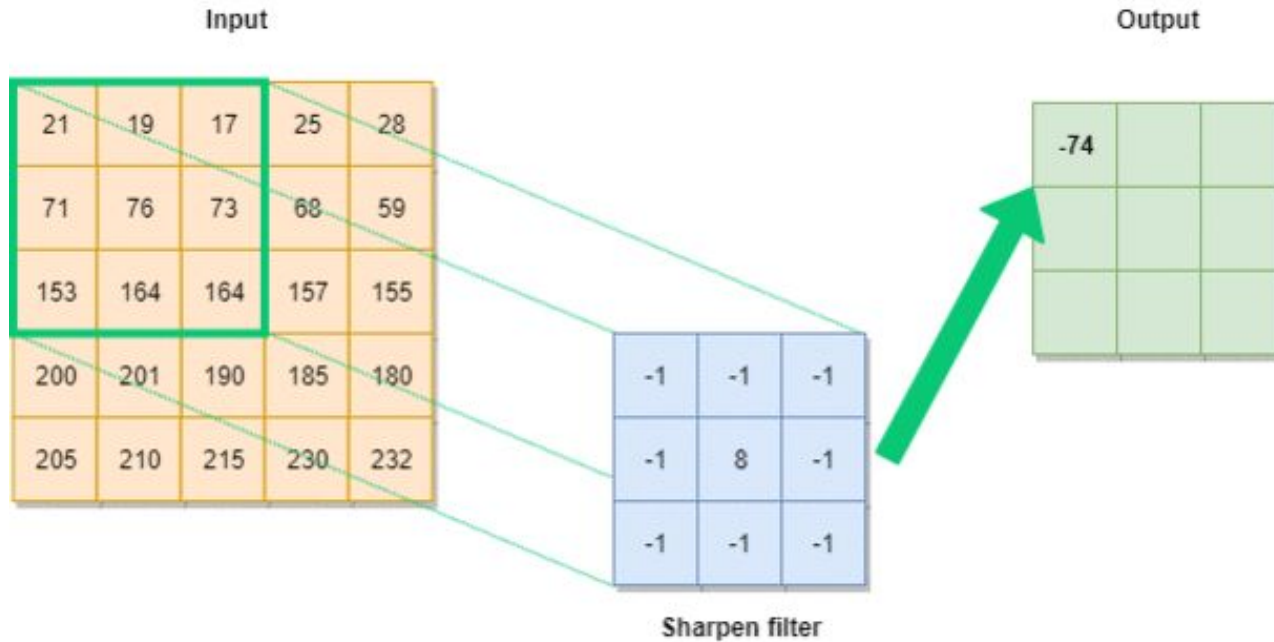
Ex : odd filters(3×3 , 5×5)

Stride: How fast your performing/reading the information. How fast system can scan

Padding: When you do Convolutional operation image get shrinks to avoid this we use padding option. Retain the originality of the image

Convolutional Neural Networks

Filter

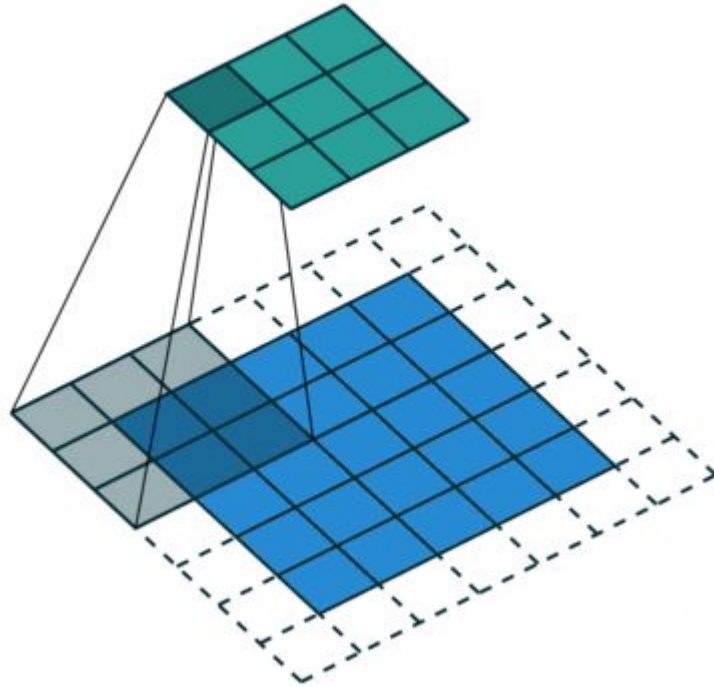


Kernel

- **Kernel** is nothing but a filter that is used to extract the features from the images.
- The **kernel** is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.
- **Kernel** moves on the input data by the stride value

Convolutional Neural Networks

Stride : 2



Stride

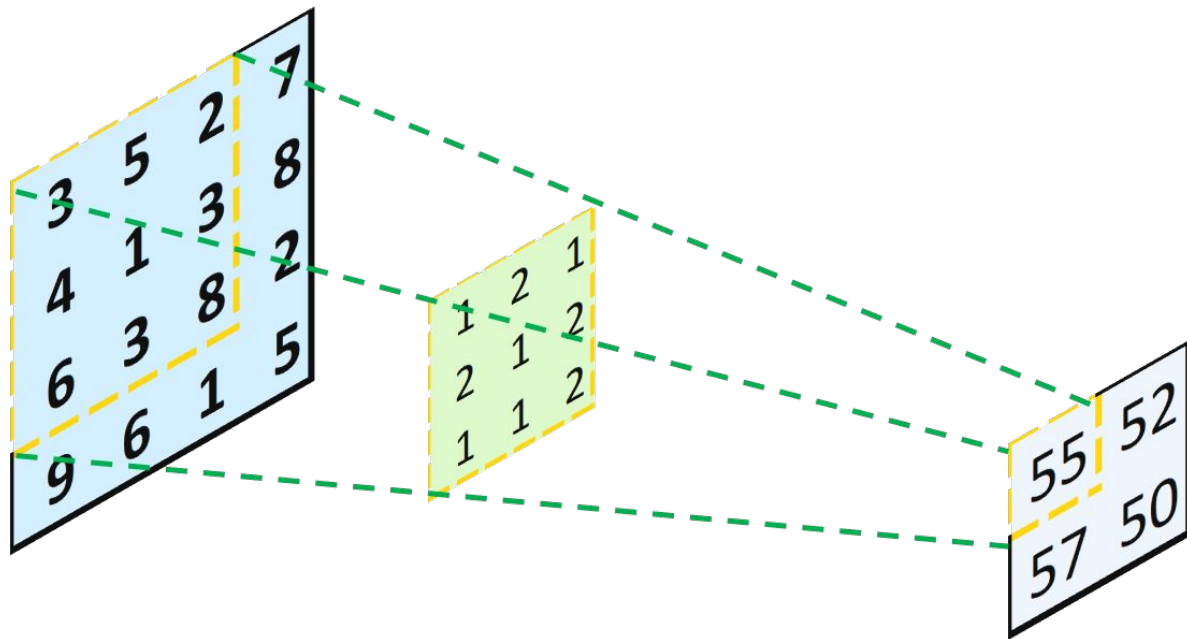
- The filter is moved across the image left to right, top to bottom, with a one-pixel column change on the horizontal movements, then a one-pixel row change on the vertical movements.
- The amount of movement between applications of the filter to the input image is referred to as the stride, and it is almost always symmetrical in height and width dimensions.
- The default stride or strides in two dimensions is (1,1) for the height and the width movement.

Convolutional Neural Networks

Stride : 1

5	5	2	5	5	
5	5	2	5	5	
7	7	5	7	7	
5	5	2	5	5	
5	5	2	5	5	
					stride=1

Convolutional Neural Networks



$$[n-f+1]*[n-f+1] = [4-3+1]*[4-3+1] \\ = 2*2(\text{valid})$$

$$[n+2p-f+1]*[n+2p-f+1] = \\ [4+2(1)-3+1]*[4+2(1)-3+1] \\ = 4*4(\text{same})$$

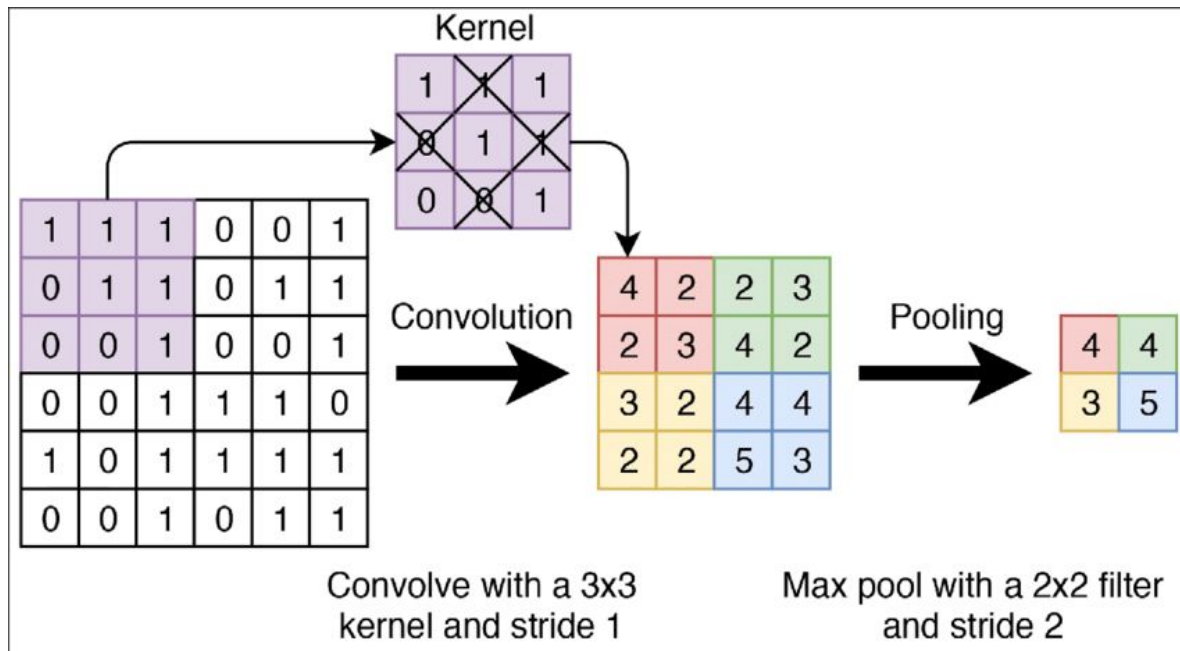
$$P = (f-1)/2$$

Padding

- Padding is the best approach, where the number of pixels needed for the convolutional kernel to process the edge pixels are added onto the outside copying the pixels from the edge of the image.
- Fix the Border Effect Problem With Padding.

Convolutional Neural Networks

Max pooling

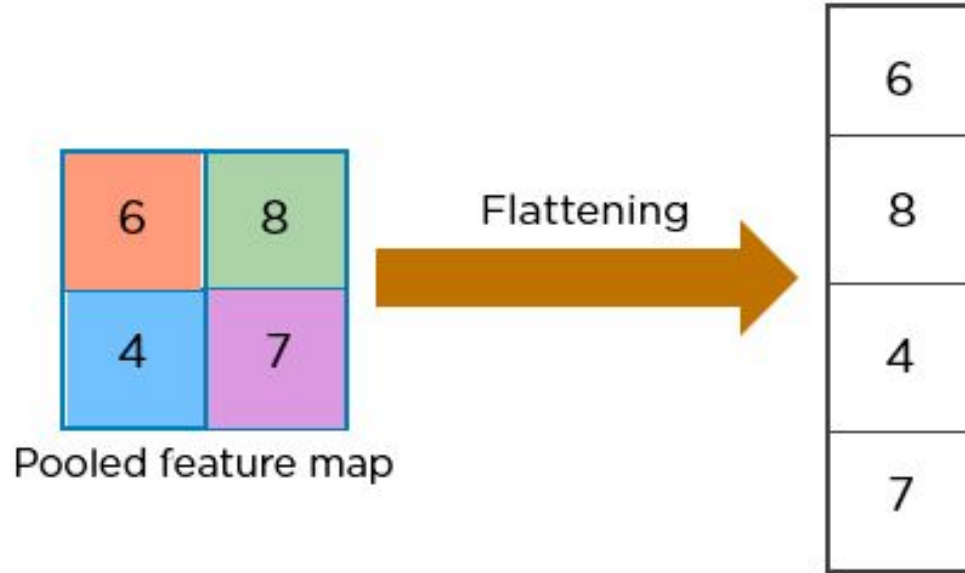


Pooling

- Pooling is required to down sample the detection of features in feature maps.
- Pooling layers provide an approach to down sampling feature maps by summarizing the presence of features in patches of the feature map. Two common pooling methods are average pooling and max pooling that summarize the average presence of a feature and the most activated presence of a feature respectively.

Convolutional Neural Networks

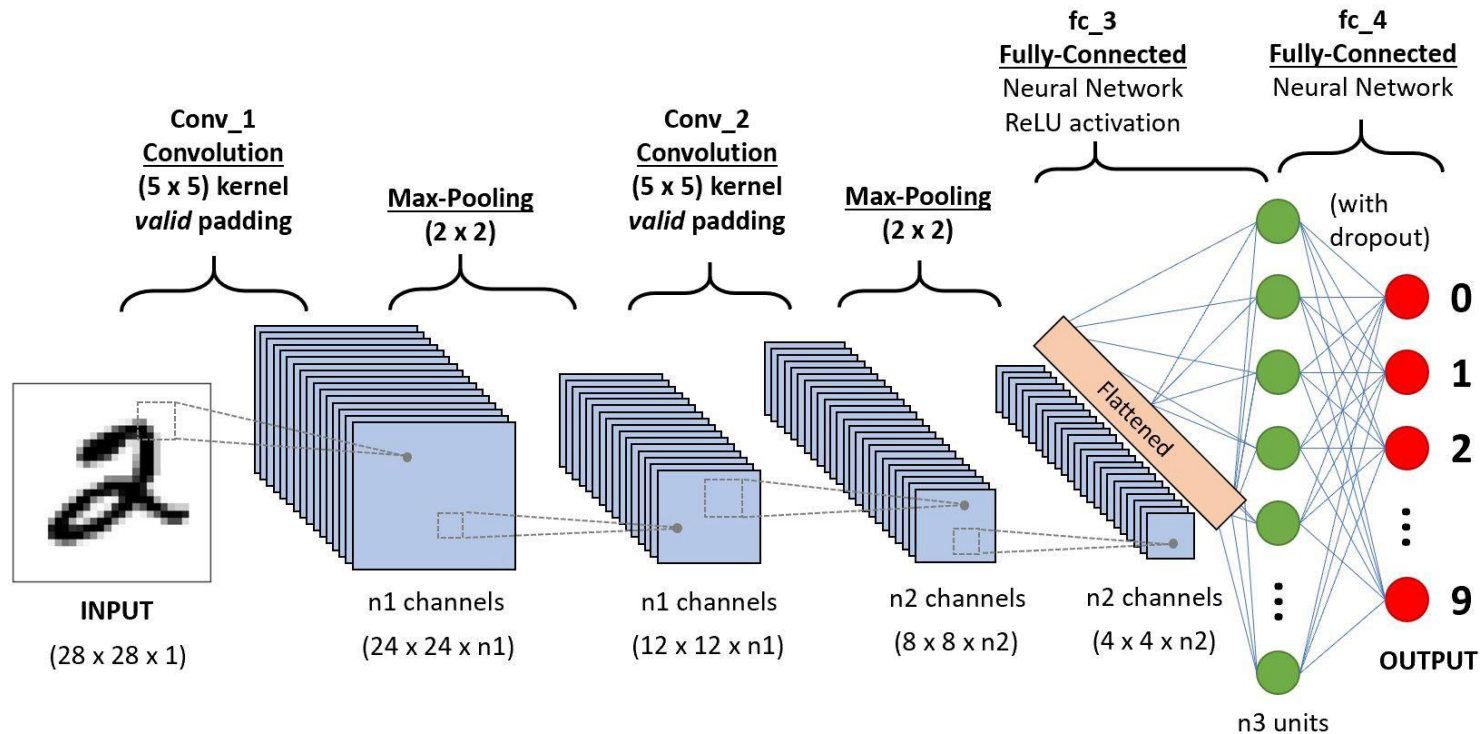
Flattening



Flattening

- Once the pooled featured map is obtained, the next step is to flatten it.
- It involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

Convolutional Neural Networks



Convolutional Neural Networks

CNN - Basic Sequential Model

Basic Sequential CNN - Conv2D

We will design a basic sequential model in two parts, the convolutional frontend and the DNN backend. We start by adding a convolutional layer of 16 filters as the first layer using the `Conv2D` class object.

The size of each filter will be a 3x3, which is specified by the parameter `kernel_size` and a stride of 2 by the parameter `strides`. Note that for strides a tuple of (2, 2) is specified instead of a single value 2. The first digit is the horizontal stride (across) and the second digit is the vertical stride (down). It's a common convention for stride that the horizontal and vertical are the same; **therefore one commonly says a "stride of 2" instead of "a 2x2 stride"**.

We pick an input size for our images. We like to reduce the size to as small as possible without losing detection of the features. In this case, we choose 128 x 128 (grayscale). The `Conv2D` class always requires specifying the number of channels, **instead of defaulting to one for grayscale; we specified it as (128, 128, 1) instead of (128, 128)**.

```
from keras.models import Sequential, Conv2D, MaxPooling2D, Flatten
from keras.layers import Dense, ReLU, Activation

model = Sequential()
# Create a convolutional layer with 16 3x3 filters and stride of two as the input layer
model.add(Conv2D(16, kernel_size=(3, 3), strides=(2, 2), padding="same", input_shape=(128,128,1)))
```

Convolutional Neural Networks

CNN - Basic Sequential Model

Basic Sequential CNN - MaxPooling2D - Flatten

The output from the convolution layer is passed through a rectified linear unit activation, which is then passed to the max pooling layer, using the `MaxPool2D` class object. The size of the pooling region will be 2x2, specified by the parameter `pool_size`, with a stride of 2 by the parameter `strides`.

The pooling layer will reduce the feature maps by 75% into pooled feature maps. The pooled feature maps are then flattened, using the `Flatten` class object, into a 1D vector for input into the DNN. We will glance over the parameter `padding`. In almost all cases, you will use the value `same`; it's just that the default is `valid` and therefore you need to explicitly add it.

```
# Pass the output (feature maps) from the input layer (convolution) through a rectified linear unit activation
# function.
model.add(ReLU())
# Add a pooling layer to max pool (downsample) the feature maps into smaller pooled feature maps
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
# Add a flattening layer to flatten the pooled feature maps to a 1D input vector for the DNN classifier
model.add(Flatten())
```

Convolutional Neural Networks

CNN - Basic Sequential Model

Basic Sequential CNN - Attaching the DNN Classifier

Finally, we attach the DNN classifier. In this example, we assume the classifier will be trained for recognizing 26 handwritten lowercase (or hand-signed) characters of the English alphabet.

```
# Add the input layer for the DNN, which is connected to the flattening layer of the convolutional frontend
model.add(Dense(512))
model.add(ReLU())
# Add the output layer for classifying the 26 hand signed letters
model.add(Dense(26))
model.add(Activation('softmax'))
# Use the Categorical Cross Entropy loss function for a Multi-Class Classifier.
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

CONTENTS:

- Hyper Parameter Tuning
- Regularization
- Optimization

SETTING UP YOUR MACHINE LEARNING APPLICATION

- Basic Prerequisites:
- When training a neural network you have to make a lot of decisions
- How many layers will your neural network have?
- How many hidden units do you want each layer to have?
- What's the learning rates?
- What are the activation functions you want to use for the different layers?

TRAIN/DEV/TEST SETS

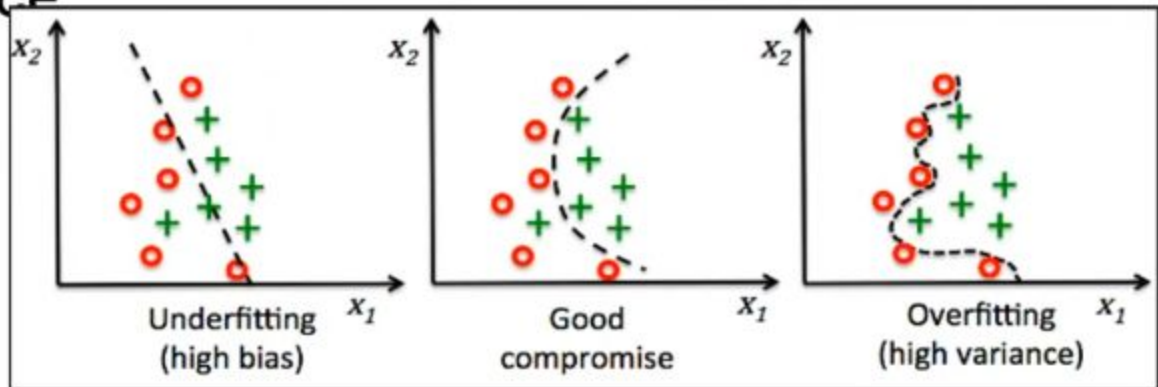
- The data you have and carve off some portion of it to be your training set.
- Some portion of it to be your hold-out cross validation set, and this is sometimes also called the development set. And for brevity I'm just going to call this the dev set, but all of these terms mean roughly the same thing.
- you might carve out some final portion of it to be your test set



- it was common practice to take all your data and split it according to maybe a 70/30% 70/30 train test splits.
- If you don't have an explicit dev set or maybe a 60/20/20% split
60% train, 20% dev and 20% test.
considered best practice in machine learning.
- The rule of thumb I'd encourage you to follow in this case is to make sure that the dev and test sets come from the same distribution.

BIAS/VARIANCE

- Bias and Variance is one of those concepts that's easily learned but difficult to master. Even if you think you've seen the basic concepts of Bias and Variance



HIGH BIAS AND HIGH VARIANCE

- High bias, what we say that this is underfitting the data.
- if you fit an incredibly complex classifier, maybe a deep neural network, or neural network with all the hidden units, maybe you can fit the data perfectly, but that doesn't look like a great fit either. So there's a classifier of high variance and this is overfitting.

UNDERSTAND HIGH VARIANCE

- Understand bias and variance will be the train set error and the dev set or the development set error.
- Train set error : 1%
- Dev set error : 11%
- Doing very well on the training set, but you're doing relatively poorly on the development set.
- This looks like you might have overfit the training set leads High variance

UNDERSTAND HIGH BIAS.

- Let's say,
- training set error is 15%.
- Dev set error is 16%.
- In this case, assuming that humans achieve roughly 0% error, that humans can look at these pictures and just tell if it's cat or not, then it looks like the algorithm is not even doing very well on the training set.
- So if it's not even fitting the training data seem that well, then this is underfitting the data so this algorithm has high bias

HIGH BIAS AND HIGH VARIANCE

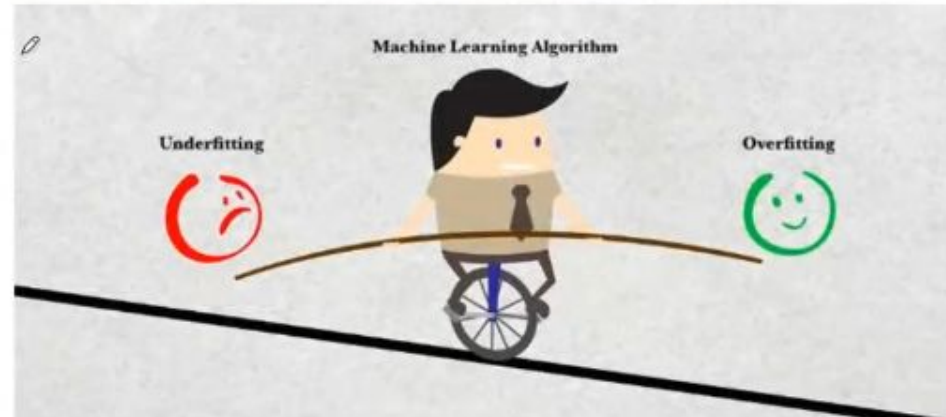
- Understand High bias and High variance will be the train set error and the dev set or the development set error.
- Train set error : 15%
- Dev set error : 30%
- In this case, this algorithm as having high bias, because it's not doing that well on the training set, and high variance.

BASIC RECIPE:

- if it does have high bias, does not even fit in the training set that well,
- some things you could try would be to try pick a network, such as more hidden layers or more hidden units, or you could train it longer.
- Maybe run trains longer or try some more advanced optimization algorithms.
- Whereas getting a bigger network almost always helps and training longer doesn't always help, but it certainly never hurts*****

LOW BIAS AND LOW VARIANCE

- if you have 0.5% training set error, and 1% dev set error, then maybe our users are quite happy, that you have a cat classifier with only 1% error, than just we have low bias and low variance.



- if you have high variance, well, best way to solve a high variance problem .

“To get more data but sometimes you can't get more data Or we can try regularization”

if you actually have a high bias problem, getting more training data is actually not going to help. Or at least it's not the most efficient thing to do.

Bigger Network: Reduce Bias

More Data : Reduce Variance

REGULARIZATION

- Overfitting \rightarrow “High Variance”
- The first things you should try is regularization.
- The other way to address high variance, is to get more training data that's also quite reliable.
- But you can't always get more training data, or it could be expensive to get more data. But adding regularization will often help to prevent overfitting, or to reduce the errors in your network.

- But adding regularization will often help to prevent overfitting, or to reduce the errors in your network.
- So let's see how regularization works. Let's develop these ideas using logistic regression.

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$
$$\min_{\theta} J(\theta)$$

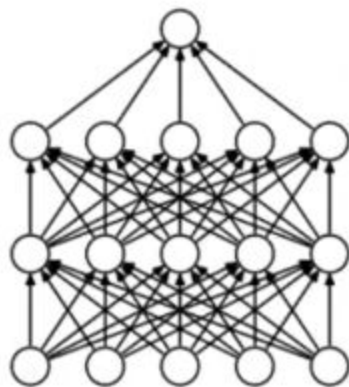
ISSUES OF REGULARIZATIONS

- L2 regularization is the most common type of regularization.
- You might have also heard of some people talk about L1 regularization.
- If you use L1 regularization, then w will end up being sparse. And what that means is that the w vector will have a lot of zeros in it.
- L1 regularization to make your model sparse, helps only a little bit

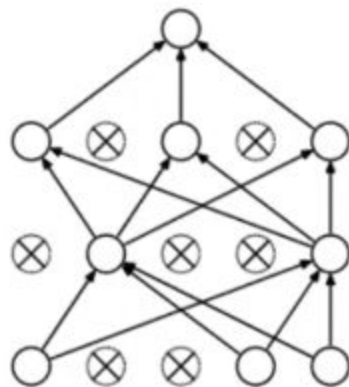
WHY REGULARIZATION REDUCES OVERFITTING?

- L2 Regularization:
- Set the weight matrices W to be reasonably close to zero.
- And if that's the case, then this much simplified neural network becomes a much smaller neural network.
- Another trial we can do by using the Tanh.

DROPOUT REGULARIZATION:



(a) Standard Neural Net



(b) After applying dropout.

very powerful regularization
techniques is called "dropout."

HOW DROPOUT WORKS:

- It prevents overfitting and provides a way of approximately combining exponentially many different neural network architectures efficiently.
- The term “dropout” refers to dropping out units (hidden and visible) in a neural network.
- By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in Figure 1.
- The choice of which units to drop is random.

DROPOUT

- the effect of implementing drop out is that it shrinks the weights and does some of those outer regularization that helps prevent over-fitting.
- To summarize, it is possible to show that
- drop out has a similar effect to L2 regularization.

FINAL WORD ON DROPOUT

- Many of the first successful implementations of drop outs were to computer vision. So in computer vision, the input size is so big, inputting all these pixels that you almost never have enough data.
- And so dropout is very frequently used by computer vision.
- Drop out is a regularization technique, it helps prevent over-fitting

DATA AUGMENTATION

- Let's take a look. Let's say you fitting a cat classifier, If you are over fitting getting more training data can help, but getting more training data can be expensive and sometimes you just can't get more data.
- But what you can do is augment your training set



- Minor changes such as flips or translations or rotations. Our neural network would think these are distinct images anyway.
- This can be an inexpensive way to give your algorithm more data and therefore sort of regularize it and reduce over fitting.
- ImageDataGenerator keras provides



EARLY STOPPING

- Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set.
- When we see that the performance on the **validation set** is getting worse, we immediately stop the training on the model. This is known as **early stop**.



EARLY STOPPING

- Training neural networks is challenging.
- When training a large network, there will be a point during training when the model will stop generalizing and start learning the statistical noise in the training dataset.
- This overfitting of the training dataset will result in an increase in generalization error, making the model less useful at making predictions on new data

BATCH NORMALIZATION

- Batch normalization is a technique designed to automatically standardize the inputs to a layer in a deep learning neural network.
- The layer will transform inputs so that they are standardized, meaning that they will have a mean of zero and a standard deviation of one.
- During training, the layer will keep track of statistics for each input variable and use them to standardize the data.

BATCH NORM AS A SOLUTION.

- Batch normalization mitigates the effects of a varied layer inputs. By normalizing the output of neurons, the activation function will only receive inputs close to zero. This ensures a non-vanishing gradient.
-

BENEFITS OF BATCH NORMALIZATION

- Helps prevent vanishing gradient in networks with saturable nonlinearities (sigmoid, tanh, etc)
- Regularizes the model
- Allows for Higher Learning Rates



Thank You