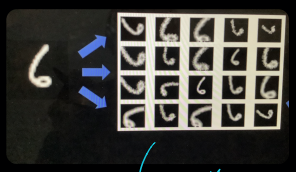
Data Augmentation:



→ this makes CNN robust against rotation, scale, cropping.

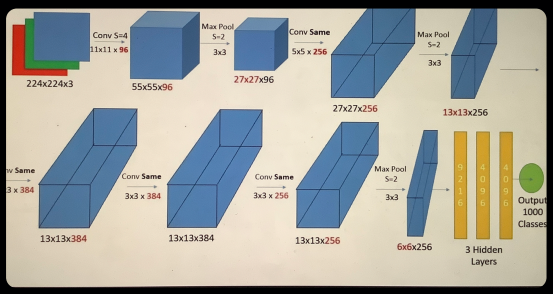
↳ also add some noise, contrast, gaps, shears, crops, etc.

small dataset

↓ data augmentation

BIG ASS DATASET

Alex Net



VGG Net

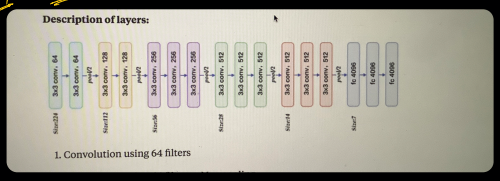
↳ Alex Net ++ → better & simplified version.

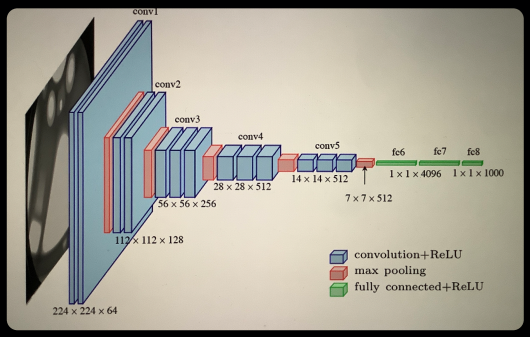
3x3
S = 1
p = same
} all convolutions

2 × 2
S = 2
} max pool

VGG(16) → # layers
VGG(19)

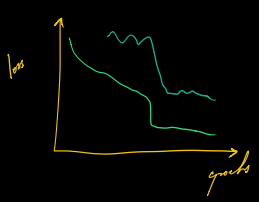
VGG Architecture



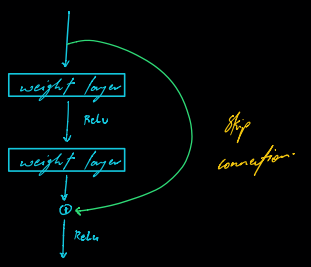


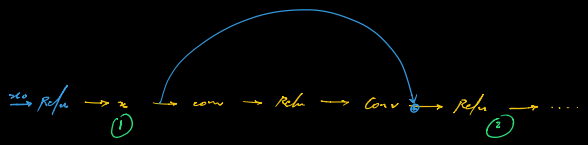

```
# Block 1
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

# Block 2
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

# Block 3
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

# Block 4
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)

# Block 5
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
```

#Resnet ⇒ Microsoft Research



56 layers
20 layers





→ Regularizers reduce useless weights to zero. → Here we are skipping layers.

$Relu(Relu(a_2)) = Relu(a_1)$

∴ Output of ② can be the same as ①

→ This guarantees that adding new layers will not hurt → regularizer will make weights of layers will be skipped.

Let's add some fucking weights then, they made

Resnet 18 34 50 101 & 152

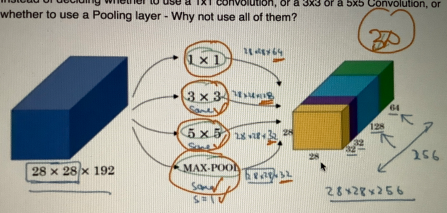
Inception Net

1x1? 3x3? 5x5? maxpool?

Why not have them all!

Motivation for Inception Network

Instead of deciding whether to use a 1x1 convolution, or a 3x3 or a 5x5 Convolution, or whether to use a Pooling layer - Why not use all of them?

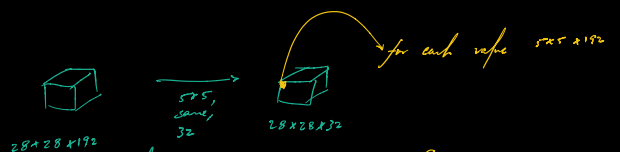


In the example above, all the filters are applied to the input to generate a stacked outpu, whioch contains the output of each filter stacked on top of each other. The Padding is kept at 'same' to ensure that the output from all the filters are of the same size.

Disadvantage: Huge memory cost

Solving the problem of memory cost

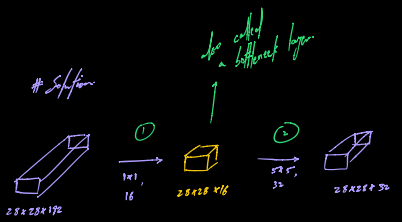
→ No choosing of kernel size.



So for 28×28×32 values:

× 5×5×192

= 120 million !!



first compress 192 channels to 16 → then apply 5×5×16 kernels

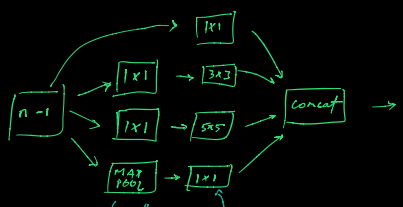
① + ② [28×28×16 × 1×1×192] + [28×28×32 × 5×5×16]

= 12.4 million.

This is a lot more manageable than 120M

(→ reduced by factor 10!

Inception Module



↳ still 192 channels, that's why 1x1 to reduce channel size.

The optimization from 150 → 12.4 is fucking awesome. #Google Engineers

Transfer Learning

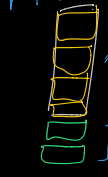
→ Use pretrained models instead of using random weights.

if I'm good at body building → I'll be good at strongman in no time

if [M] trained on [dataset A]

we can use [M] on [dataset B] given that A & B are similar.

TASK A & TASK B have to be similar. → using vgg16 will not be a great help with radiology tasks for example.



or make the entire thing trainable

Have vgg16 might act act as a feature extractor for the green model.

CASE 1 : Task similar to Imagenet but low data.

SOLN : Train last few layers, freeze the rest.

CASE 2 : Similar to Imagenet & large dataset.

SOLN : Finetune the whole thing → small learning rates.

CASE 3 : Small dataset & not similar to Imagenet.

SOLN : Use output of first couple of layers as features & flatten.

Use same logic, this thing depends on the application we're working on.