ML

NIRMA
UNIVERSITY

Convolutional Neural Networks

Machine Learning Lecture: 3

Image Data

Image data is represented using matrices of pixels cascaded on top of each other(RGB).

For this kind of data, spatial information plays a huge role.

Unlike data like housing data, etc. the features are not localized to one set of locations but they are spread around the image and may occupy the edges, etc.

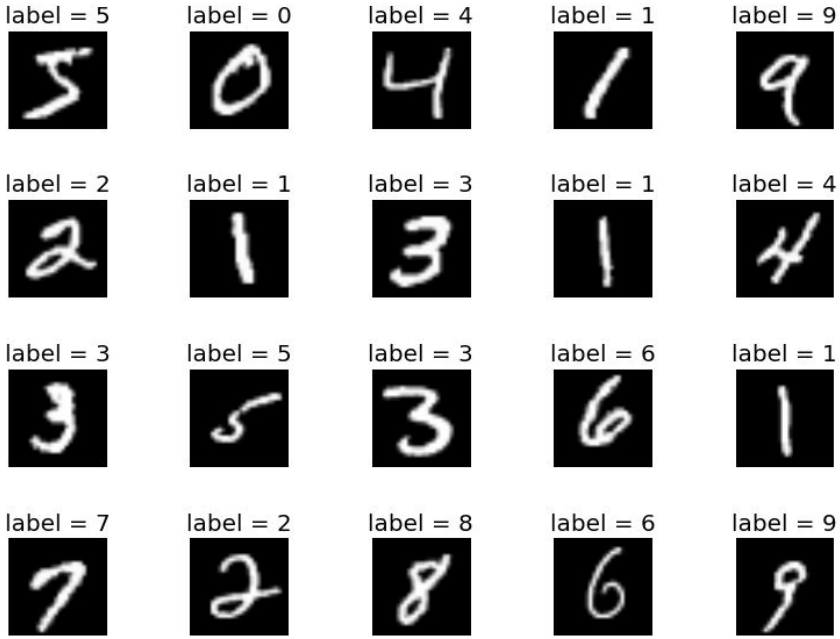
A feed forward ANN would not be able to provide high accuracies for images.

Why Did ANN work for MNIST?

Localized Features.

No noise.

Similar background throughout the dataset.



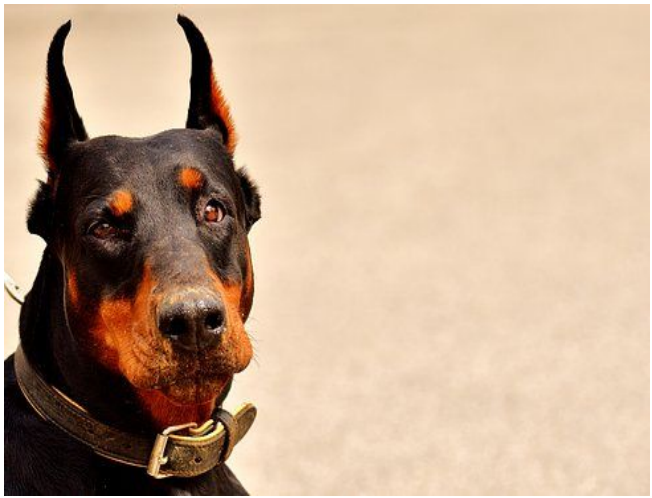
Why ANN may not work

- Large number of neurons, for an RGB image of 256x256 resolution, neurons in input layer = 1,96,608
- The arrangement of object in an image may vary, for example, a dog may be facing the camera or it may be sleeping, etc.
- Not just the orientation, but also the size, colour, and background of the object varies from image to image which may lead to improper weights for important features.

Dog but not dog

Dog 1

Dog 2



Hence we need to come up with a different network which can keep the spatial arrangement and learn from it.

Convolutional Neural Nets

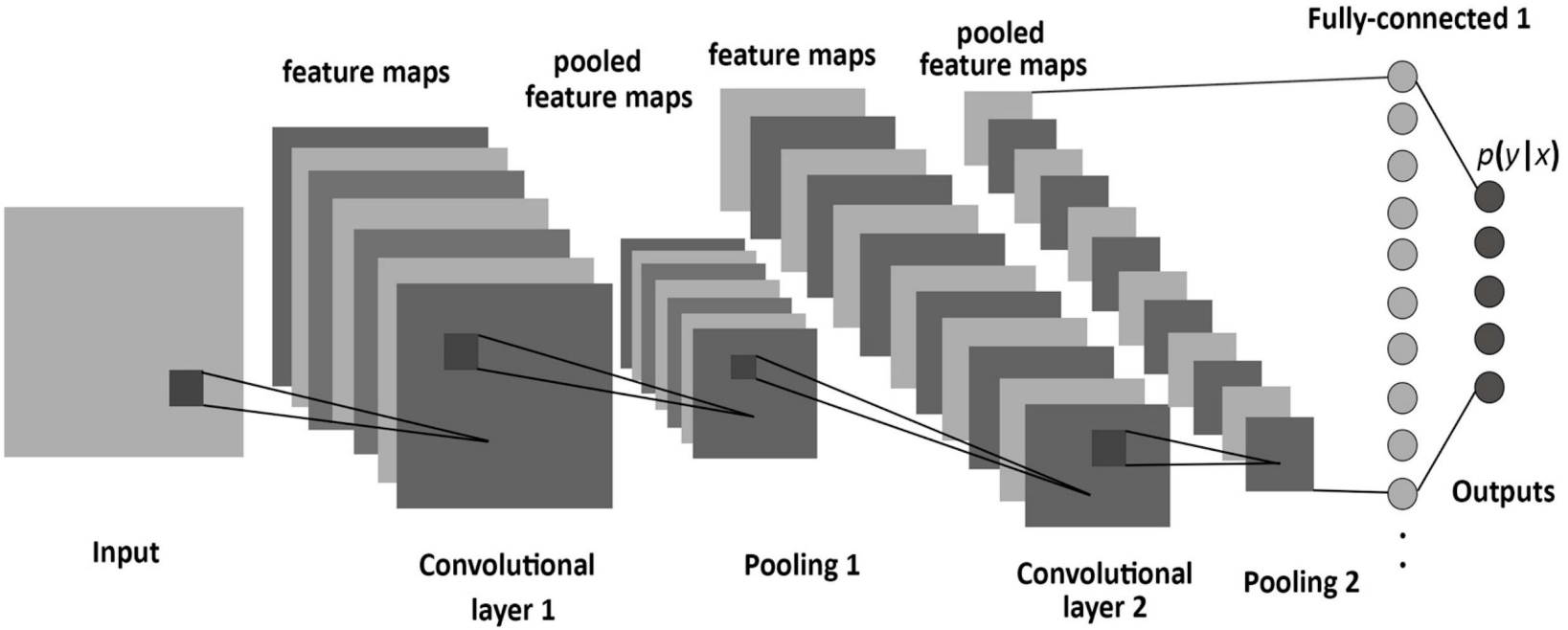
What are they?

CNNSs look at spatially local patterns by applying the same geometric transformation to different spatial locations (patches) in an input matrix(pixels of image).

This results in representations that are translation invariant, making convolution layers highly data efficient and modular.

They follow a weight sharing mechanism to overcome the large number of neurons problem faced in ANN.

CNN



Layers of a CNN

Convolution Layer

They have 2 fundamental properties

1. Sparse Connectivity
    a. Not all neurons in i-th layer is connected with all neurons in (i+1)-th layer.
2. Parameter Sharing
    a. Not all connections have unique weights, but they share a set of filters or kernels.

Convolution Layer

Every Conv layer accepts input spatially and has 3D volume, i.e Height, Width, and Depth of the input.

The weights are represented as 3D volumes known as filters which have spatial extent and depth which is usually equal to the depth of the input.

Different filter columns(filters) learn different features.

Their weights can be randomly initialized.

Hyperparameters of Conv Layer

1. Filter Size.
2. Depth of output(number of channels generated after conv operation), this is same as the number of filters/kernels used.
3. Stride - It is the shift, in terms of pixels, that is utilized while performing conv operation.
4. Padding - We can add rows and columns on the exterior of the input to conv layer, this is called padding. We usually keep the pixel values of these Rs and Cs as 0.

Conv Operation

Takes an input volume of size $\mathbf{W_1 \times H_1 \times D_1}$

Requires four hyperparameters:
K - Number of filters,
F - Filter's spatial extent (size of filter),
S - Stride,
P - Extent of zero padding.

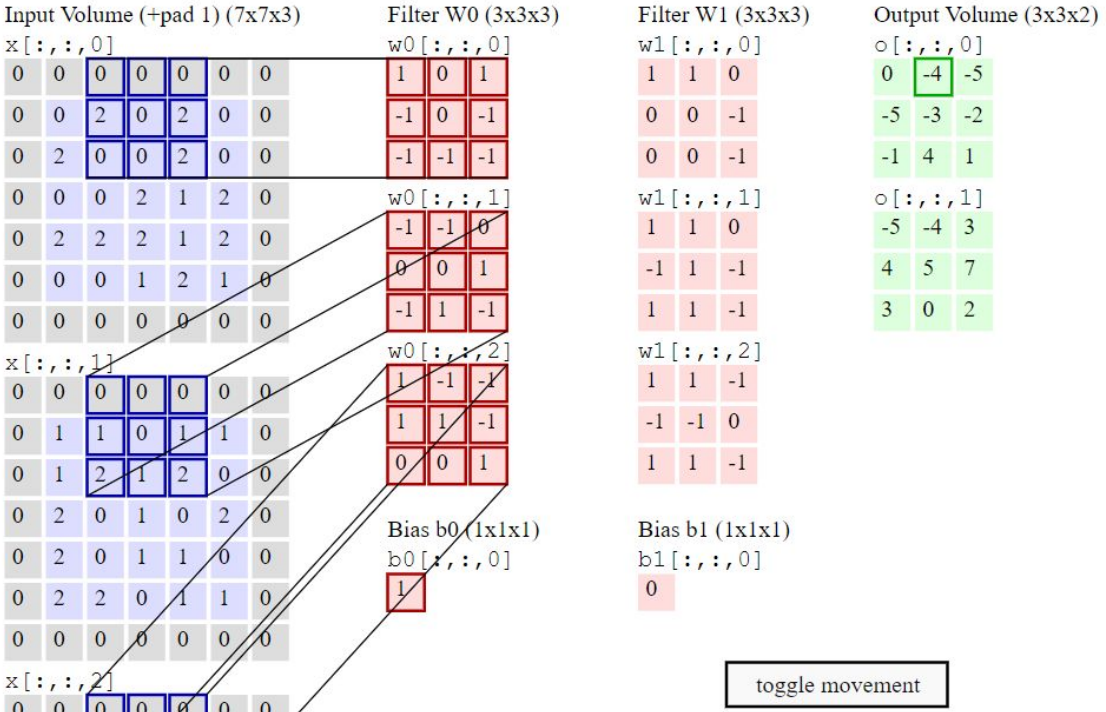
Conv Operation

Produces an output of volume $\mathbf{W_2 \times H_2 \times D_2}$ where:

$\mathbf{W_2 = [(W_1 - F + 2P)/S] + 1}$

$\mathbf{H_2 = [(H_1 - F + 2P)/S] + 1}$ (i.e. width and height are computed equally by symmetry)

$\mathbf{D_2 = K}$


Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 2 0 2 0 0
0 2 0 0 2 0 0
0 0 0 2 1 2 0
0 2 2 2 1 2 0
0 0 0 1 2 1 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 1 1 0 1 1 0
0 1 2 1 2 0 0
0 2 0 1 0 2 0
0 2 0 1 1 0 0
0 2 2 0 1 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 2 0 0 0
0 2 2 2 0 1 0
0 1 2 2 1 2 0
0 2 2 1 0 2 0
0 1 1 2 2 2 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
1 0 1
-1 0 -1
-1 -1 -1
w0[:,:,1]
-1 -1 0
0 0 1
-1 1 -1
w0[:,:,2]
1 -1 -1
1 1 -1
0 0 1
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
1 1 0
0 0 -1
0 0 -1
w1[:,:,1]
1 1 0
-1 1 -1
1 1 -1
w1[:,:,2]
1 1 -1
-1 -1 0
1 1 -1
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
0 -4 -5
-5 -3 -2
-1 4 1
o[:,:,1]
-5 -4 3
4 5 7
3 0 2
toggle movement

Activation - ReLU

ReLU is the widely used activation function for a CNN.

Every convolution operation is followed by a ReLU operation.

You may find people keeping ReLU as a separate layer after convolution layer.

Pooling Layer

Pooling layer or downsampling layer selects value of some neurons and discards the rest. This is controlled by a threshold.

It is very common to put pooling layer in a CNN to reduce the number of neurons that propagate further and, to save space and time.

Pooling action takes place similar to conv, except it has no trainable parameters.

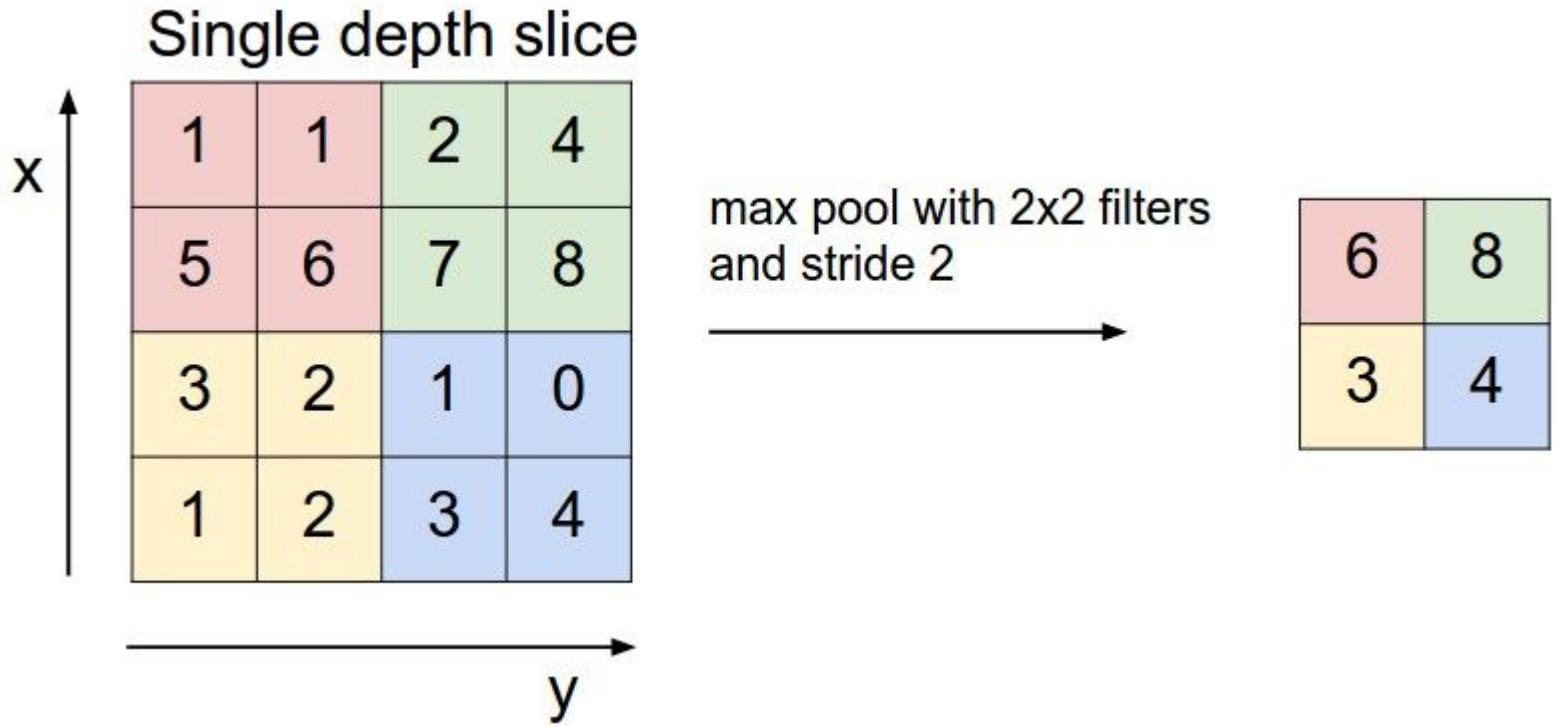
Max Pooling

Max Pool looks at a volume of the size equal to its spatial extent and applies MAX function on the values, it keeps the neuron with the maximum value and discards the rest.

Most common Max pool variant is 2x2 pool with a stride of 2, it reduces the input by 75% [Only 1 out of 4 is kept]

We keep the task of downsampling to pool layer and try to maintain the spatial extent in conv layers by appropriate padding.For eg. P = (F-1)/2 for S = 1

Max Pooling



Max Pooling

It requires 2 hyperparameters and no trainable parameters.

1. **F** - Spatial extent of Pool Filter
2. **S** - Stride

For an input, $W_1 \times H_1 \times D_1$, it outputs a volume of,

$$W_2 = [(W_1 - F)/S] + 1$$

$$H_2 = [(H_1 - F)/S] + 1$$

$$D_2 = D_1$$

Fully Connected Layer

Just like ANN, the fully connected layer has each neuron from previous layer connected to each neuron in current layer.

We typically see fully connected layer in a CNN after few layers of Conv and Pool, since Conv and Pool are responsible for feature extraction, we gather these features for classification by applying fully connected layers at the end.

They don’t follow parameter sharing.

Softmax

Softmax functions are most often used as the output of a classifier, to represent the probability distribution over n different classes. Softmax units naturally represent a probability distribution (Bernoulli distribution) over a discrete variable with k possible values, so they may be used as a kind of switch.

$$\mathrm{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Softmax

Even though the output of Softmax lies between 0 and 1, it is different from Sigmoid, as the sum of all the outputs of a Softmax will be 1 since it is a probability distribution, this accurately presents the confidence.

Benchmark CNN Architectures

- LeNet
- AlexNet
- Inception
- Xception
- VGG16 / VGG19
- ResNet
- YOLO
- ZFNet

For implementations

In practice: use whatever works best on ImageNet. If you're feeling a bit of a fatigue in thinking about the architectural decisions, you'll be pleased to know that in 90% or more of applications you should not have to worry about these. Instead of rolling your own architecture for a problem, you should look at whatever architecture currently works best on ImageNet, download a pretrained model and finetune it on your data. You should rarely ever have to train a ConvNet from scratch or design one from scratch.

Issues with CNN

CNNs usually require large amount of memory and may need a GPU for fast computation.

From each Conv Layer we generate activations which are stored in memory for further propagation.

Every Conv layer has kernels which constitute trainable parameters and, gradients during back-propagation.

We need some extra memory to store the complete batch volumes, intermediate operations, images, etc.