STEVENS INSTITUTE OF TECHNOLOGY

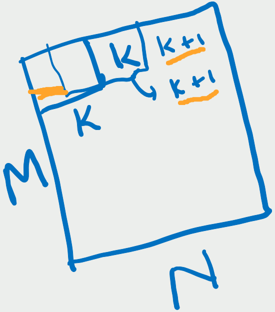
Deep Learning: Convolutional Neural Network

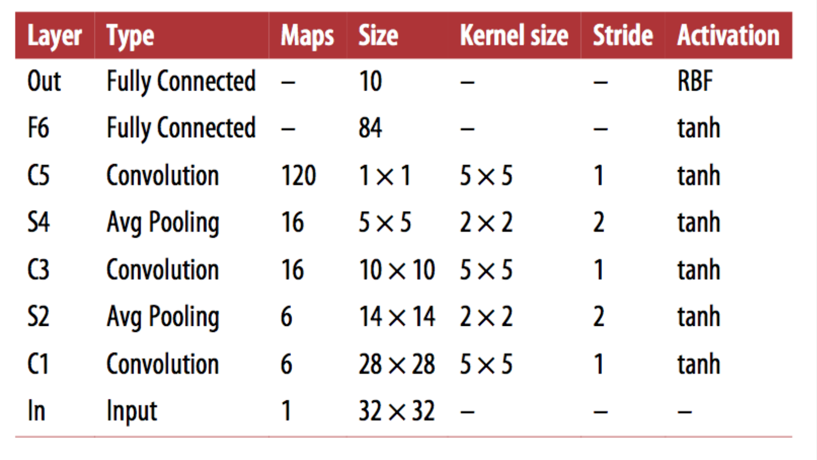
Rensheng Wang,
`https://sit.instructure.com/courses/29876`

Feburary 28, 2019

Classic Networks – LeNet-5

- ❑ The LeNet-5 architecture is perhaps the most widely known CNN architecture. It was created by Yann LeCun in 1998 and widely used for hand written digit recognition (MNIST).
- ❑ It is composed of the layers shown below.



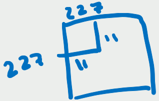


Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

i/p → M×N
Kernel → K×K
o/p → ?
$= M-K+1, N-K+1$
↓
for stride = 1

for color → 5×5×3 channels
↳ RGB

Classic Networks – AlexNet



- ❑ It was developed by Alex Krizhevsky et al (hence the name).
- ❑ It is quite similar to LeNet-5, only much larger and deeper, and it was the first to stack convolutional layers directly on top of each other, instead of stacking a pooling layer on top of each convolutional layer

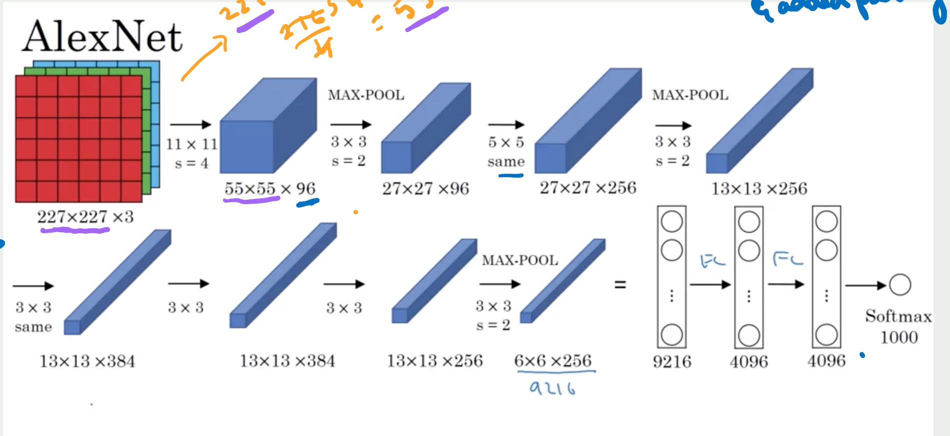
227-11 = 216

216/4 = 54+1 = 55

same → stride=1 & added padding to make same o/p dimensn

how many parameters for 11x11 kernel?

total 121 elements ✓ but when we apply to Alexnet 121x3 = 363 parameters



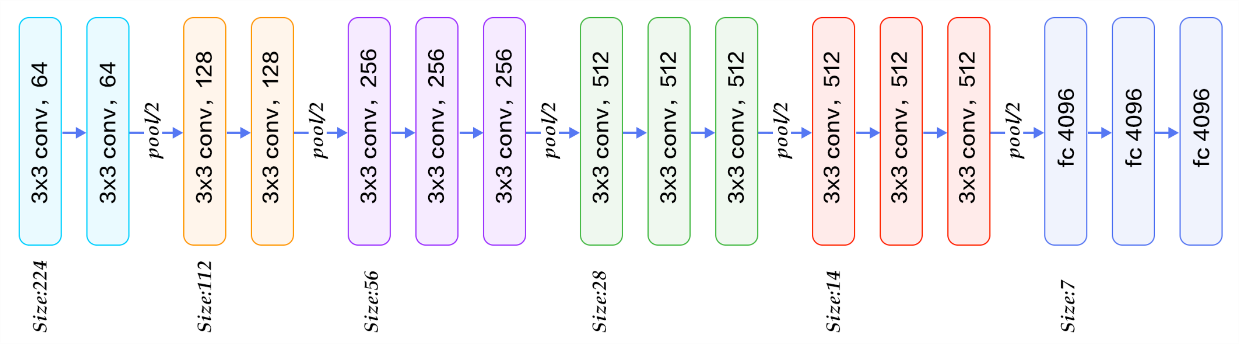
96 → # of kernels or filters
↓
our choosing

Classic Networks – VGG16

- VGG16 (also called OxfordNet) is a convolutional neural network architecture named after the Visual Geometry Group from Oxford.
- It was used to win the ILSVR (ImageNet) competition in 2014.
- Application:
  - Given image → find object name in the image
  - It can detect any one of 1000 images
  - Input image of size 224 * 224 * 3 (RGB image)
- VGG16 consists of :
  - Convolutions layers (only 3*3 size)
  - Max pooling layers (only 2*2 size)
  - Fully connected layers at end
  - total 16 layers

Convolutional Layer

- VGG16 consists of 16 layers

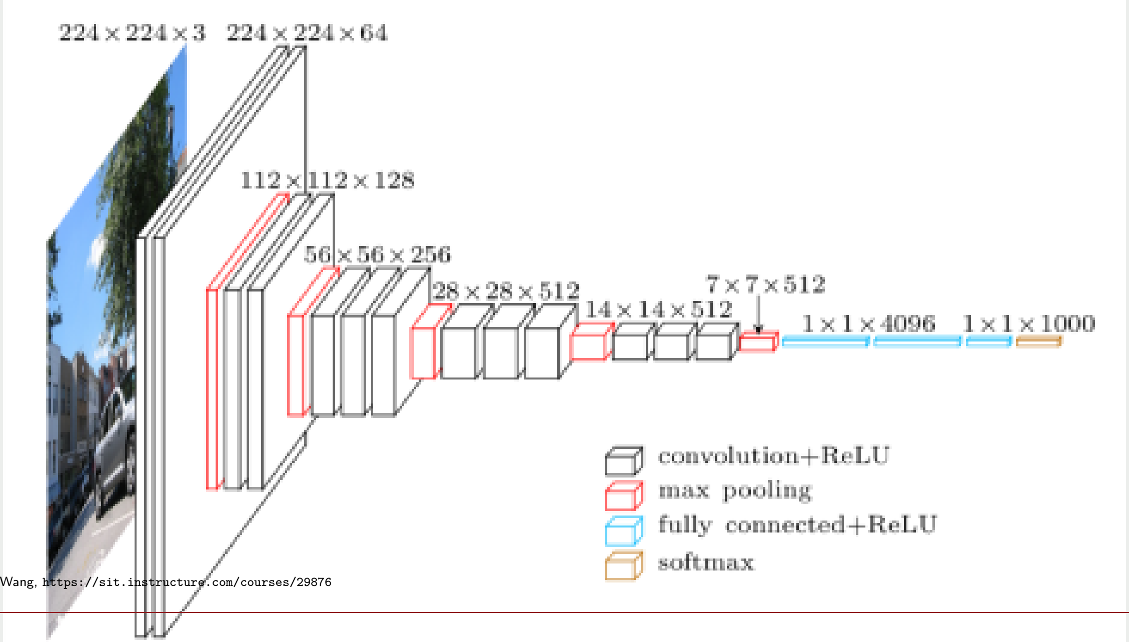


VGG-16: 16 Layers

1. Convolution using 64 filters → kernels
2. Convolution using 64 filters + Max pooling
3. Convolution using 128 filters
4. Convolution using 128 filters + Max pooling
5. Convolution using 256 filters
6. Convolution using 256 filters
7. Convolution using 256 filters + Max pooling
8. Convolution using 512 filters
9. Convolution using 512 filters
10. Convolution using 512 filters + Max pooling
11. Convolution using 512 filters
12. Convolution using 512 filters
13. Convolution using 512 filters + Max pooling
14. Fully connected with 4096 nodes
15. Fully connected with 4096 nodes
16. Output layer with Softmax activation with 1000 nodes

Convolutional Layer

- VGG16 consists of 16 layers



ResNet

❑ Problem:

When deeper networks starts converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly.

❑ Solution:

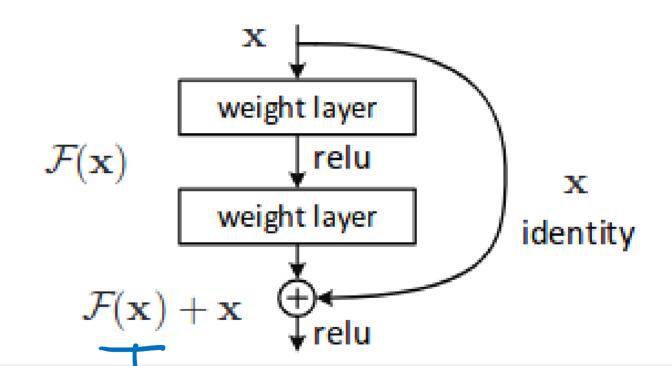
In the deeper network the additional layers better approximates the mapping than it's shallower counter part and reduces the error by a significant margin.

❑ Design:

- ❑ Use 3*3 filters mostly.
- ❑ Down sampling with CNN layers with stride 2.
- ❑ Global average pooling layer and a 1000-way fully-connected layer with Softmax in the end.

ResNet Connections

- Residual learning:


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

this won't work if the dimension of vectors f(x) & x is not same

ResNet

Observations:

- ❏ ResNet Network Converges faster compared to plain counter part of it.
- ❏ Identity vs Projection shorcuts. Very small incremental gains using projection shortcuts instead of identity in all the layers. So all ResNet blocks use only Identity shortcuts with Projections shortcuts used only when the dimensions changes.
- ❏ ResNet-34 achieved a top-5 validation error of 5.71% better than BN-inception and VGG. ResNet-152 achieves a top-5 validation error of 4.49%. An ensemble of 6 models with different depths achieves a top-5 validation error of 3.57%. Winning the 1st place in ILSVRC-2015.

Try Existing CNN Model: VGG16 with Keras

- To load VGG model is just with this

```
from keras.applications.vgg16 import VGG16
#build model
mod = VGG16()
```

- When you run this code for the first time, you will automatically be directed first to download the weights of the VGG model (550 MB). To predict using this model, you have to adjust the width and height of the input image to 224 x 224. Dont forget to convert the array of the image from integer 8-bit to floating-point 64-bit.

```
from PIL import Image as pil_image
img = pil_image.open( imgdir+‘/’+ allimgname[i])
img = img.resize((224, 224))
dataimg = np.float64(np.array(img))
dataimg = np.reshape(dataimg,(1,224,224,3))
```

Try Existing CNN Model: VGG16 with Keras

- ❑ And then change the scale of numbers in the array to make it adequate with VGG model (sort kind of normalization or standardization that fits with this VGG model)

```
dataimg = preprocess_input(dataimg)
```

- ❑ And then put it in this function

```
pred = mod.predict(dataimg)
```

- ❑ To make it outputting the result of classification, add this

```
predlabel = decode_predictions(pred)
predlabel = predlabel[0][0]
print(predlabel[1]+" ("+str(predlabel[2]*100)+")")
```