Week 4 - Neural Networks and Deep Learning for Image Classification

4.1 Video: Neural Networks

4.2 Lab: Simple Neural Network for XOR

4.3 Video: Fully Connected Neural Network Architecture

4.4 Lab: Neural Network Rectified Linear Unit (ReLU) vs Sigmoid

4.5 Lab: Training A Neural Network with Momentum

4.6 Video: Convolutional Networks

4.7 Lab: Convolutional Neural Network

4.8 Lab: Data Augmentation

4.9 Video: CNN Architectures

4.10 Reading: Deploying a Model

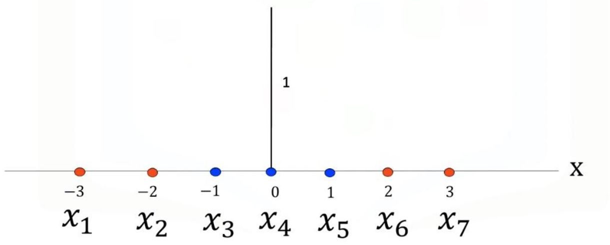
4.11 Lab: Use CNN for "Hotdog, Not Hotdog" Classifier and Deploy Model with CV Studio

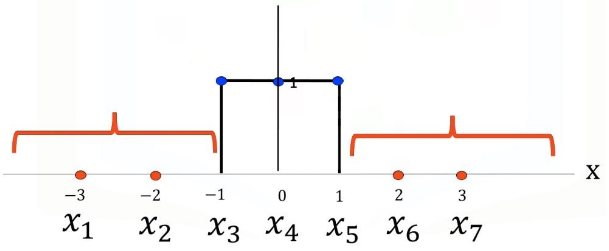
总结：

4.1 Neural Networks

Features and targets: Example

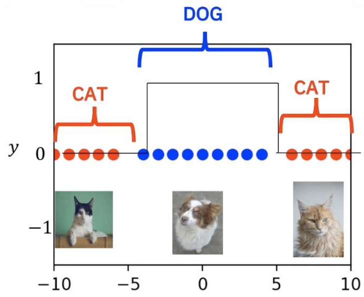
- It is helpful to view the sample y as a decision function of x

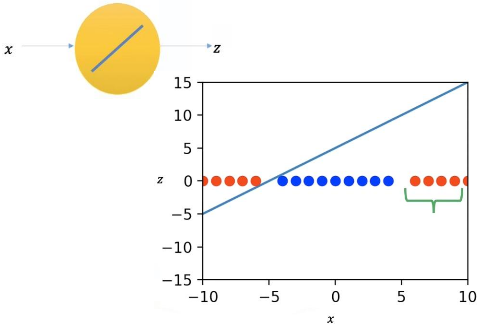


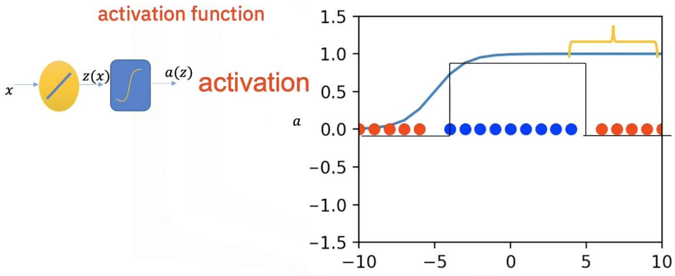


In this video we will discuss Neural Networks. Consider the following non linearly separable dataset. We will use one dimension for simplicity. Let's look at an example of classification where we overlaid the color over the feature. in the context of neural networks its helpful to think of the classification problem as a decision function, just like a function when y equals one the value is mapped to one on the vertical axis. We can represent the function like a box , this box function is an example of a decision function, any values of x in the following region is one, any value of x in this region is mapped to zero.
A neural network will approximate this function using learnable parameters.

Linear Classifiers vs Neural Networks

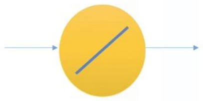


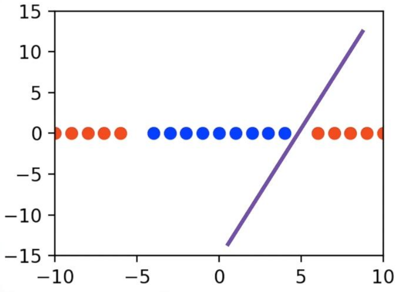


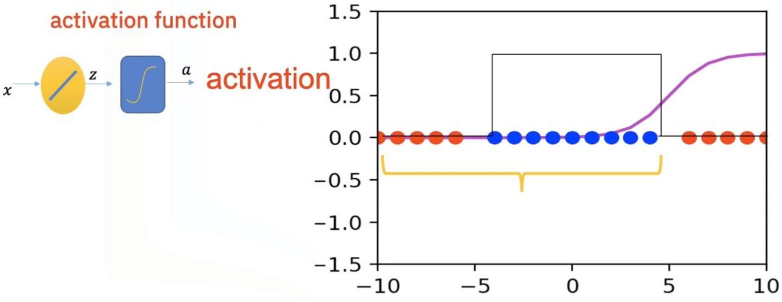


We can also view the problem as trying to approximate the box function using logistic regression, anything in this region y will be a one i.e dog, anything in this region y will be a 0 i.e cat. If this was our cat dog dataset.
In this example we cannot use a straight line to separate the data. This line can be used to linearly separate some of the data, but some of the data is on the wrong side of the line.
we can use the following node to represent the line and the edges to represent the input x and output z. If we apply the logistic function, in the context of neural networks this is called the activation function. These values of the function are incorrect and we get an incorrect result in this region. We can represent the sigmoid function with the following node taking the input z from the linear function and producing an output, technically “A” is a function of z and x. We will call the function “A” the activation function and the output of of “A” is called the activation.

Neural Networks

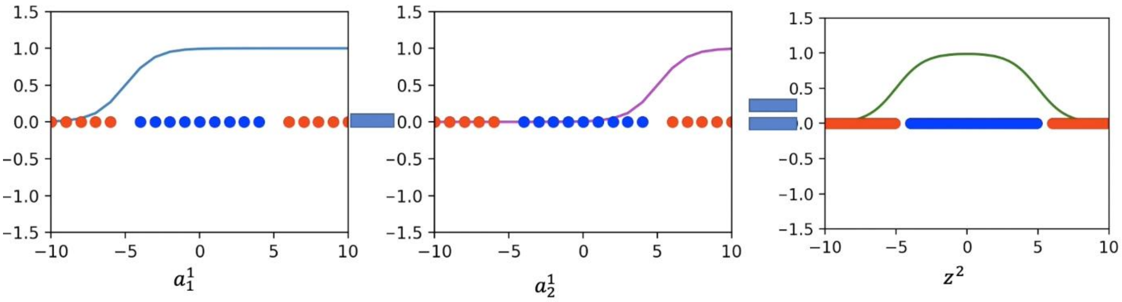


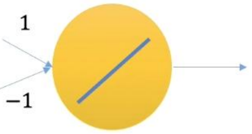




This line can also be used to linearly separate some of the data but some of the data is on the wrong side of the line. This line looks like it can be use to separate the data but lets see what happens when we apply the sigmoid function.
After applying the sigmoid or activation function, we get an incorrect result for some of the samples.

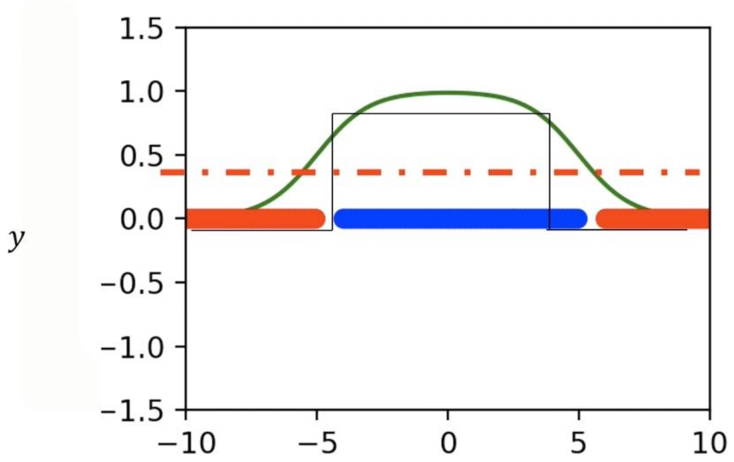
总结:





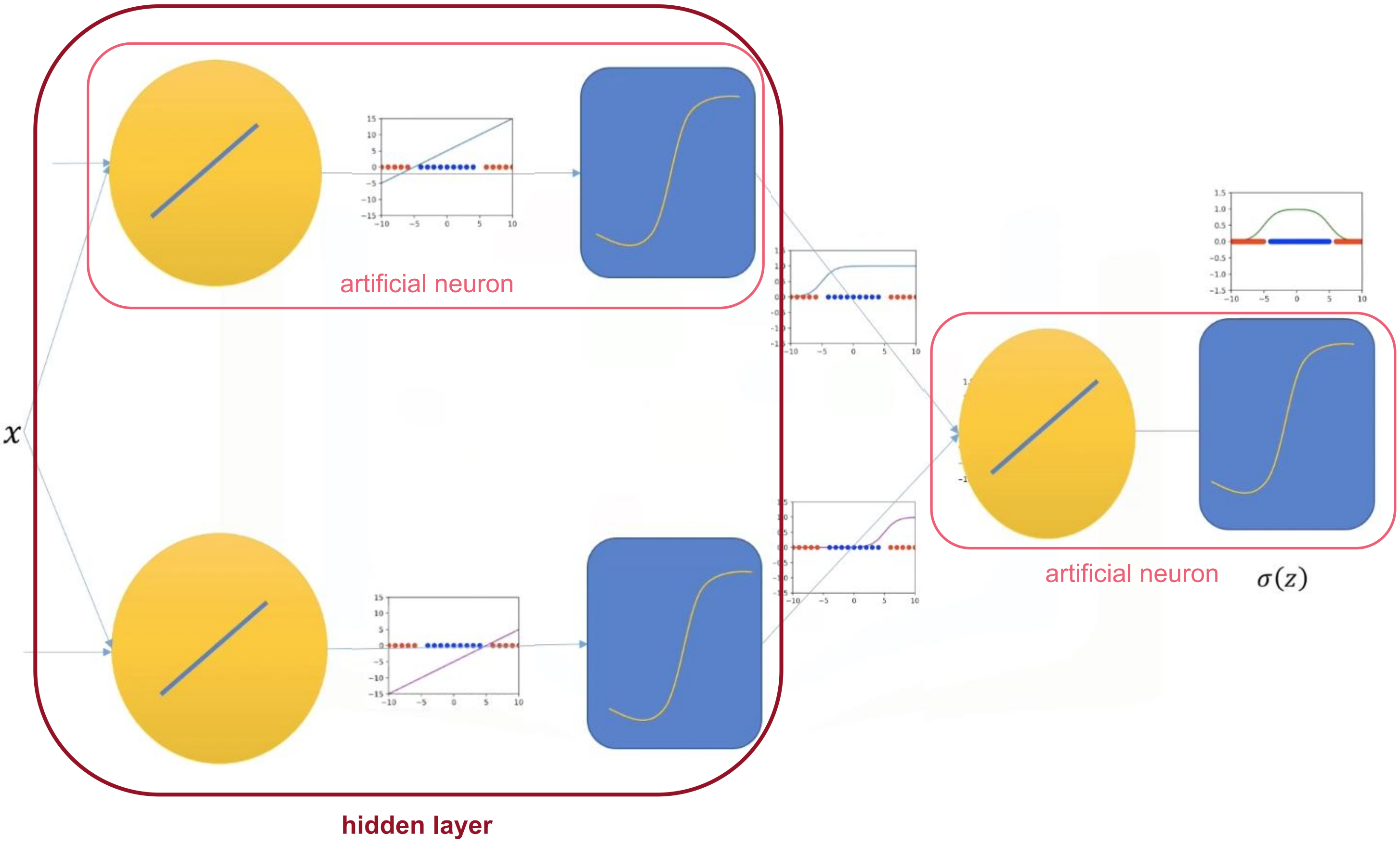
Consider the following sigmoid functions we call them "A sub script one” and “A sub script two”. If we subtract the second sigmoid function from the first sigmoid function we get something similar to the decision function. We can also apply the following operations with a linear function i.e just subtract the second activations from the first activation function.

These values will be learnable parameters.



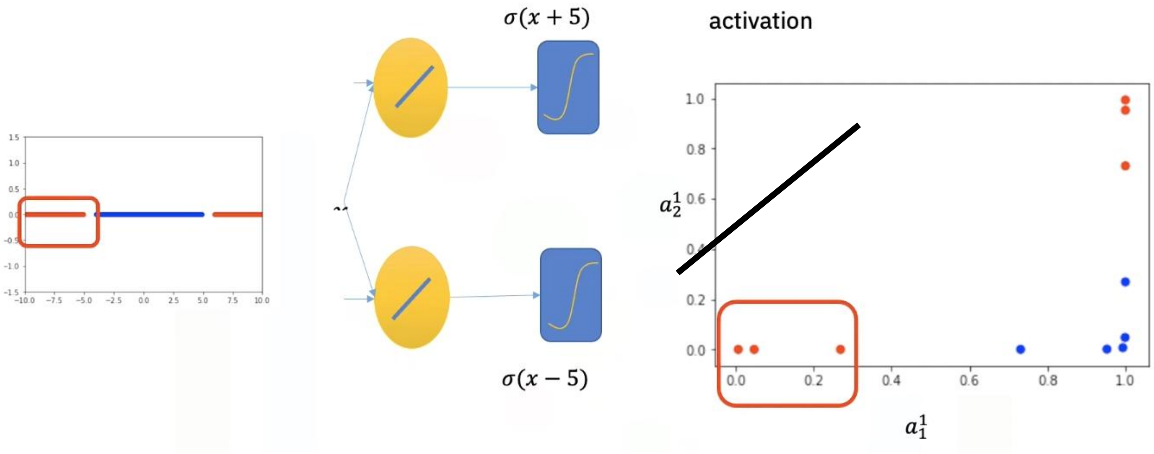
If we apply a threshold setting every value less than 0.5 to zero and grater than 0.5 to one, we get the exact function we are trying to approximate.

We can now classify the data, we obtain the parameters via gradient descent

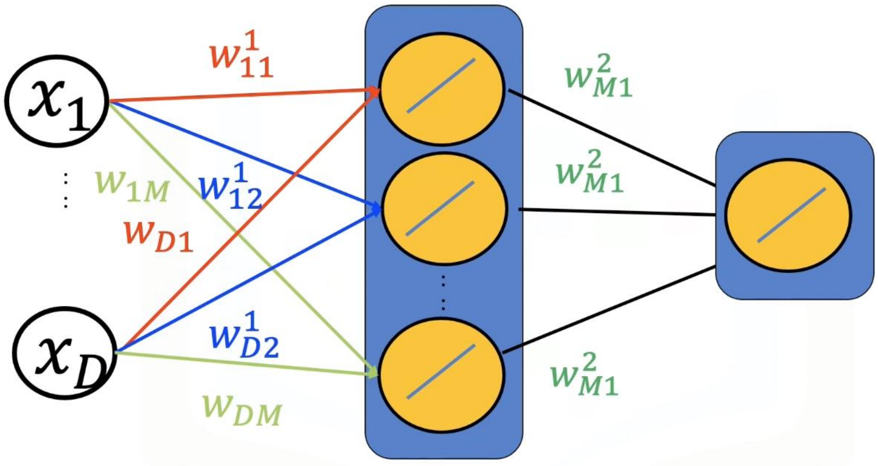


总结：

We can use the graph to represent the process. We apply two linear functions to x and we get two outputs to each linear function we apply a sigmoid we then a apply a second linear function to the outputs of the sigmoid, we usually apply another function to the output of this linear function then apply a threshold.
This diagram is used to represent a two-layer neural network. We have the hidden layer. Each linear function and activation is known as an artificial neuron, in this case the hidden layer has two artificial neurons, the output layer has one artificial neuron, as it has two inputs the input dimension for this neuron is two.



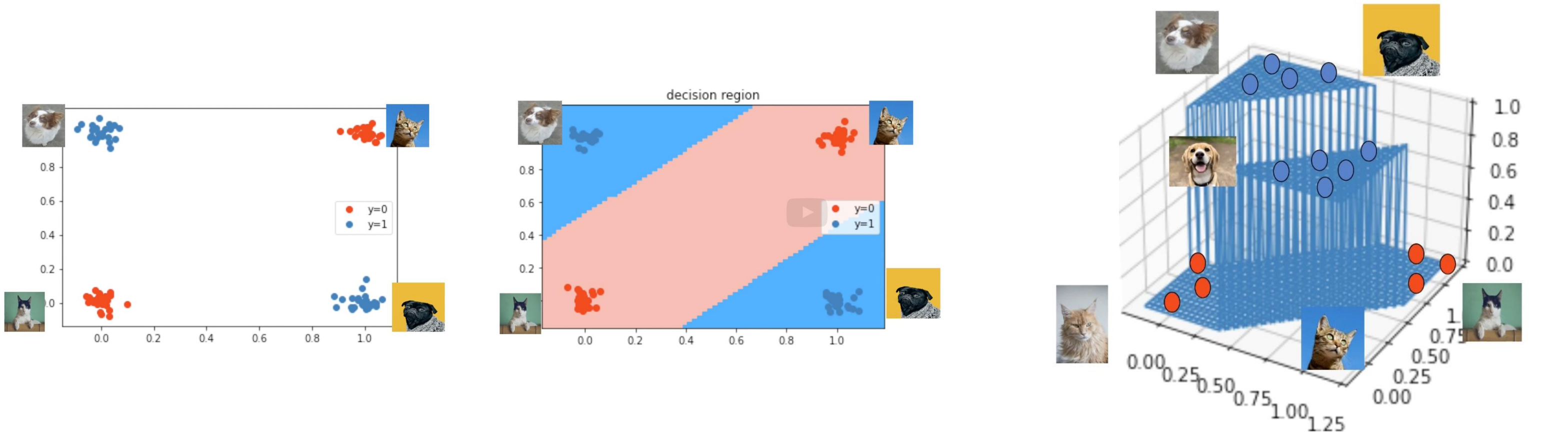
Its helpful to look at the output components of the activation, the outputs of the activation function is a 2D plane that looks like this. These red data points get mapped to these points in the 2D plane, these blue data points get mapped to these points in the 2D plane and so on.
It turns out that we can split the point using the following plane this is what the linear function on the second layer does.



In the same way we can add more dimensions to the input, notice that there are a lot more weights between the input layer and hidden layer, we will leave out the bias terms.

We see a neural networks had a lot of learnable parameters, for example a logistic regression model may have hundred's of learnable parameters a modern neural network will have millions.

Generally these type of Neural Networks are called Feedforward Neural Networks or fully connected networks, but we will usually refer to them as Neural Networks.



We can use neural networks to classify multiple dimensions. Here we have a non-linearly separable dataset in two dimensions. Here we have a neural network with 2 dimensions. The more dimensions the more Neurons we require.

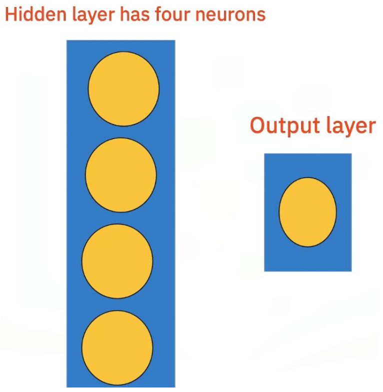
We can plot the decision function in 2 dimensions, the horizontal axis is the value yhat, we have cats that are mapped to zero, we have dogs that are mapped to one.
Check out the labs for more.

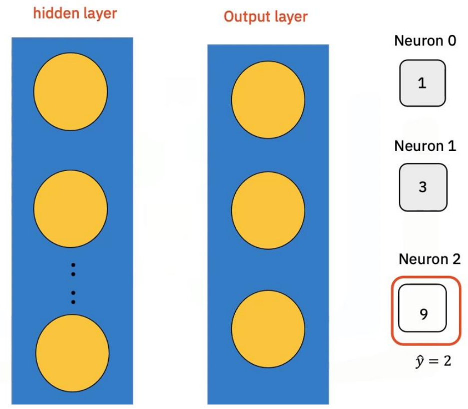
4.2 Lab: Simple Neural Network for XOR

详见 4.2_Simple Neural Network for XOR.ipynb

总结:

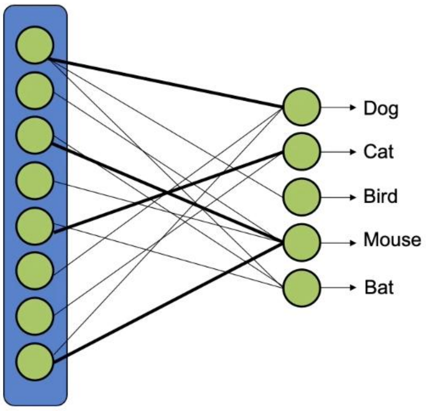
4.3 Fully Connected Neural Network Architecture

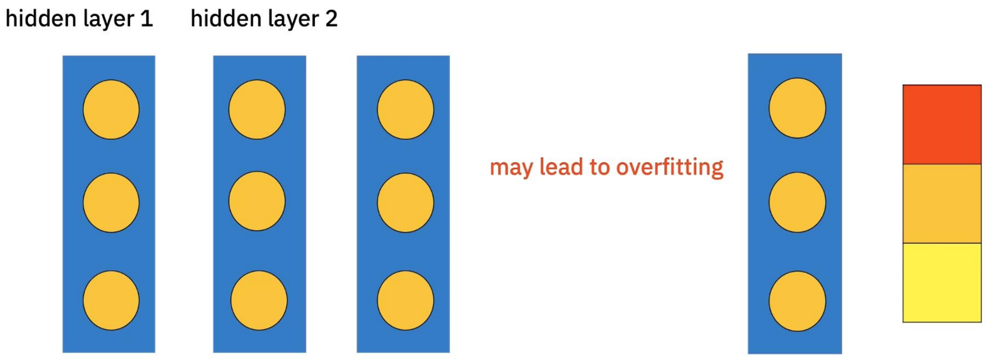




In this video we will cover **Fully Connected Neural Network Architecture**, This deals with how to arrange the different number of hidden layers and neurons. Neural networks are usually represented without the learnable parameters, in this example the hidden layer has four neurons, the output layer has one neuron.

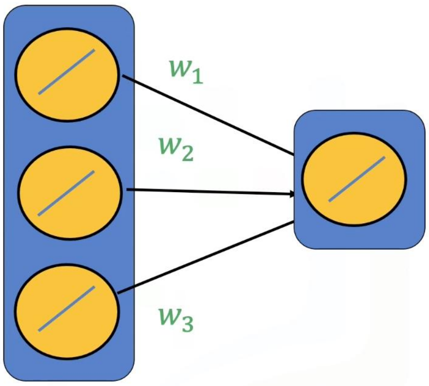
We can make multiclass predictions using neural networks, we just add more neurons to the output layer. The process can be thought of as just replacing the output layer with a **SoftMax function**, here the output layer has three neurons for three classes. For a given input we obtain an output for each neuron We choose the class according to the index of the neuron that has the largest value. In this case neuron 2 has the the largest value, so the output of our model is two.

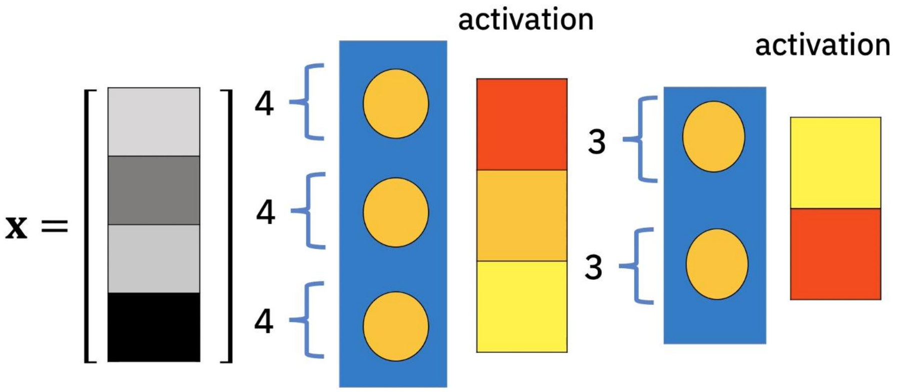




We can use the following diagram, we have 5 neurons in the output layer, one for each class dog, cats and so on.

We can add hidden layers, here we Have two hidden layers, if we have more than one hidden layer the neural network is called a **deep neural network**. More neurons or more layers may lead to overfitting.
The output or activation of each layer is the same dimension as the number of neurons, this layer has three neurons, the output or activation has three dimensions.

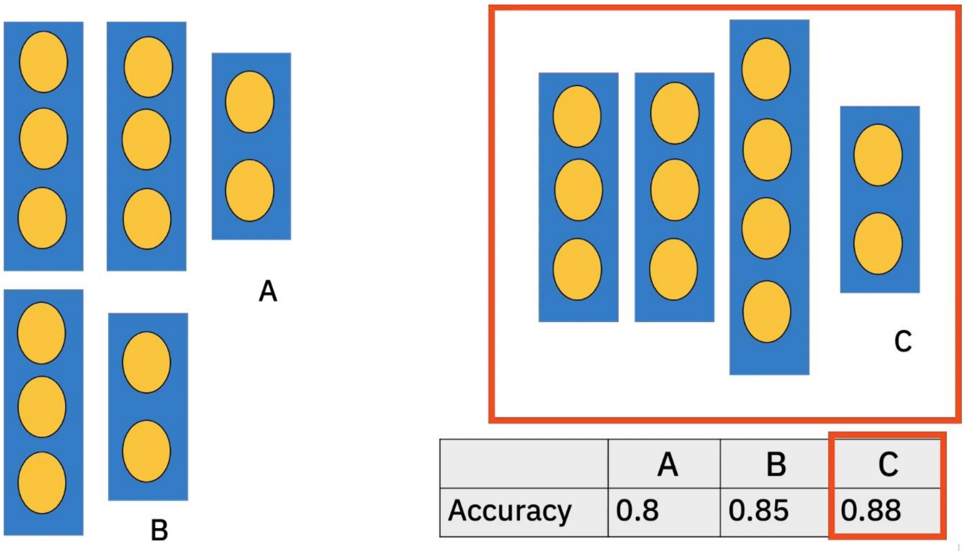




Each neuron is like a linear classifier, therefore each neuron must have the same number of inputs as the previous layer. In this case the previous layer has three neurons, so this neuron has three inputs.

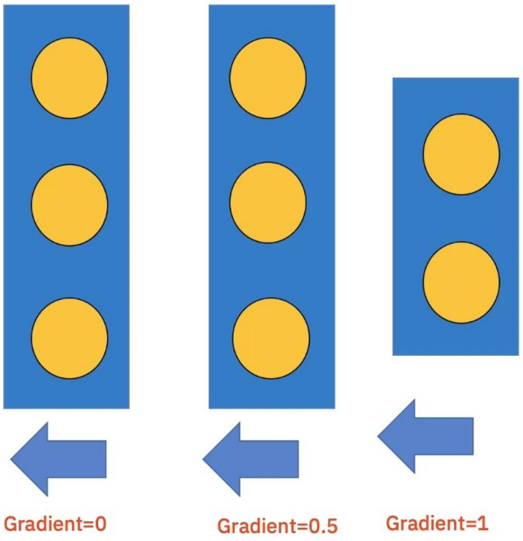
Let's see how the following neural network makes a prediction. In the following layers, consider the following input vector with four dimensions. Each neuron in the 1st layer has 4 inputs as there are three neurons, the activation has a dimension of three, each neuron in the next layer has an input dimension of 3 as there is 2 neurons in the second layer the output activation has a dimension of two.

总结：



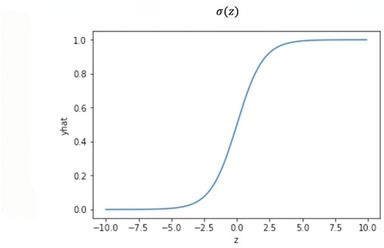
	A	B	C
Accuracy	0.8	0.85	0.88

One way to select a Fully Connected Neural Network Architecture is to use the validation data. Consider the following network Architecture A , B and C we select the Architecture with the best performance on the validation data, in this case C.



Vanishing Gradient

Sigmoid function

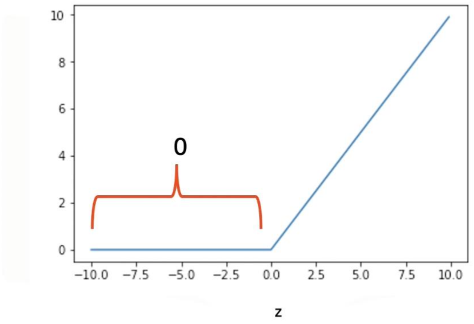


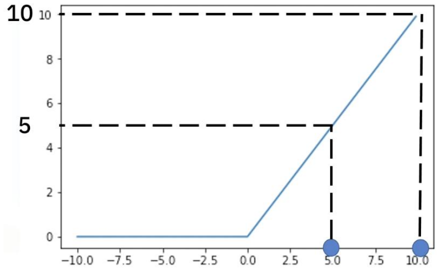
$a(z)=\frac{1}{1+e^{-z}}$

It turns out that deep networks work better, but are hard to train. If you recall to perform gradient descent to obtain our learning parameters, we have to calculate the gradient, but the deeper the network the smaller the gradient gets this is called the **vanishing gradient.** As a result its harder to train the deeper layers of the network.

One of the main drawbacks with using the sigmoid activation function is the vanishing gradient.

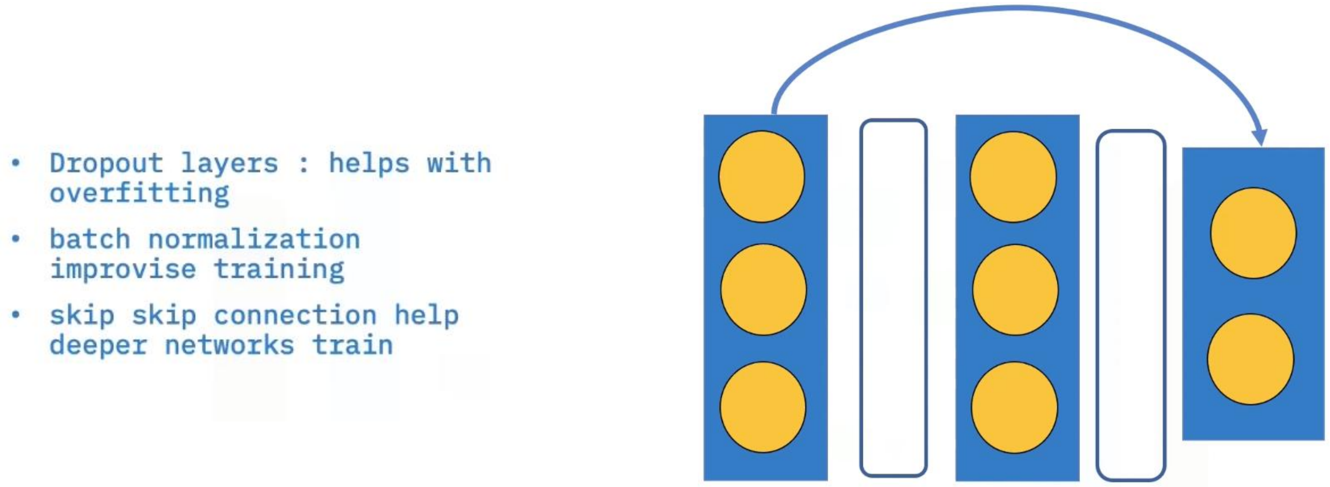
Relu





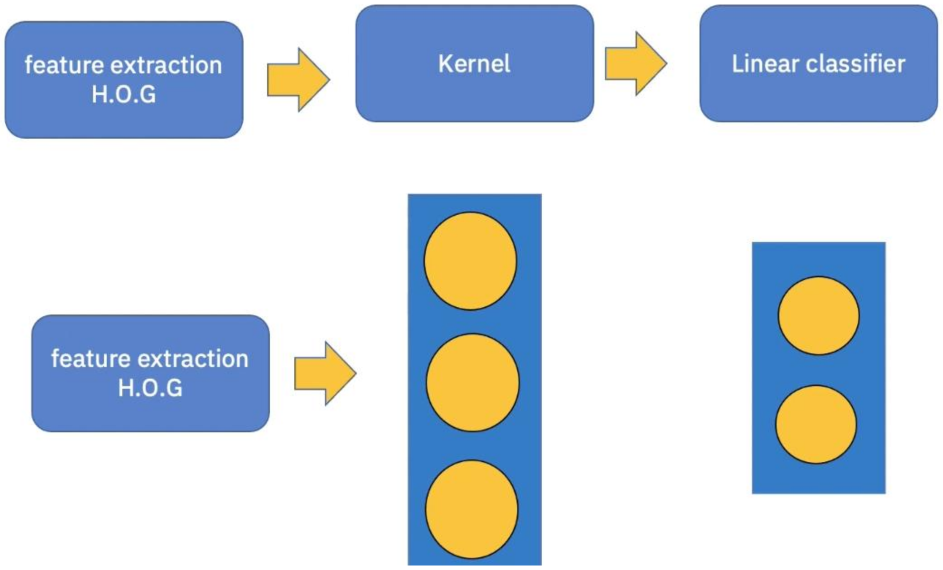
One activation function is the **rectified linear unit function** or **Relu function** for short. The value of the relu function is 0 when its input is less then zero. Relu is only used in the hidden layers. If the input z is larger than 0 the input of the function will equal its output. If the input z equals 5, the output equals 5. If the input z equals 10, the output equals 10

总结：

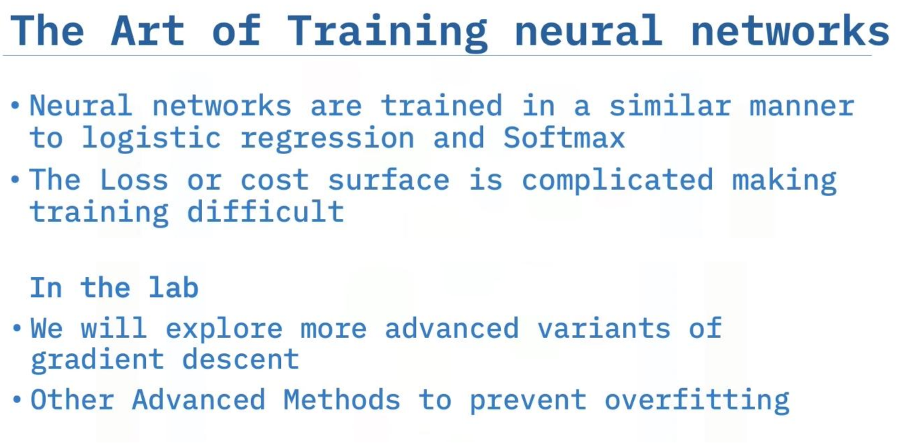


- Dropout layers : helps with overfitting
- batch normalization improvise training
- skip skip connection help deeper networks train

Networks have layers that help with training, we will skip the details, but some methods like **dropout** prevent overfitting, **batch normalization** to help with training, **skip connections** allow you too train deeper networks by connecting deeper layers during training.



The hidden Layers of Neural networks replace the Kernels is SVM's. We can use the raw Image or features like HOG.



The Art of Training neural networks

- Neural networks are trained in a similar manner to logistic regression and Softmax
- The Loss or cost surface is complicated making training difficult

In the lab

- We will explore more advanced variants of gradient descent
- Other Advanced Methods to prevent overfitting

Training neural networks is more of an art than a science, so we will use the lab to try out different methods. Generally Neural networks are trained in a similar manner to logistic regression and Softmax.
The Loss or cost surface is complicated making training difficult
In the lab: we will explore more advanced variants of gradient descent.
Other Advanced Methods to prevent overfitting. That's it, check out the lab for more

4.4 Lab: Neural Network Rectified Linear Unit (ReLU) vs Sigmoid

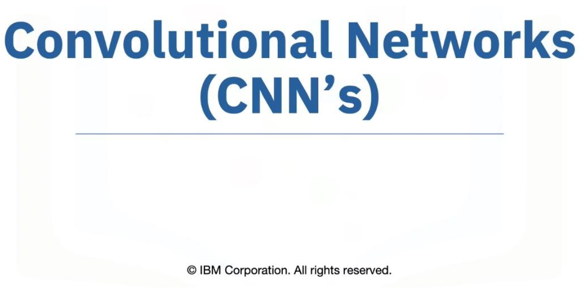
详见 4.4_Neural Network Rectified Linear Unit (ReLU) vs Sigmoid.ipynb

总结：

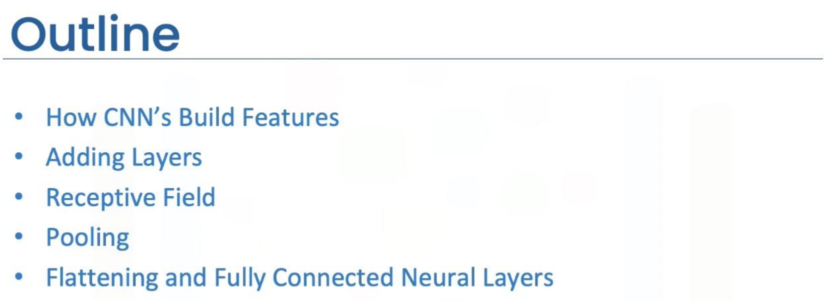
4.5 Training A Neural Network with Momentum

详见 4.5_Training A Neural Network with Momentum.ipynb

4.6 Convolutional Networks



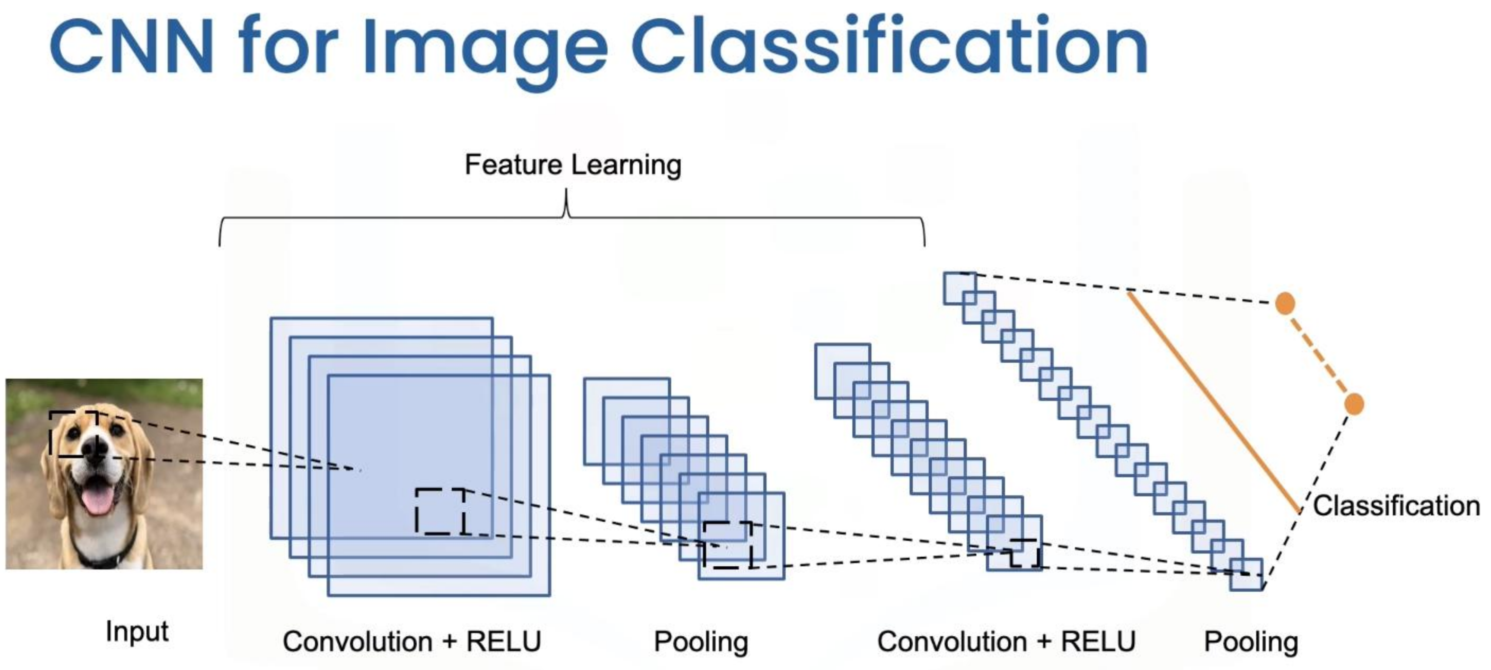
Convolutional Networks (CNN's)



Outline

- How CNN's Build Features
- Adding Layers
- Receptive Field
- Pooling
- Flattening and Fully Connected Neural Layers

In this video, you will learn about deep Convolutional Networks CNN'S for short. In this Video we Will review:
1. How CNN's Build Features
2. Adding Layers
3. Receptive Field
4. Pooling
5. Flattening and the Fully Connected Layers

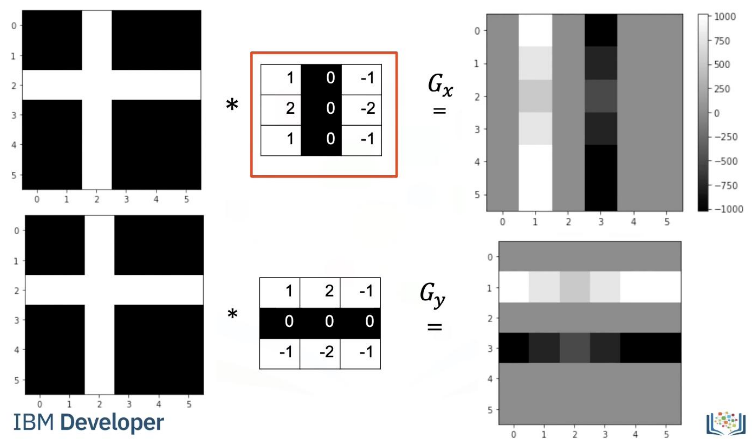


CNN for Image Classification

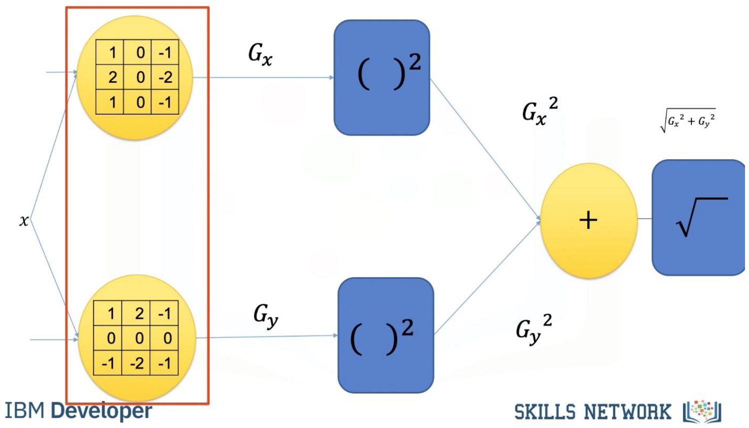
A convolutional network or CNN pictured here is a neural network with special layers, the model classifies an image by taking a part of the image, each input image will pass it through a series of convolution layers with filters, pooling layers, fully connected layers and while applying activation functions to classify an object.
Convolution and pooling layers are the first layers used to extract features from an input, these can be thought of as the feature learning layers, the fully connected layers are simply a neural network.
Both are learned simultaneously by minimizing the cross-entropy loss.

总结:

How CNN’s Build Features

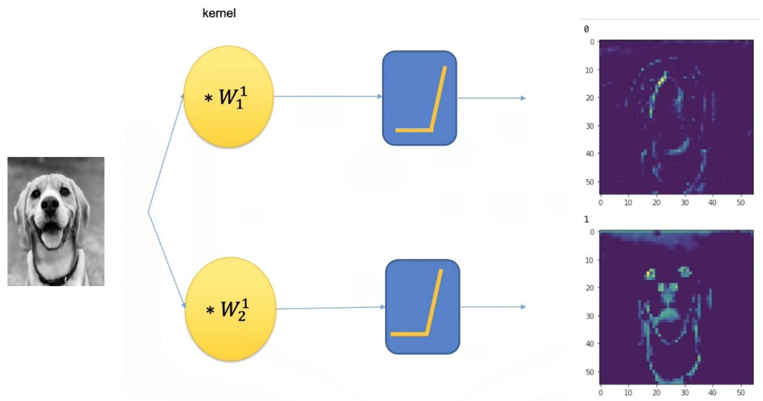


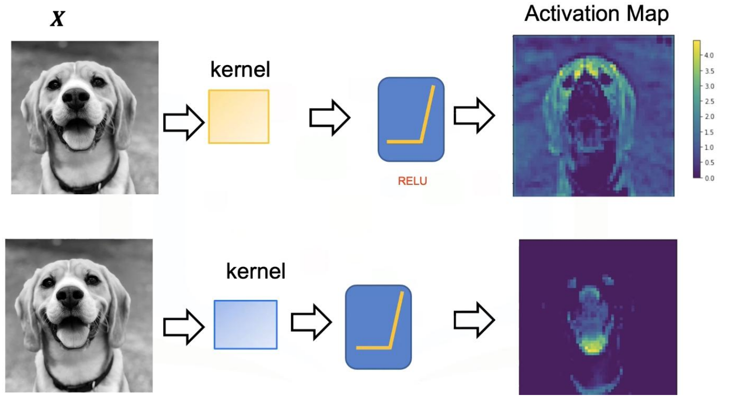
$$\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix} \qquad \begin{bmatrix}1 & 2 & -1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$



How CNN’s Build Features.
If you recall the H.O.G feature used Sobel kernels to detect vertical and horizontal edges. Looking at the kernels we see the vertical edge detector kernel looks like a vertical edge and the horizontal edge looks like a horizontal edge.

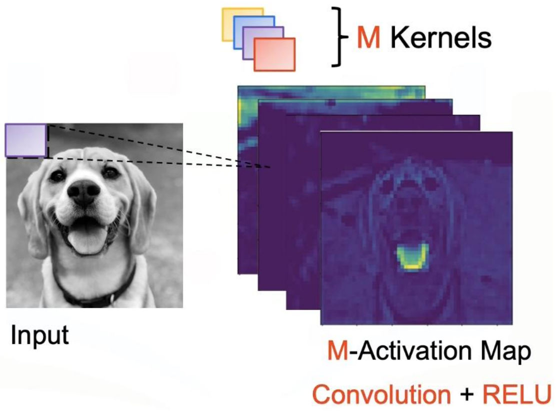
We can represent H.O.G with a diagram that looks similar to a neural network, we replace the linear function with a convolution and we have the squaring and square root operations.

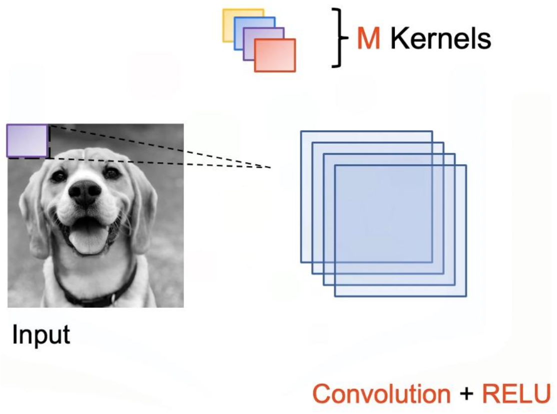




In a CNN we have neurons but the kernels are learnable parameters. The activation Functions in this case RELU are applied to each pixel, instead of an activation the output is an activation map or feature map, similar to a one channel image.

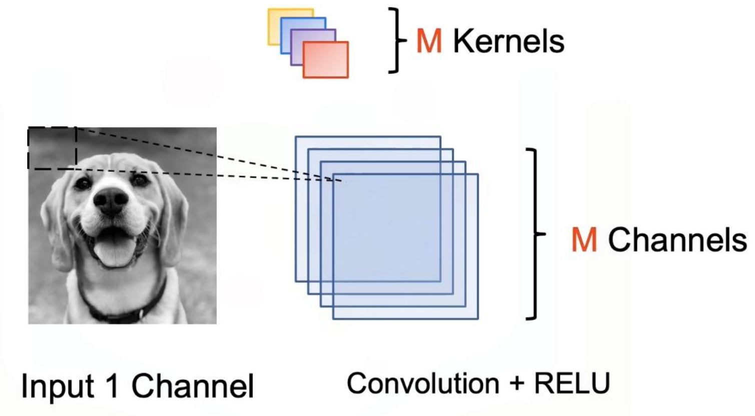
Like the HOG’s Sobel kernel each kernel of a CNN will detect a different property of the image, for example this activation map shows this kernel detects regions around the dogs eyes. This kernel picks up regions around the mouth, usually each output channel is smaller.





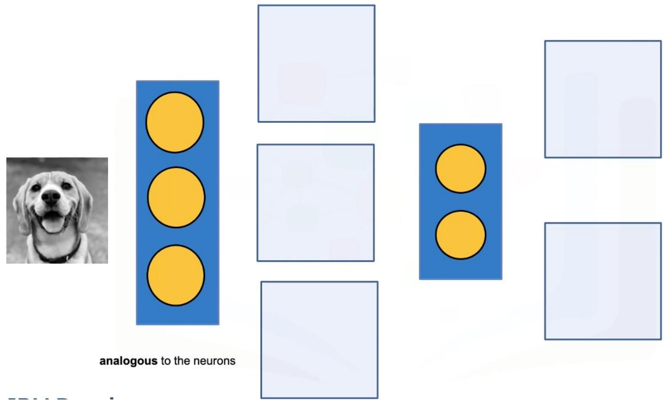
We use multiple kernels analogous to multiple neurons, if we have M Kernels we will have M feature maps. For each map we apply the Convolution + RELU. Generally we will use clear squares to represent the feature or activation maps

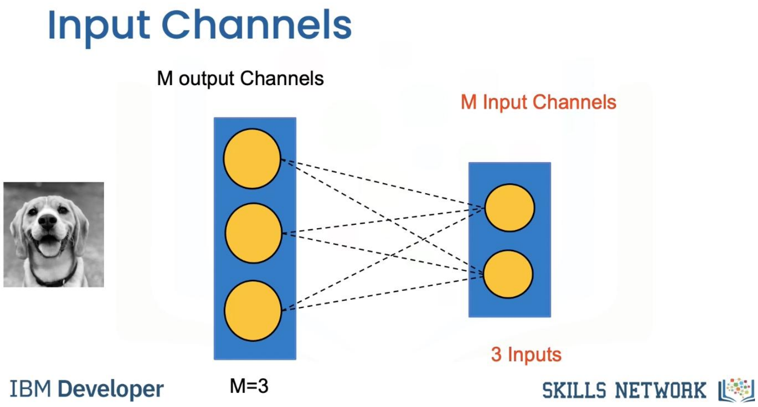
总结：



A gray scale image Can be seen as a 1 channel input, if we have **M kernels**, each feature map will be a channel, therefore we will have **M output channels**.

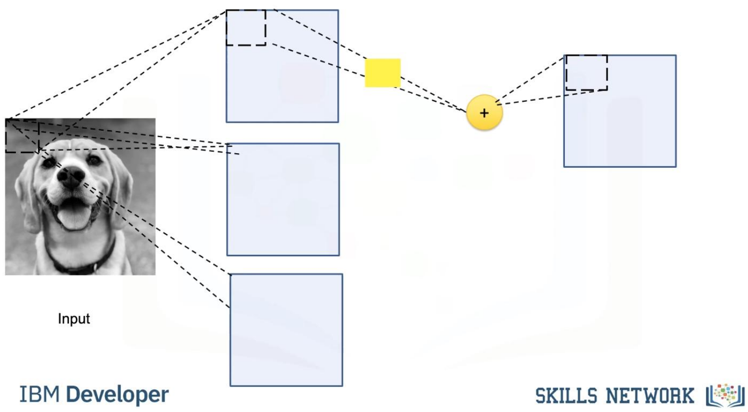
Adding Layers

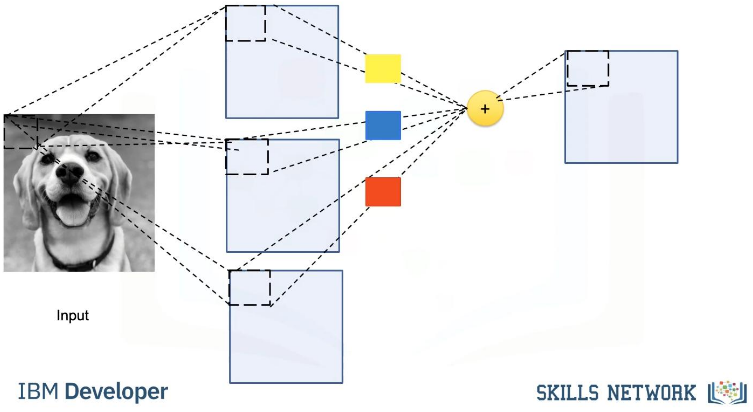




Adding Layers.
We can also stack convolutional layers, each output channel is analogous to the neurons, like a neuron. The input of the next layer is equal to the output of the previous layer. Here we have three outputs, the next layer will take three outputs as Inputs, if this layer has two outputs it will output two feature maps.

The next layer will apply a convolutional kernel to each input, then add them together, then apply an activation function.

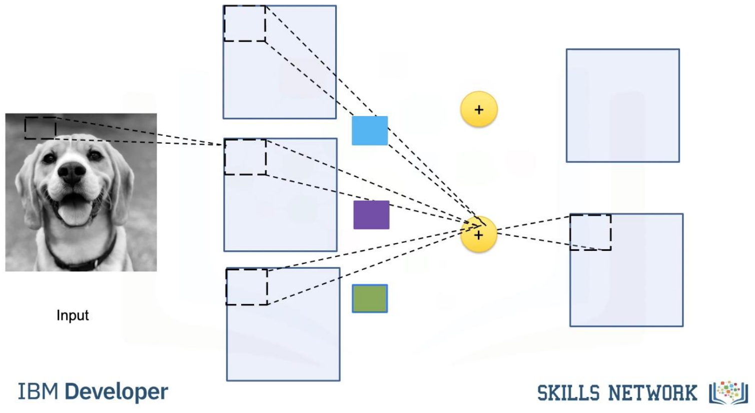


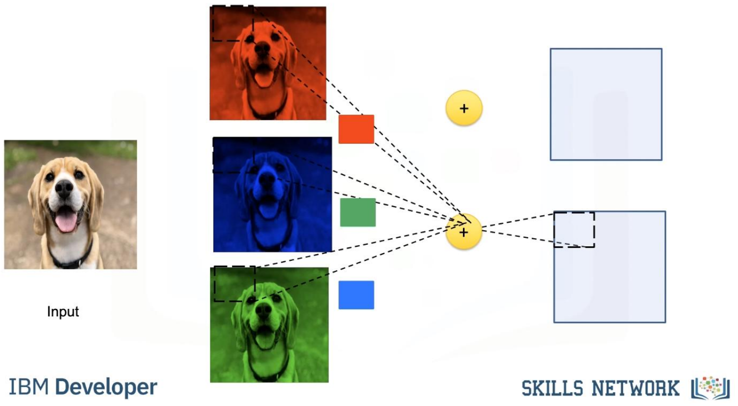


The neurons are replaced with kernels. The previous layer has three output channels, for the first input we apply the convolution and activation to the output of the first channel, we obtain the feature map.

We repeat the process for the second input channel **adding it to the activation map**, repeating for the third input channel.

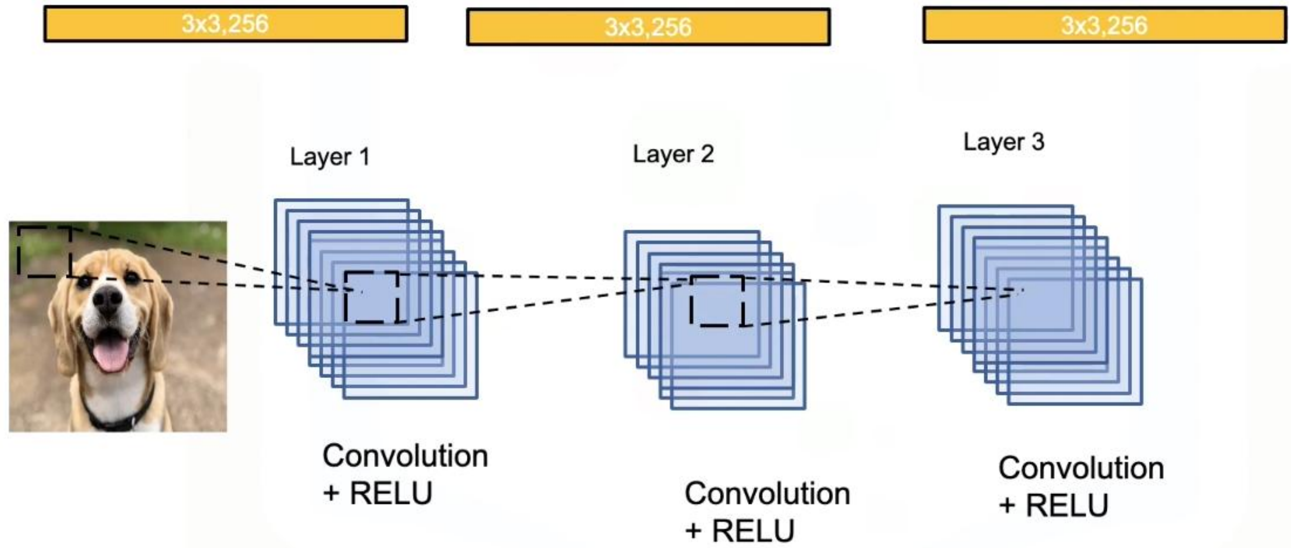
总结：





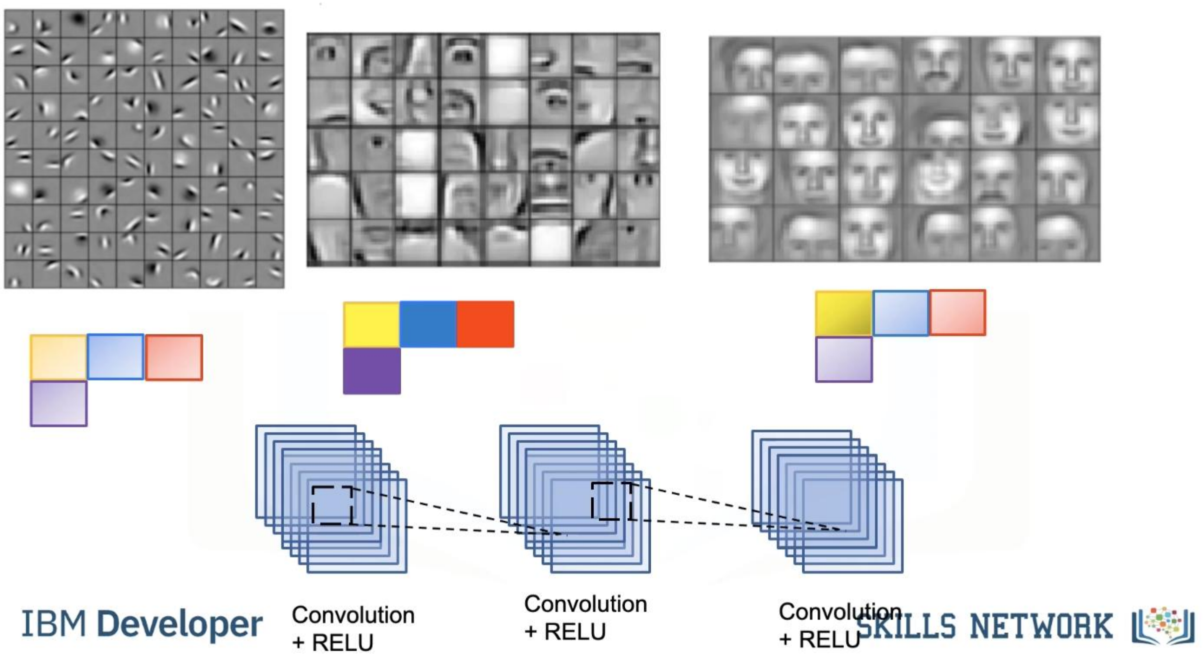
The process is repeated, for the next input.

We can process color images by having three input channels, just like a neural network we can stack layers, using the 3d representation.



We can also represent them with these yellow boxes indicating the kernel size and the number of channels.

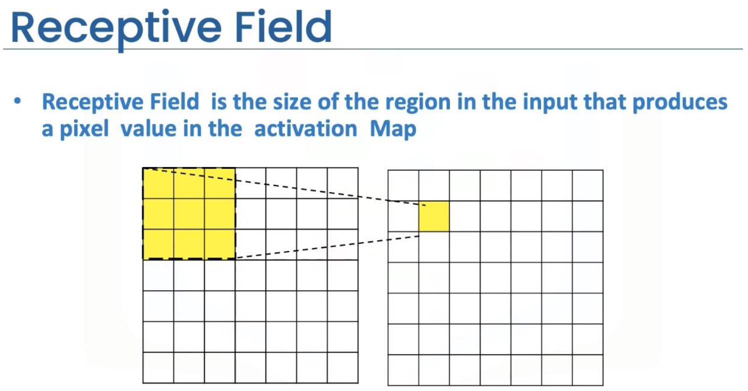
Method similar to CNN https://developer.nvidia.com/blog/deep-learning-nutshell-core-concepts/



It’s helpful to look at the kernels to understand what the different layers are doing. if you recall the Sobel kernels looked like they detect vertical and horizontal edges they were trying to detect.
Consider a CNN used to see faces, the kernels in the first layer look like edges and corners, the second layer looks like parts of the face, the final layer looks like faces. We see that adding more layers builds more complex features.

总结：

Receptive Field

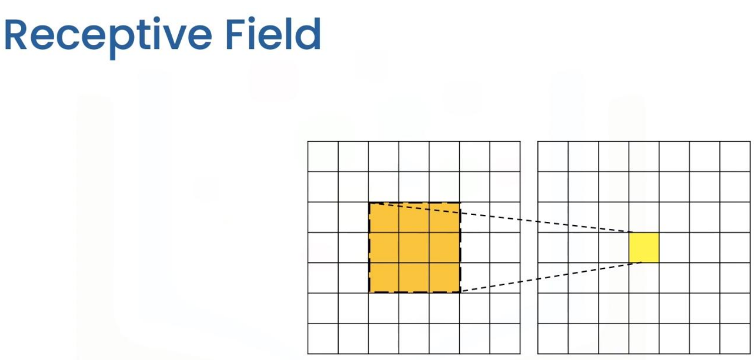


Receptive Field

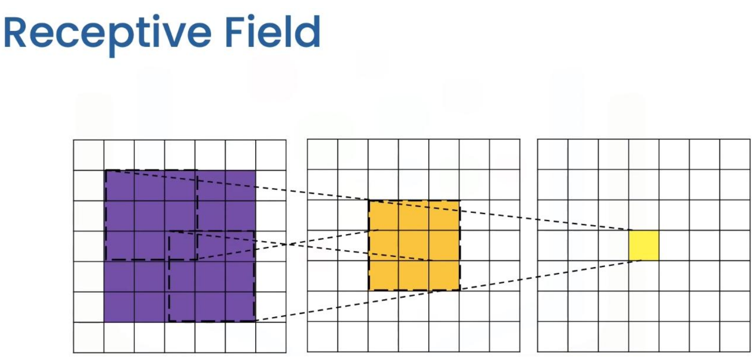
- **Receptive Field is the size of the region in the input that produces a pixel value in the activation Map**

Receptive Field is another important factor in CNNs.

Receptive Field is the size of the region in the input that produces a pixel value in the activation map, consider the following pixel in the following activation map, its Receptive Field is given by the larger the Receptive Field the more information the activation map contains about the entire image.



Receptive Field

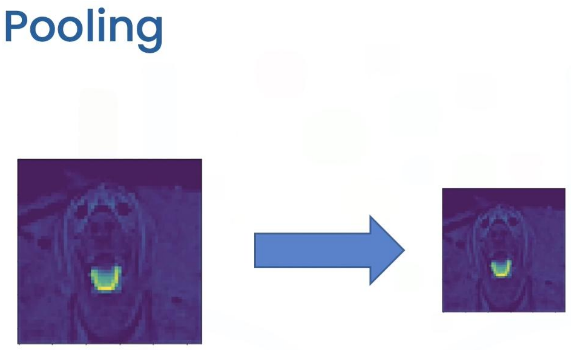


Receptive Field

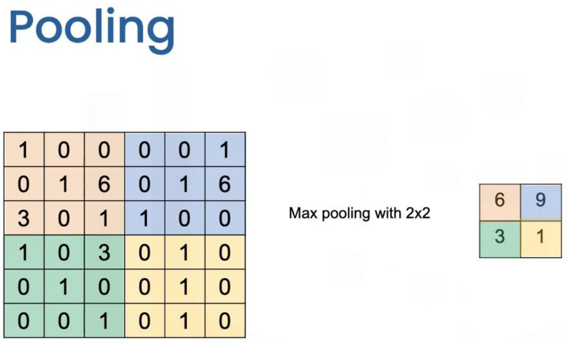
We can increase the receptive field by adding more layers, this requires less parameters then increasing the size of the kernel. If we have one layer the receptive field would be the following region.

Adding a second layer increases the receptive field further.

Pooling



Pooling



Pooling

1	0	0	0	0	1
0	1	6	0	1	6
3	0	1	1	0	0
1	0	3	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

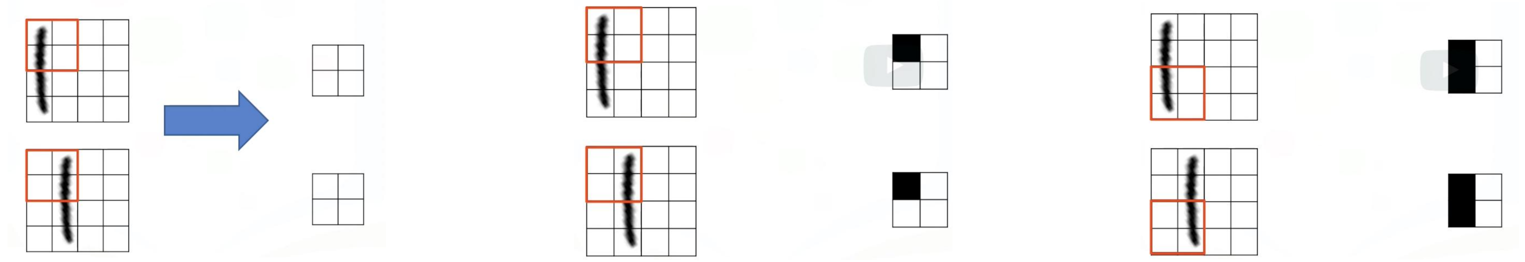
Max pooling with 2x2

6	9
3	1

Pooling helps to reduce the number of parameters, increases the receptive field while preserving the important features. To make it easier to understand, we can think about it as resizing the image.

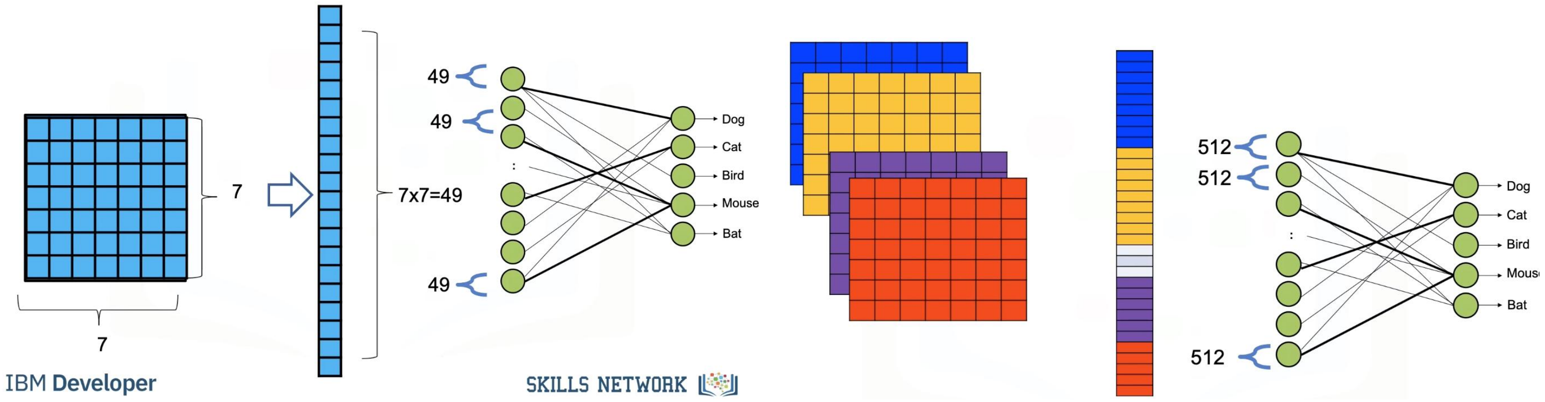
Max pooling is the most popular type of pooling. Consider the feature map. We Apply Max pooling with dimensions 2x2, it takes the maximum pixel value and we get a smaller feature map.

总结:



Pooling also makes CNN's more immutable to small changes in the image.
We have two images they are identical but one slightly shifted let's see the activation map after max pooling, we apply max pooling in first region and repeat. We see the activation map output is now identical

Flattening and Fully Connected Neural Network



Finally we have Flattening and the Fully Connected layers.
We simply flatten or reshape the output of the Feature Learning layers and use them as an input to the fully connected layers, for example if the output of the max pooling layer is 7 units of width and 7 units of height, we flatten or reshape the output, this is analogous to a feature vector this is the input to the fully connected layer, each neuron has the input dimension as a flatten output.

For more channels we apply the similar procedure, if we have 32 output channels each channel is 4x4 for a total of 16 elements as there are 32 channels multiplied by 16 we have a total of 512 elements, we flatten or reshape the output to have 512 outputs as a result each neuron will have 512 input dimensions.

4.7 Lab: Convolutional Neural Network

详见 4.7_Convolutional Neural Network.ipynb

4.8 Lab: Data Augmentation

详见 4.8_Data Augmentation.ipynb

总结:

4.9 CNN Architectures

Important CNN Architecture

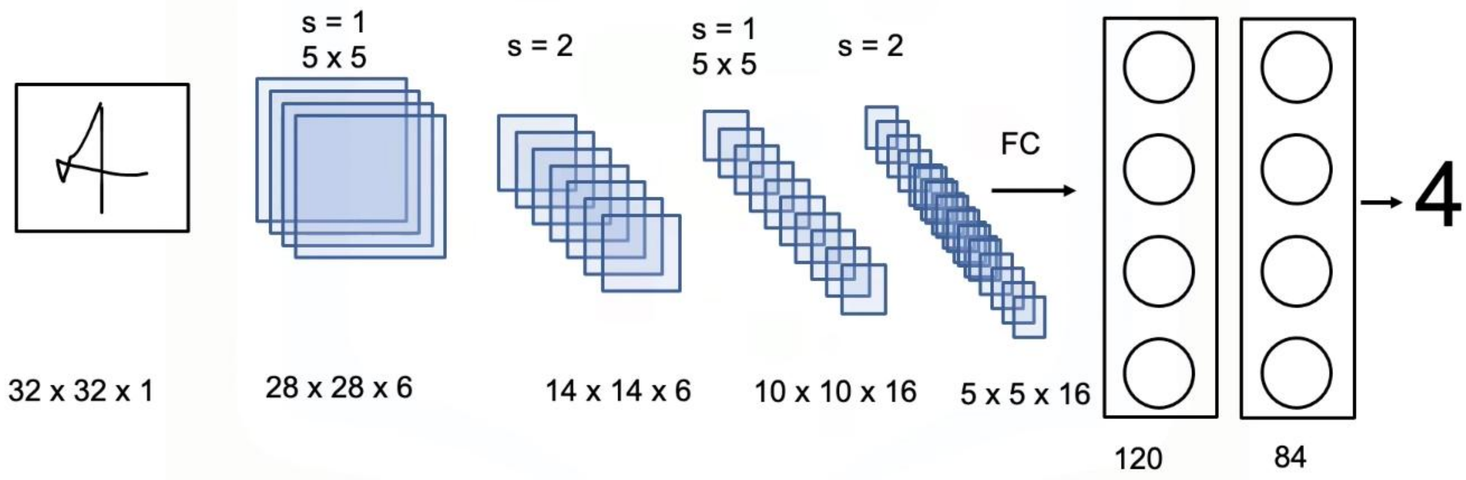
Important CNN architectures include :

- LeNet-5
- AlexNet
- VGGNet
- ResNet

- Transfer Learning

In this video, you will learn about different CNN Architectures.
Popular architectures include :
LeNet five, AlexNet, VGGNet, ResNet.

We will go over a few of them in this session. We will also cover Transfer Learning.



One of the first CNNs was proposed by Yann LeCun et al. in 1989. The most successful use case of the LeNet-5 is the MNIST Data set of handwritten digits.
LeNet receives an input image, normally a grayscale image, it uses a 5 by 5 filter with a stride 1 and results in a volume of 28 by 28 outputs. The next layer is a pooling layer with 14 by 14 outputs. It repeats itself with a filter and pooling layer till it gets to the fully connected layers where it flattens to create 120 neurons and another with 84 neurons while using a sigmoid activation function to produce an output.
For a while CNNs dropped out of popularity for image classification, and support vector machines became the default standard.

AlexNet

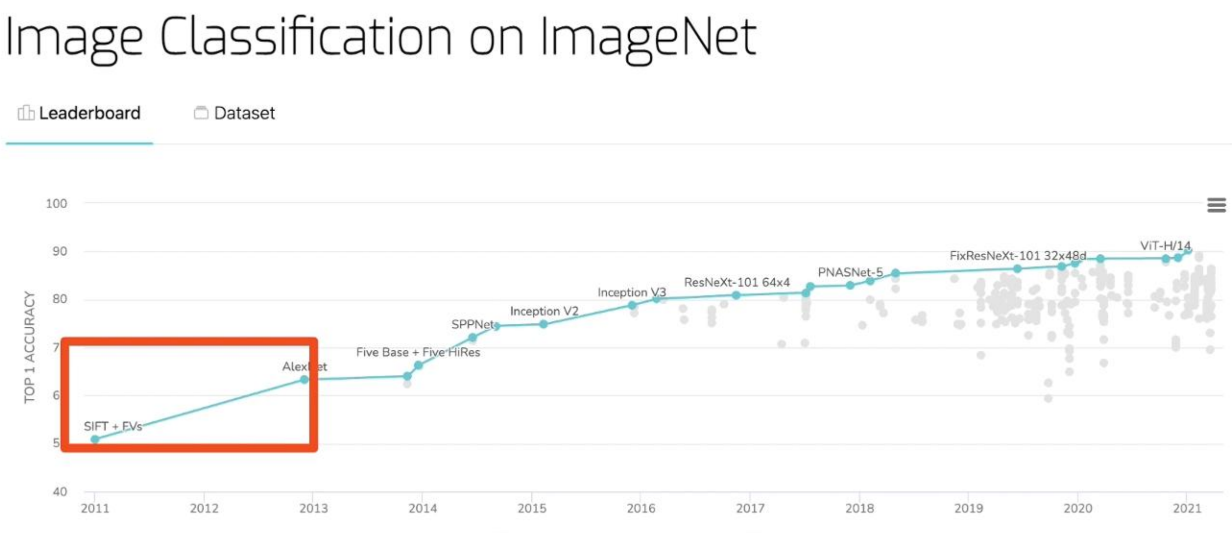


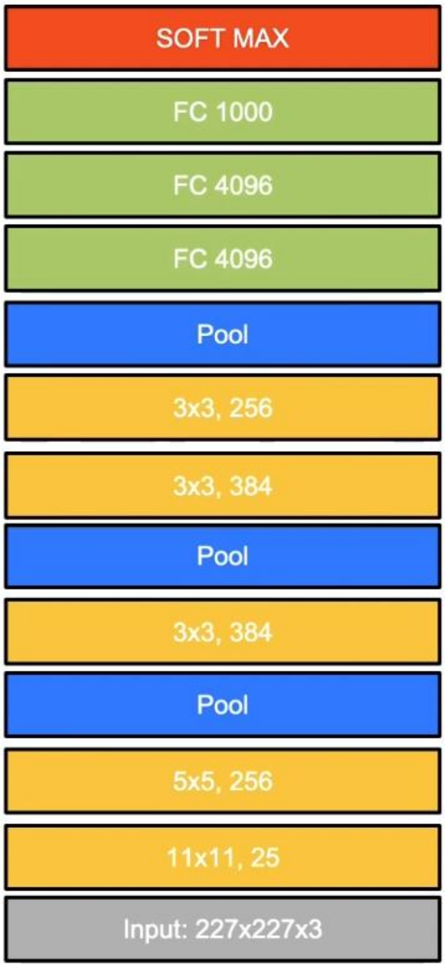
Image Classification on ImageNet

Source: Papers with Code - Imagenet Benchmark

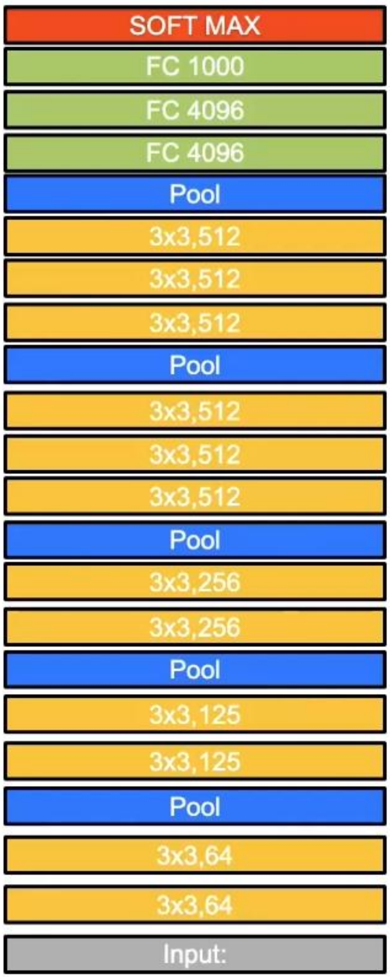
总结：

If you want to compare any image classification methods you compare the classification accuracy on a dataset. ImageNet is a benchmark dataset i.e this is the one that everyone uses to see who has the best image classification method. Here is an image of the top performer every year. Prior to 2012 a method using SIFT a feature like hog took the top spot with 51 precent accuracy. After 2012, AlexNet smashed this record with 63.3% accuracy. This jump was so large everyone started using CNNs for image classification.

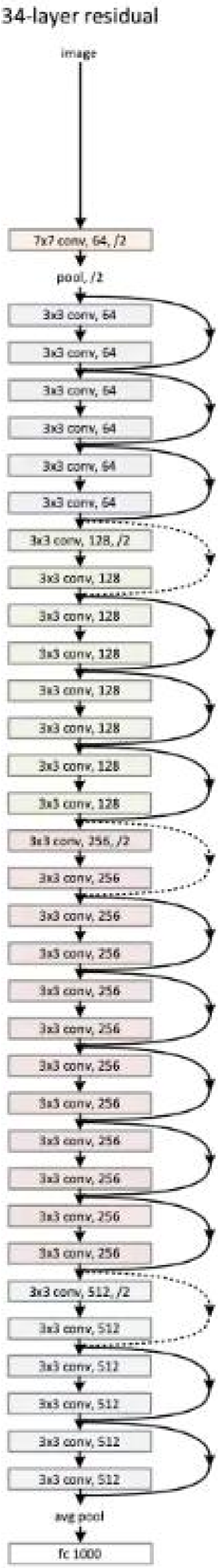
AlexNet



VGGNet



ResNet



Here is a block diagram of Alex Net we see the network has lots of parameters, we see convolution kernels are of different sizes. If you recall more parameters means you require more data. We see the first convolution kernel layers are of shape 11x11 with 25 channels, this is quite a large number of parameters.

The VGG Network is a Very Deep Convolutional Network that was developed out of the need to reduce the number of parameters in the Convolution layers and improve on training time, it also showed in general that deeper networks performed better.
VGGNet has multiple variants like the VGG 19 and VGG 16, where 16 stands for the number of layers in the network.
The key insight gained by the VGG networks is we could replace the larger kernels in the convolution layer by stacking convolution layers with 3 by 3 kernels, keeping the same receptive field while reducing the number of parameters.
This also reduced the number of computations by reducing the number of operations in the convolution layers and decreasing the feature map size.

总结：

ResNet

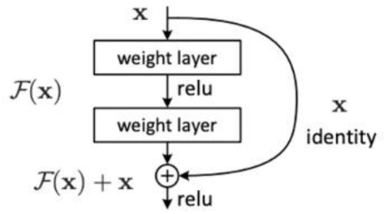

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

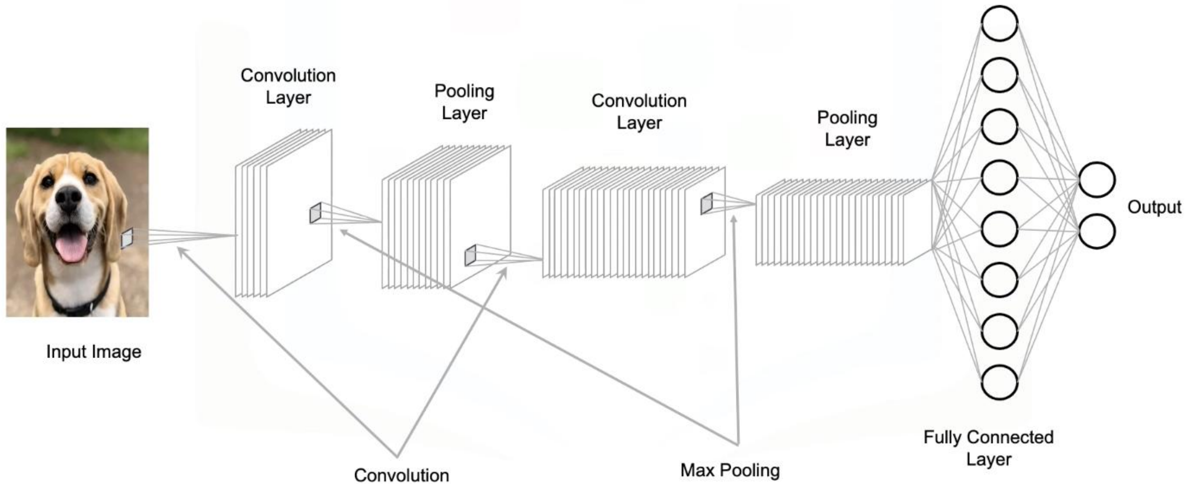
Figure 2. Residual learning: a building block.

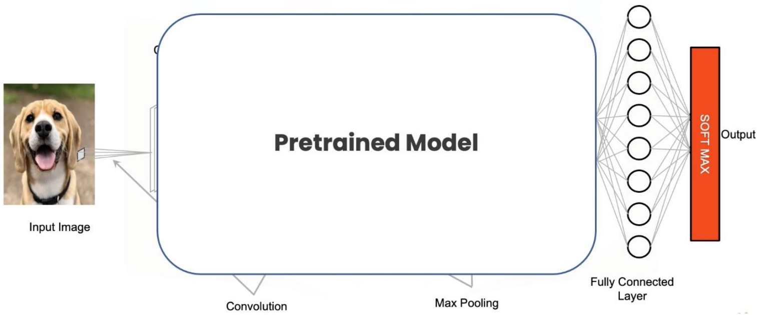
As CNNs got deeper the vanishing gradient began to become a problem again. ResNet help solve the problem by introducing Residual learning:
Residual layers or skip connections allows the gradient to bypass different layers improving performance. We can now build much deeper Networks.

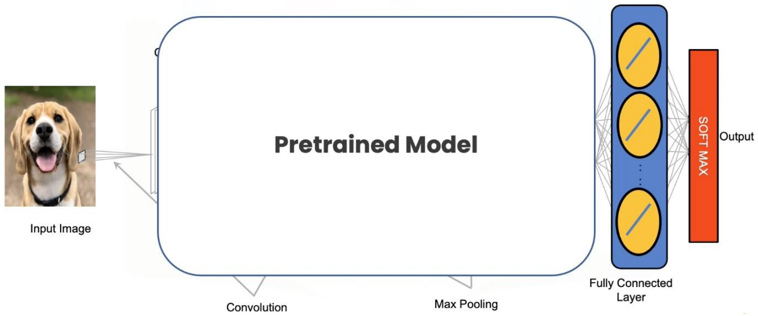
Here is a 32 layer network from the paper Deep Residual Learning for Image Recognition.

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

TRANSFER LEARNING







Transfer learning is where you use a pre trained cnn to classify an image. Instead of building your own network you can use the CNN Architectures we discussed, Pre-trained CNN 's have been trained with vast amount of data.

We will cover the simplest method, where we replace the SoftMax layer with our own SoftMax layer. The number of neurons is equal to the number of classes, we then train the SoftMax layer on the dataset we would like to classify.

The input dimension of each neuron in the SoftMax layer is equal to the last number of neurons fully connected layer. The Pretrained model can be thought of as a feature generator, depending on the size of your dataset you can use SVM instead of the SoftMax layer. Check out the labs. for more

总结：

4.10 Reading: Deploying a Model

另见下载下来的网页：4.10_Reading_Deploying a Model

4.11 Lab: Use CNN for "Hotdog, Not Hotdog" Classifier and Deploy Model with CV Studio, Incomplete

另见下载下来的网页：4.11_Use CNN for Hotdog-Not Hotdog Classifier and Deploy Model with CV Studio

总结：