

ANCHORMEN
data activators

COMPUTER VISION

Raoul Grouls

28-MAY-21



TABLE OF CONTENT

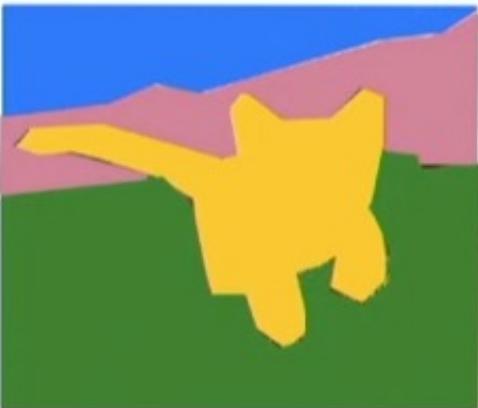
- Motivation
- Components: convolutions and pooling (basics)
- Architectures (advanced)
- Transfer learning (default practice)

COMPUTER VISION TASKS

Image
Classification



Semantic
Segmentation



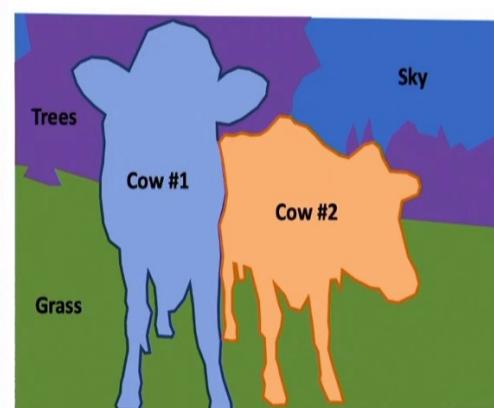
Object
Detection



Instance
Segmentation



Panoptic
Segmentation



Classify image

Classify every pixel into a category

Classify multiple objects on an image and localize them

Classify multiple objects on an image, localize them and make pixel-wise object mask

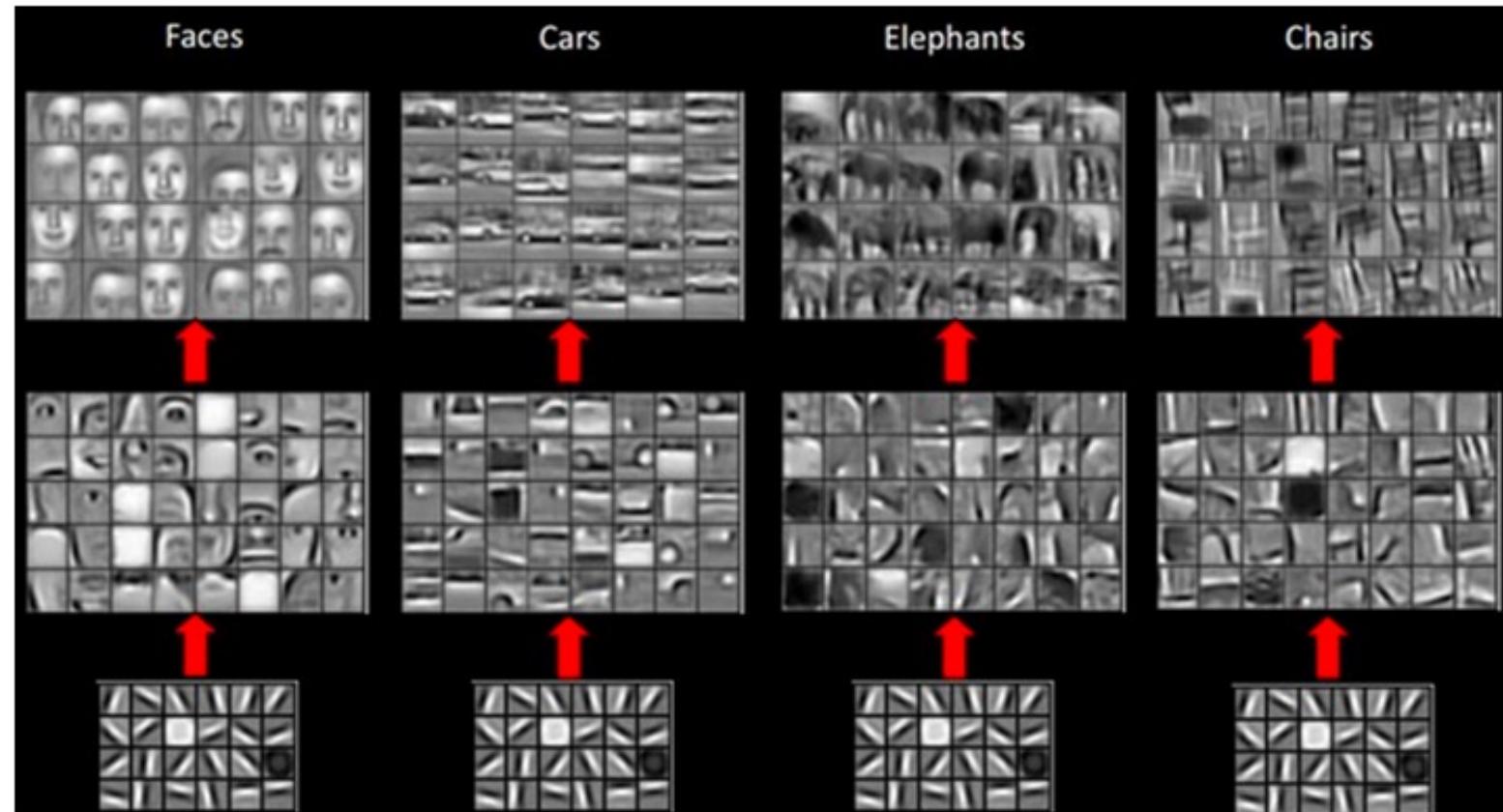
Instance segmentation for “things” and semantic segmentation for “stuff”

MOTIVATION FOR CONVOLUTIONAL LAYER

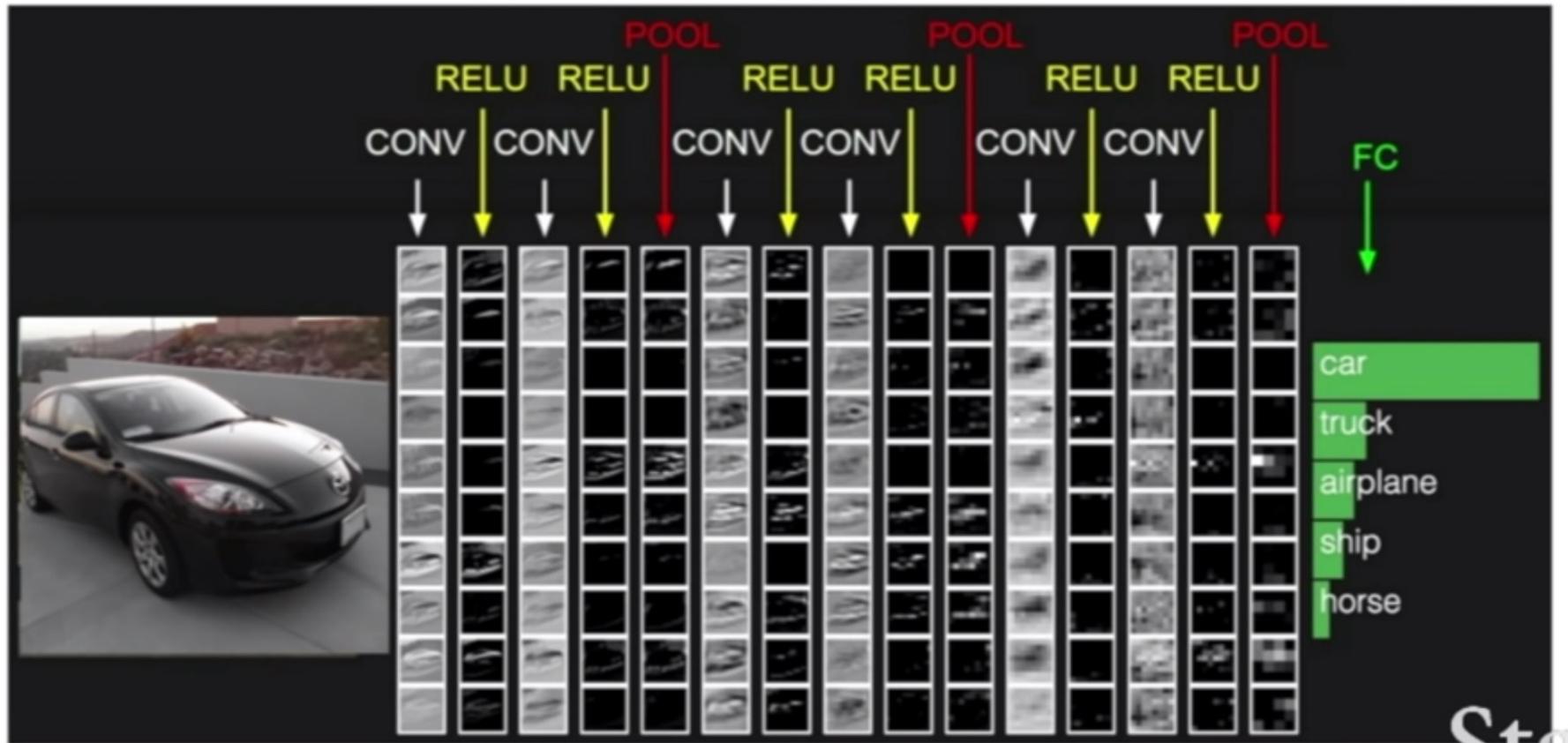
- Preserve spatial structure
 - We have been flattening the image (eg 28x28) into one vector (eg 784). With this, we destroy a lot of spatial information.
 - Idea: Don't flatten the image to one vector, but process the 'context' of a pixel.
- Advantages
 - Nearby pixels will stay nearby
 - We will generate "higher level abstractions" of groups of pixels. E.g. "this pixel is black" becomes "these pixels are part of a vertical line segment".
 - Translational invariance. If you shift the position of an object, it still means the same thing, a cat in the right corner of the image is also a cat in the left corner

MORE AND MORE COMPLEX FEATURES

- First layers more simple features
- Latter layers more complex features



BASIC CONVOLUTIONAL NETWORK STRUCTURE

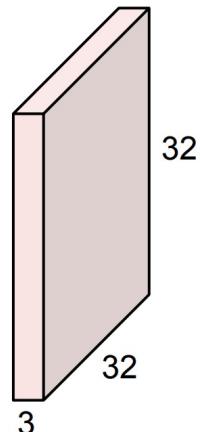


CONVOLUTIONAL NET COMPONENTS

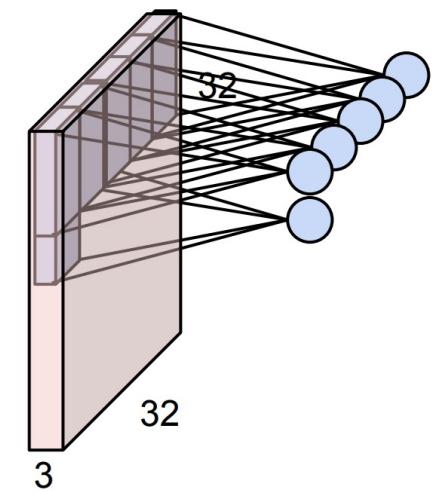
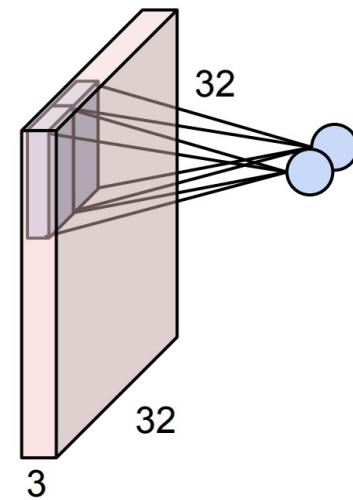
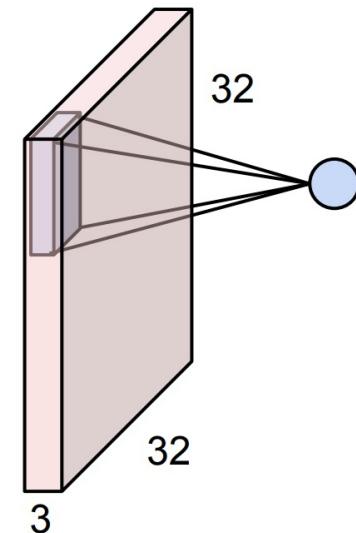
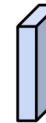
CONVOLUTIONAL LAYER

- Take a filter
- Slide over every position of the input

32x32x3 image



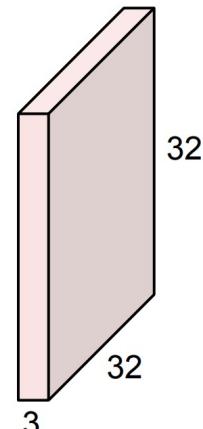
5x5x3 filter



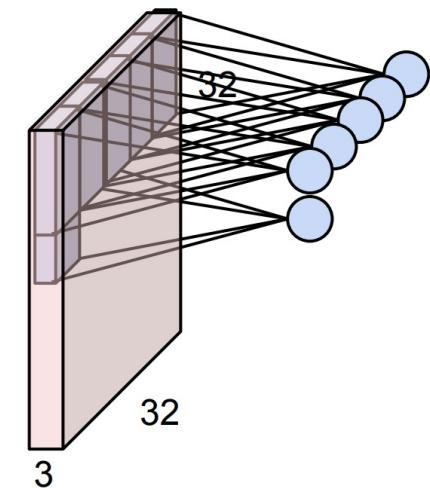
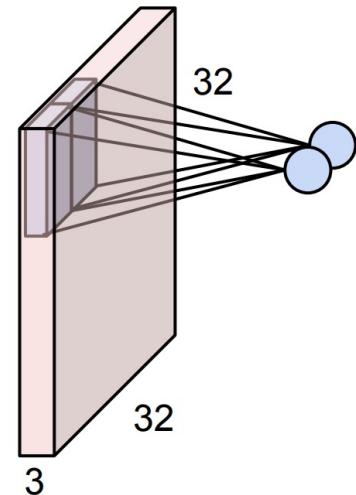
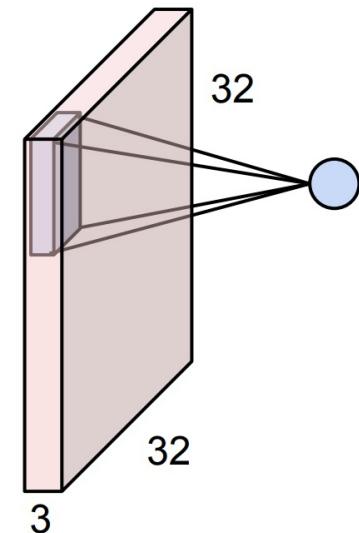
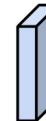
CONVOLUTIONAL LAYER

- Calculate the dot product between the filter and input segment the filter covers
- Arrange result of the dot products in a 2D array, yielding the output activation map
 - The activations in the activation map should reflect the positions of the filter, when the dot product was calculated

32x32x3 image



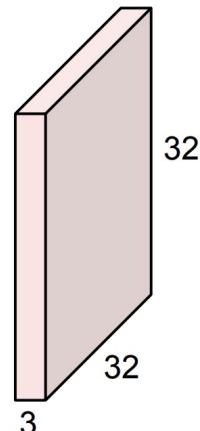
5x5x3 filter



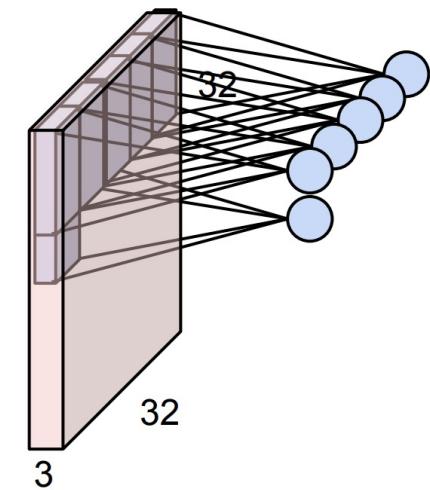
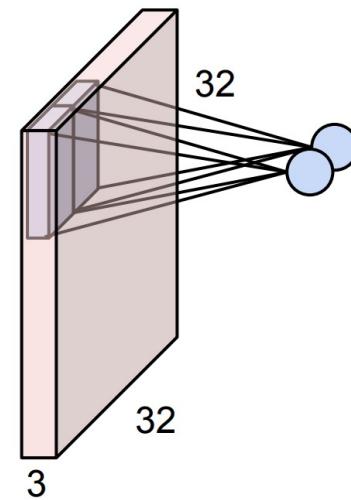
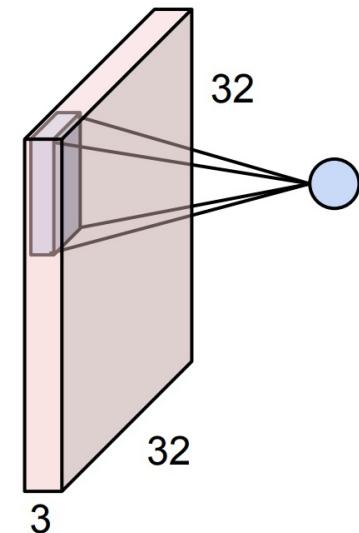
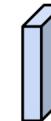
CONVOLUTIONAL LAYER

- The filter goes through the entire depth of the input
- Parameters in filter: (width, height, channel number)
- In case of first hidden layer, the channel number is 3, representing the RGB colours

32x32x3 image



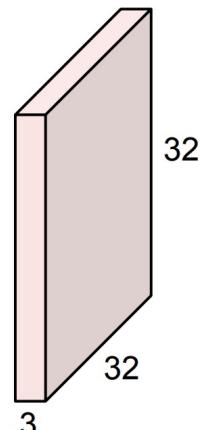
5x5x3 filter



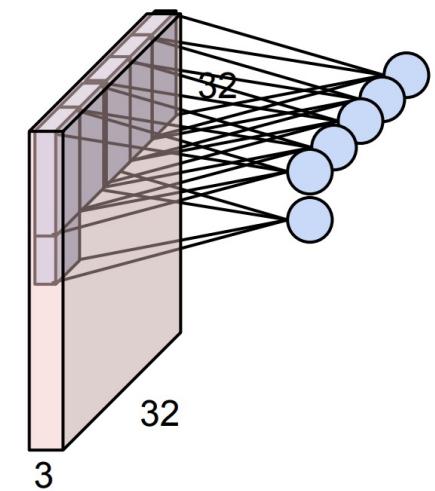
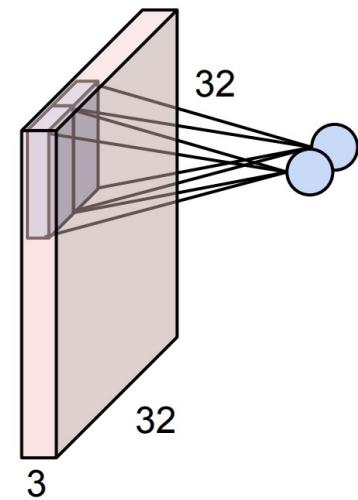
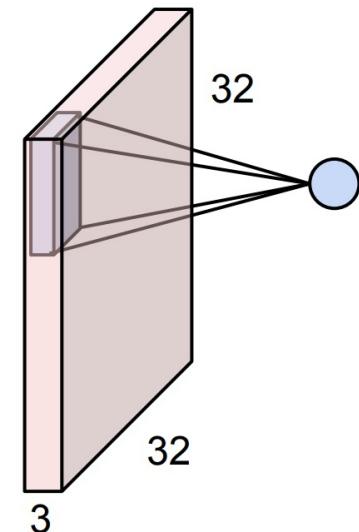
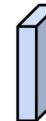
CONVOLUTIONAL LAYER

- Later on, the channel number represents the amount of filters that produced feature maps.
- So, we could end up with small feature maps (eg 5x5 pixels big) but we might have 512 of those.

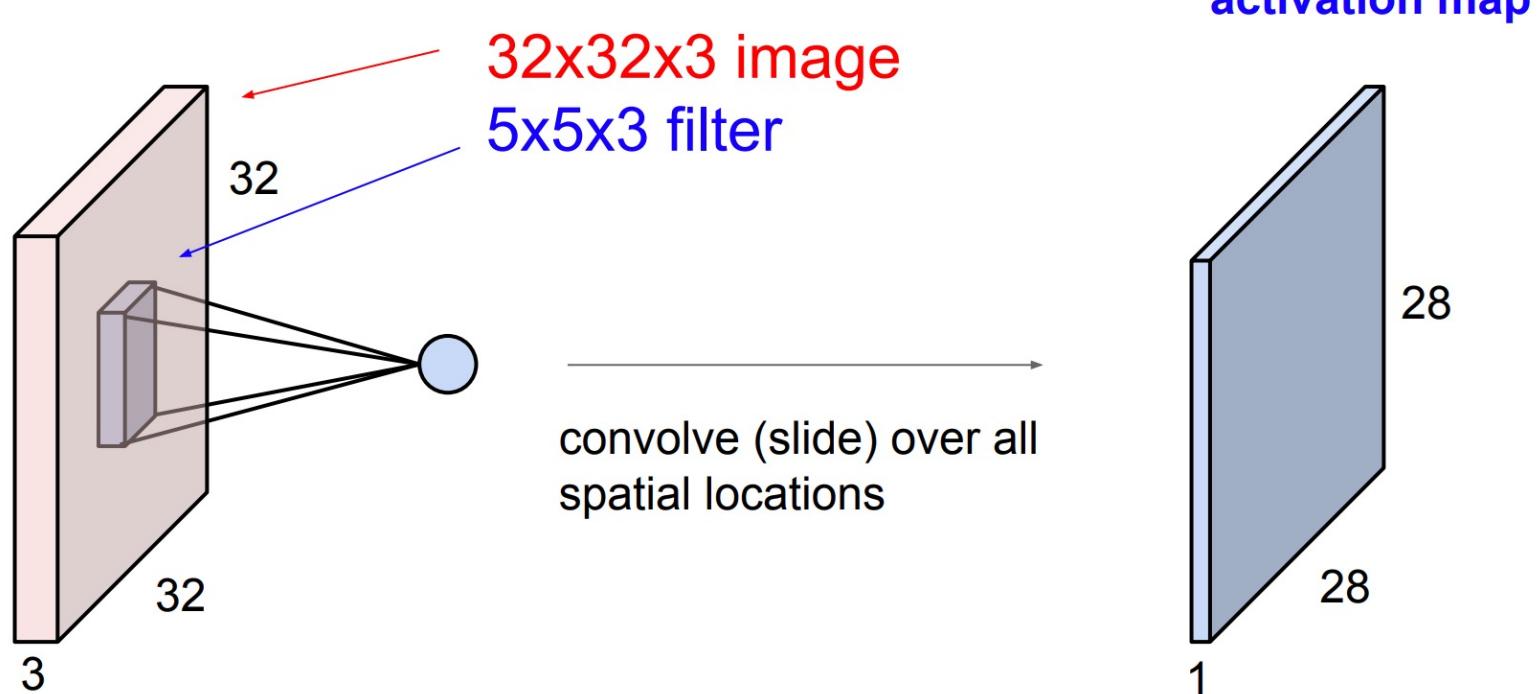
32x32x3 image



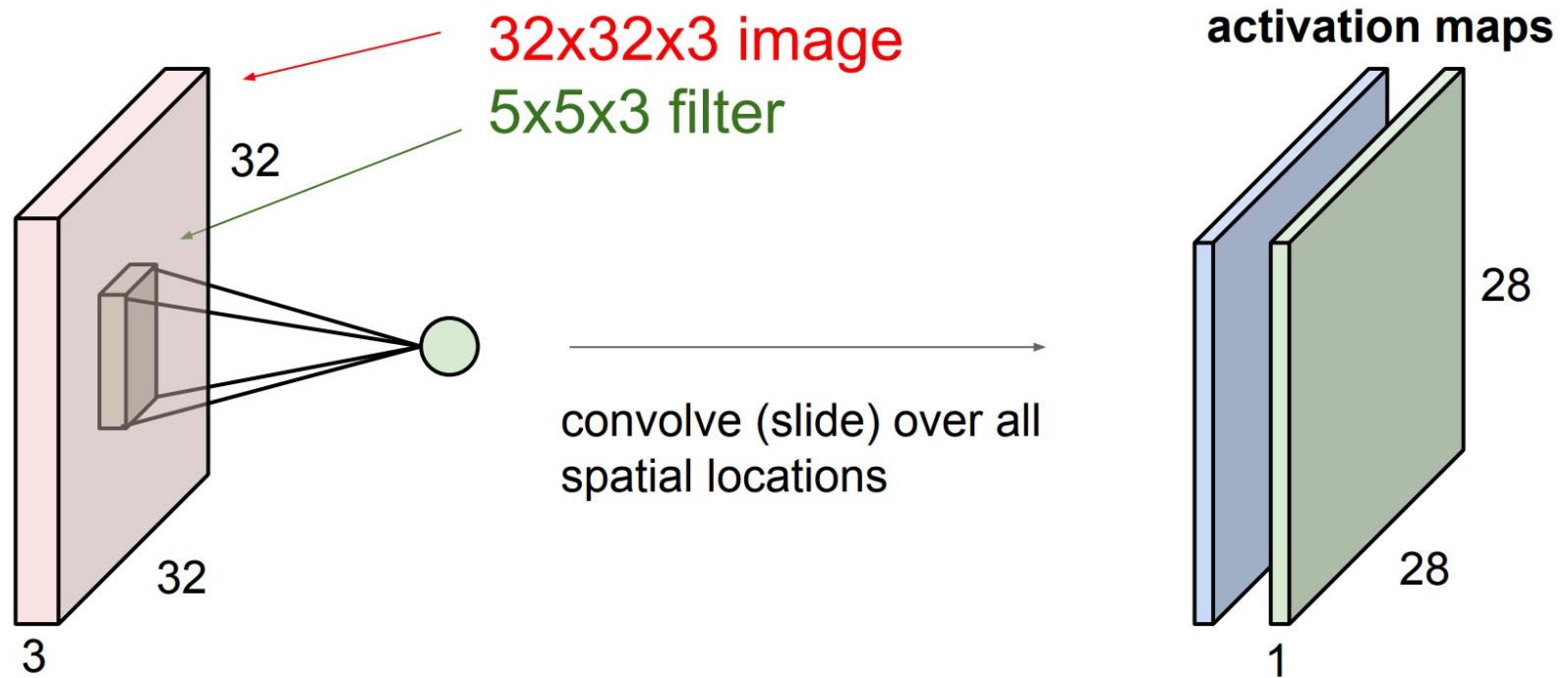
5x5x3 filter



CONVOLUTIONAL LAYER

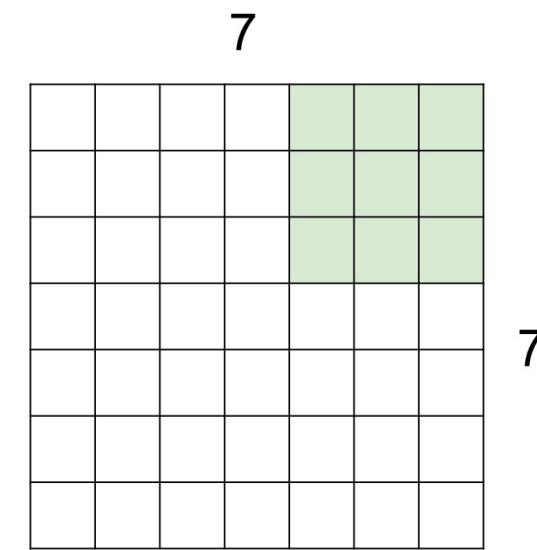
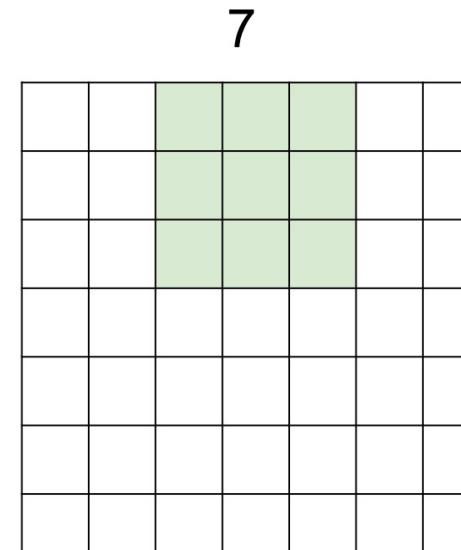
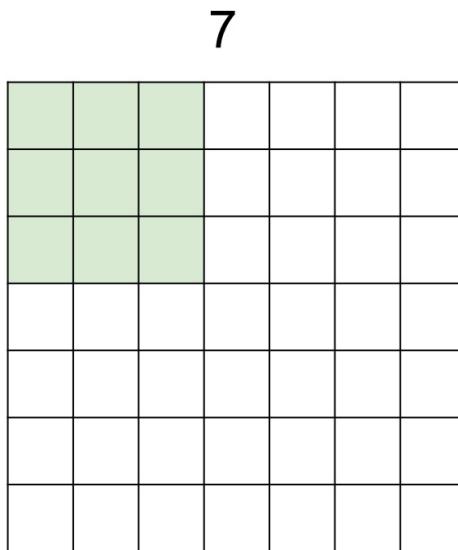


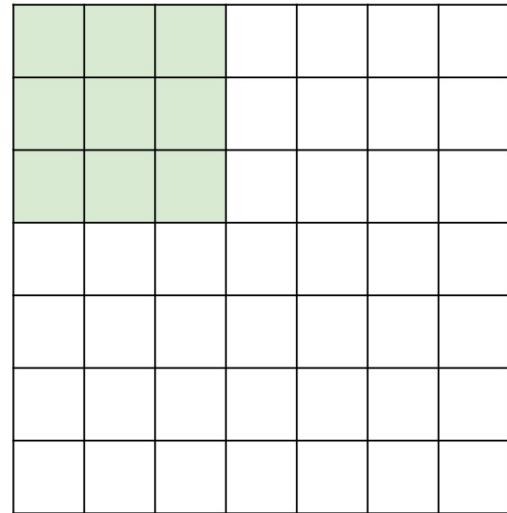
CONVOLUTIONAL LAYER



STRIDE

- Stride: how many pixels to step when sliding the filter
- Shrinks the output dimensionality, if stride > 1
 - If stride is 2, dimensionality is approx. halved

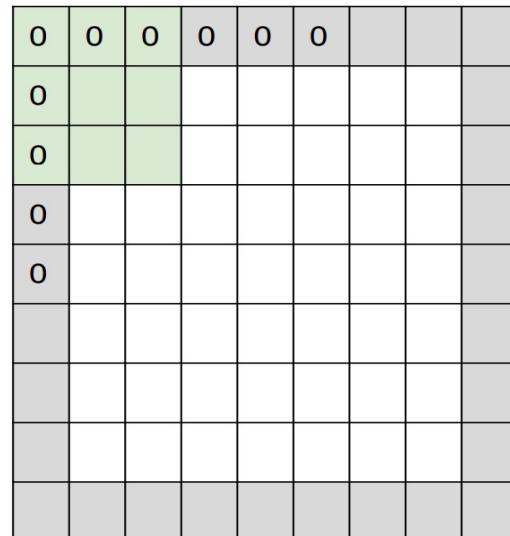




7

PADDING

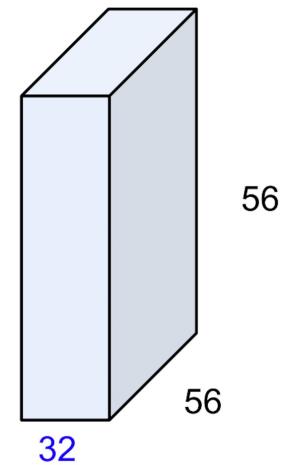
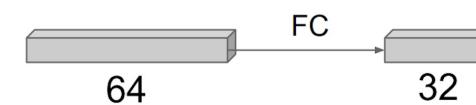
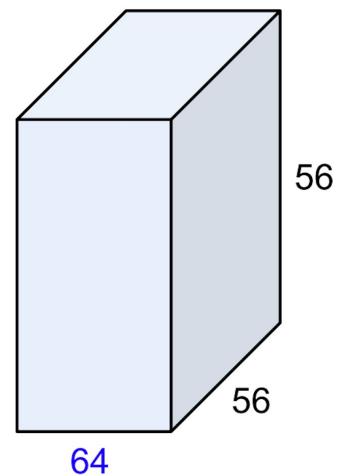
- No padding
 - Slide as far as we can, ignore the last pixels if the filter does not fit.
 - Output size will be smaller than input size.
- Padding
 - Filter can slide out of the image.
 - Output size will be the same as input size.
 - Motivation: for a deep network the activation layer could shrink very quickly



1X1 CONVOLUTIONAL LAYER

These have three advantages:

- While they can not capture spatial patterns, they can capture patterns along the **depth dimension**
- They are often configured to output fewer activation maps than their input, thus serving as “bottleneck” layers. This **reduces dimensionality** and forces the model to preserve the essential information and throw away less important information.
- Each 1x1 layer has an activation. A pair of (1x1) and (3x3) convolutions will be able to capture **non-linearities**.

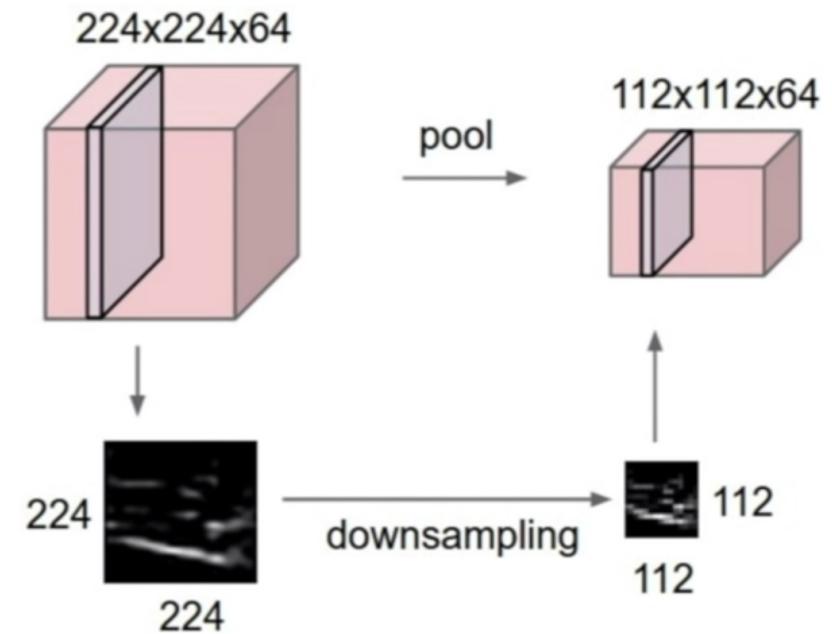


CONVOLUTIONAL LAYER DESIGN CHOICES

- Number of filters
- Filter spatial size
- Stride
- Padding

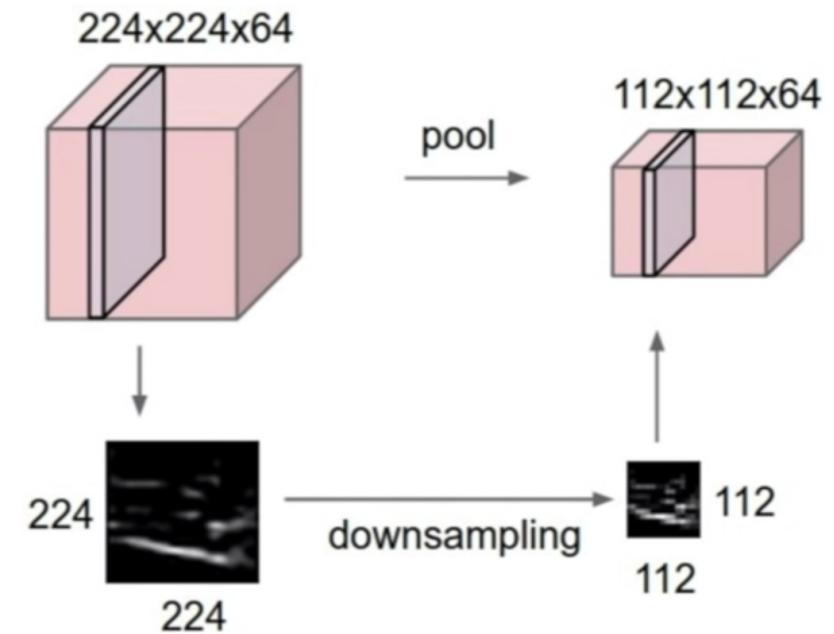
POOLING

- To downsample the input
- Swipes through the input and performs the aggregating operation
- Does this in every channel independently
 - Doesn't effect depth dimensionality, unlike the convolutional layer



POOLING

- Pooling, or convolution with stride 2?
 - Contains no learnable parameters
 - Introduces some translational invariance
 - But sometimes conv with stride > 1 works better, and some architectures remove maxpooling altogether in favor of stride=2



POOLING TYPES

- Max
 - Intuition: If there was any high feature value in the area, we only care about that, not the other values
 - More translational invariance
- Average
 - Intuition: Makes a more smooth downsampling
 - Keeps the spatial structure more
- Usually max is preferred over average pooling

Max Pooling

29	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

100	184
12	45

Average Pooling

31	15	28	184
0	100	70	38
12	12	7	2
12	12	45	6

2 x 2
pool size

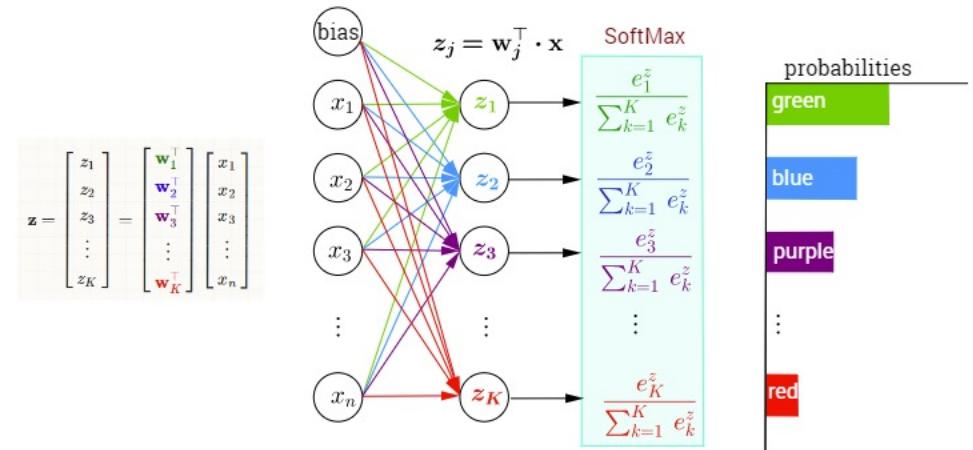
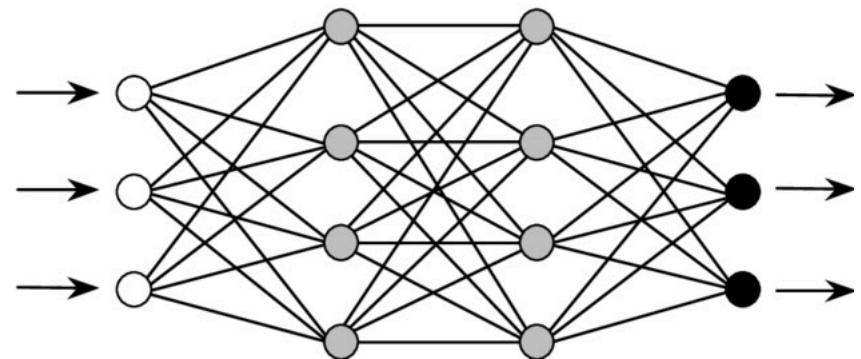
36	80
12	15

POOLING DESIGN CHOICES

- Pool size
- Stride
- Type
 - Max
 - Average
- Default settings
 - Max, pool_size=(2,2), Stride=(2,2), padding="valid"

FULLY CONNECTED LAYERS, SOFTMAX

- Flatten the last activation maps into a single 1D vector
- The output layers should either use a softmax for classification, or use no activation at all.
- Softmax turns the logits (ranging from $-\infty$ to $+\infty$) into a probability distribution (positive numbers that add up to 1 over all the output neurons)
- If you don't use a softmax, you should set `from_logits=True` in your loss function.

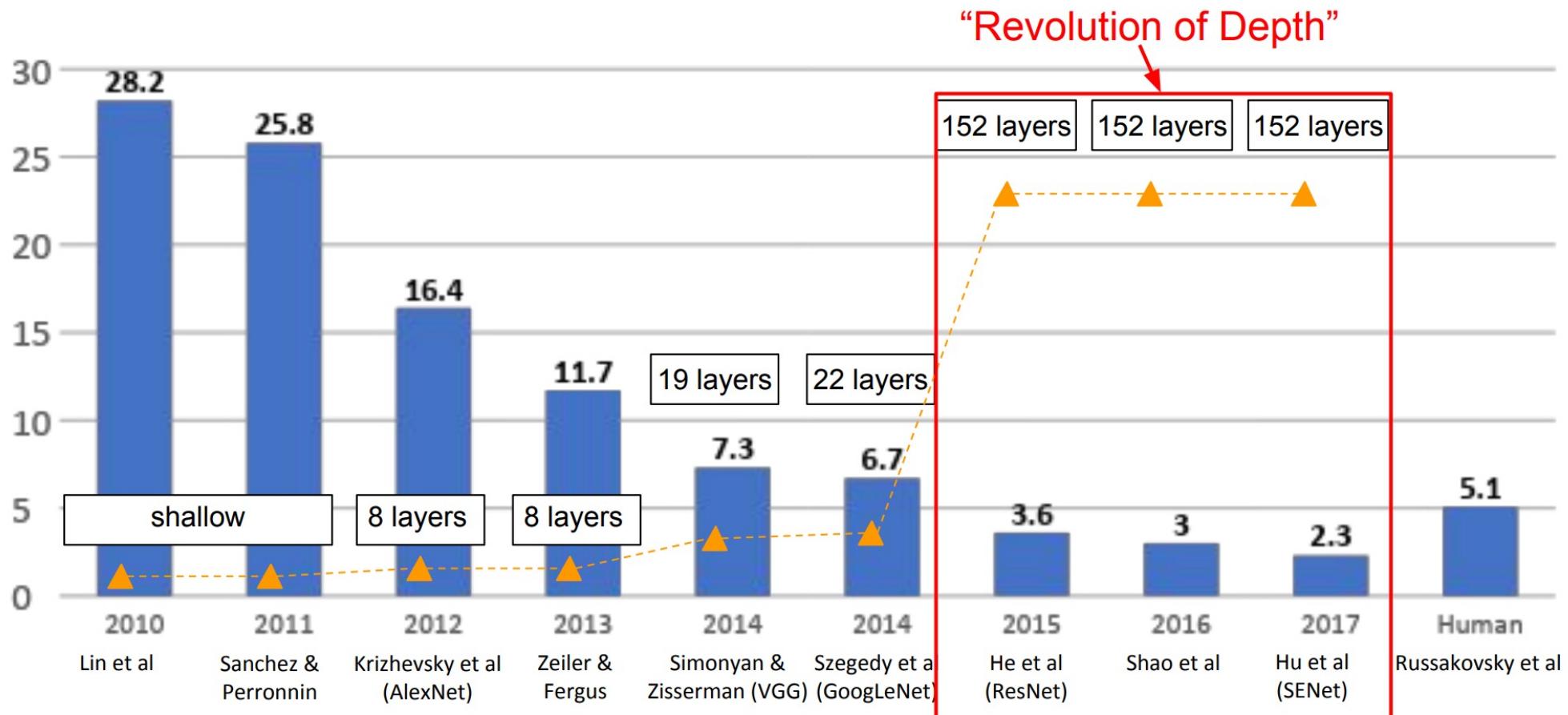


$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

ARCHITECTURES

CNN LAYER NUMBERS

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



LENET AND ALEXNET

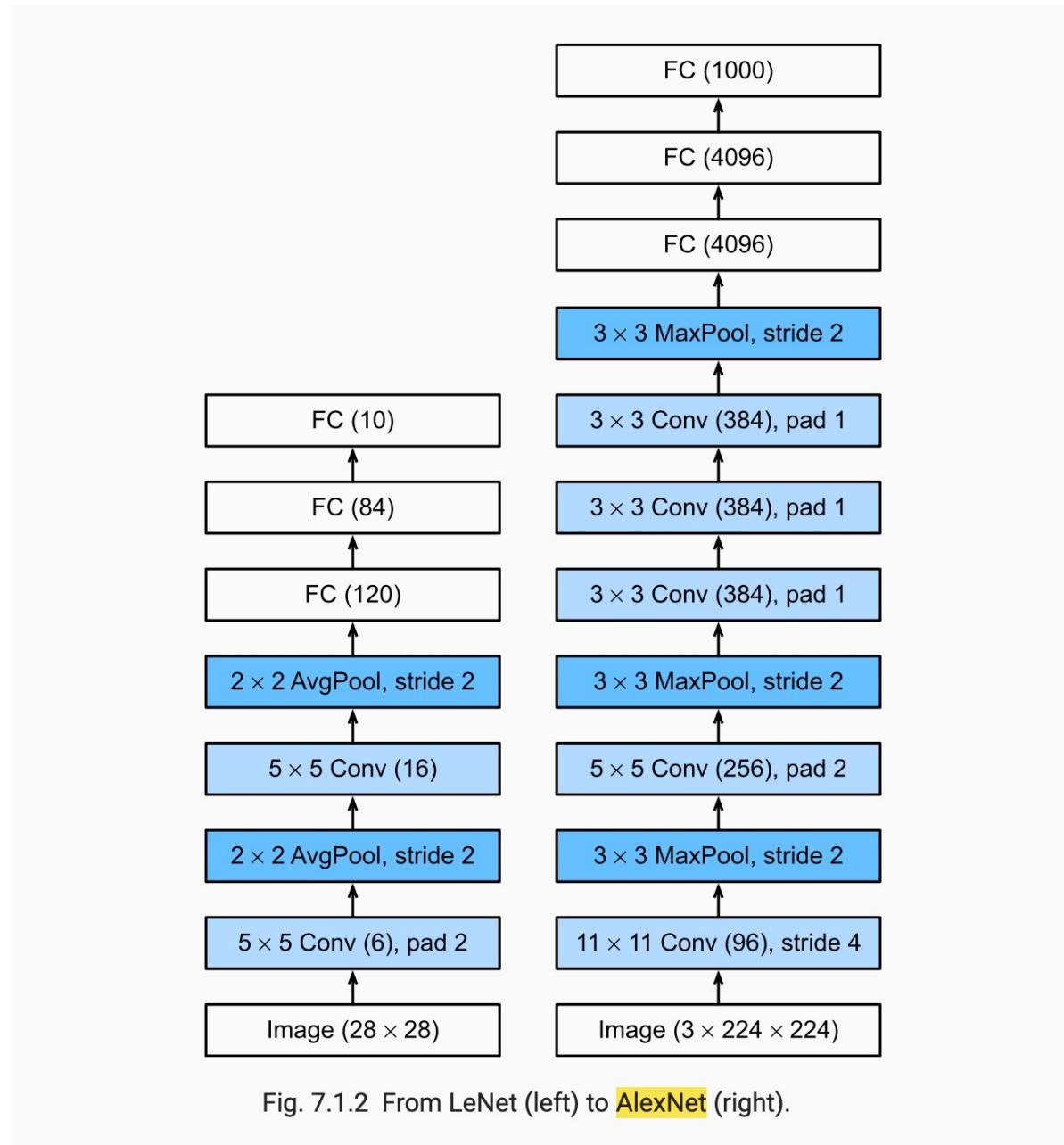
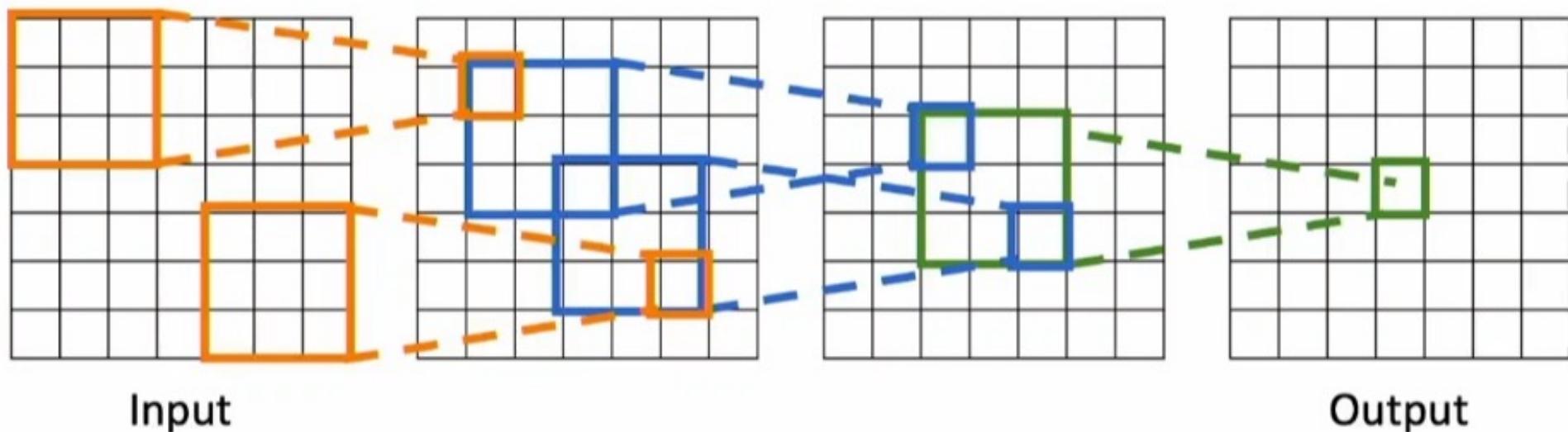


Fig. 7.1.2 From LeNet (left) to AlexNet (right).

EFFECTIVE RECEPTIVE FIELD

- Effective receptive field is the total area of inputs, a set of layers is looking at combined, to calculate one element of the activation map



The effective receptive field of 3 stacked 3x3 filters is 7x7

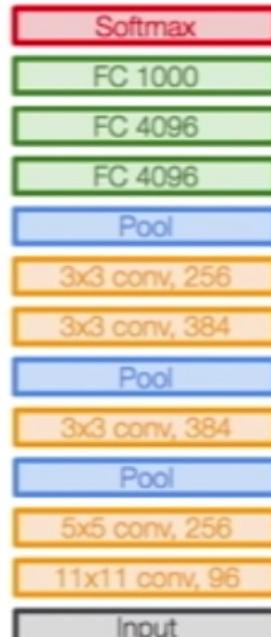
VGG

Design principals

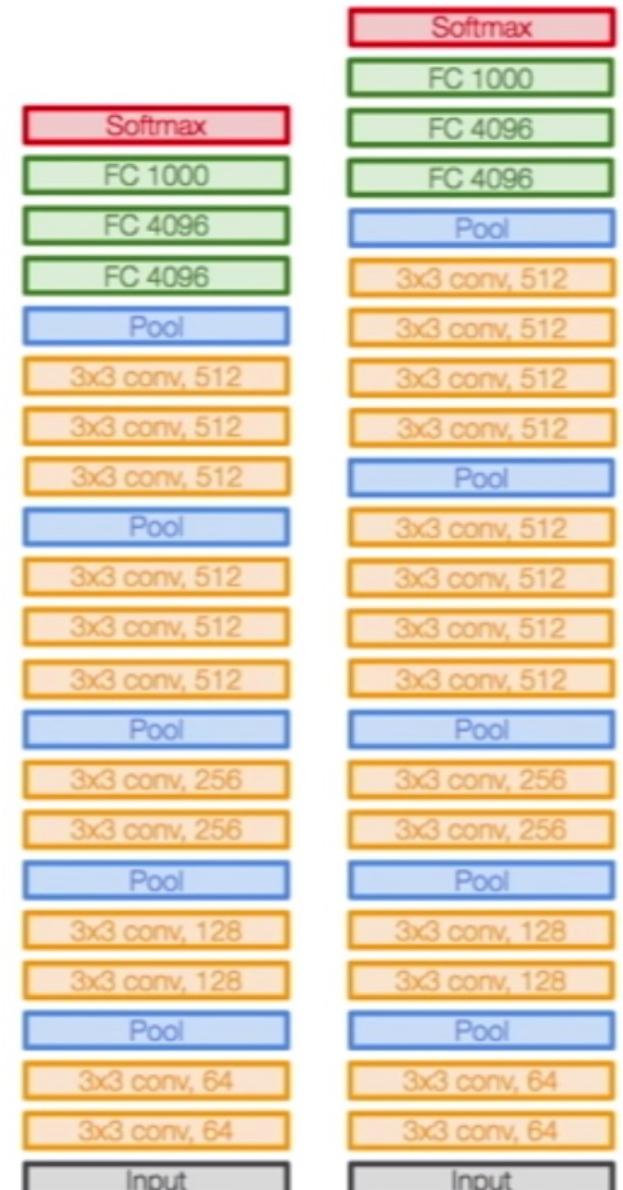
- Smaller filters, but deeper
- Double channel number after each pool
- deeper, more nonlinearities

Double channel number after each pool

- Keeps constant compute over depth (pool reduces spatial resolution)



AlexNet



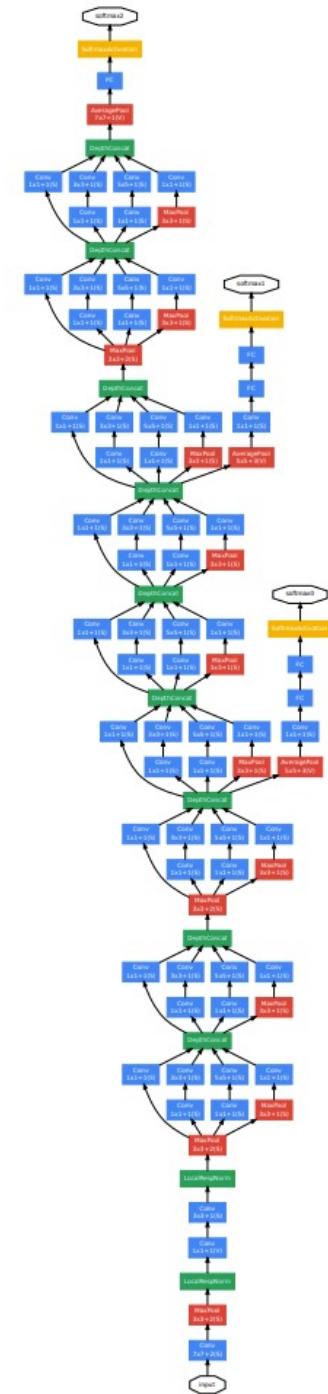
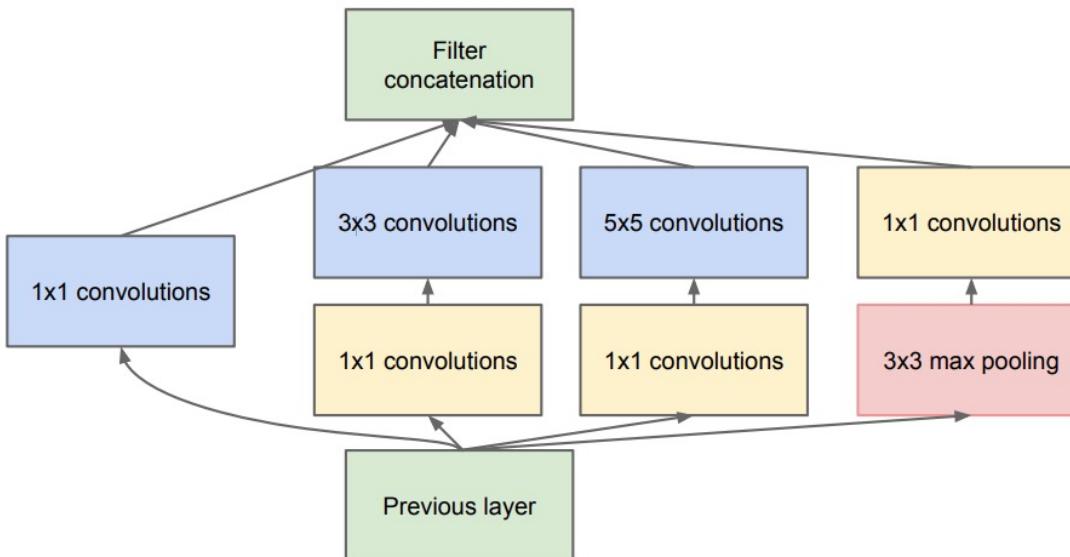
VGG16

VGG19

GOOGLENET

Inception module

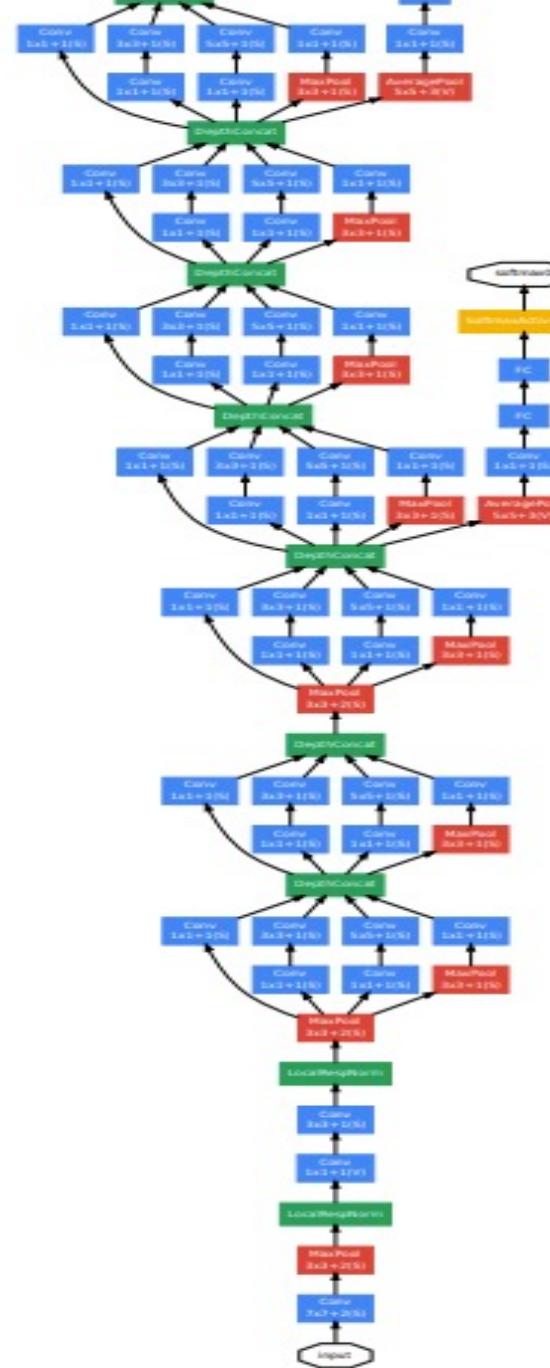
- Parallel filters
- Bottleneck layers



GOOGLENET

Some properties

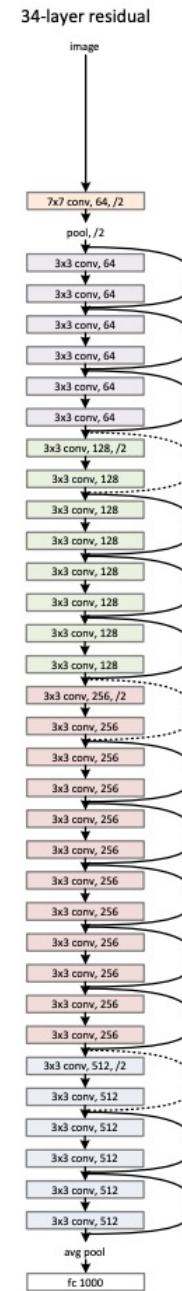
- Focuses on efficiency
- 22 layers
- Only 5M parameters
 - Alex net had 16M parameters, but GoogleNet is deeper



RESNET

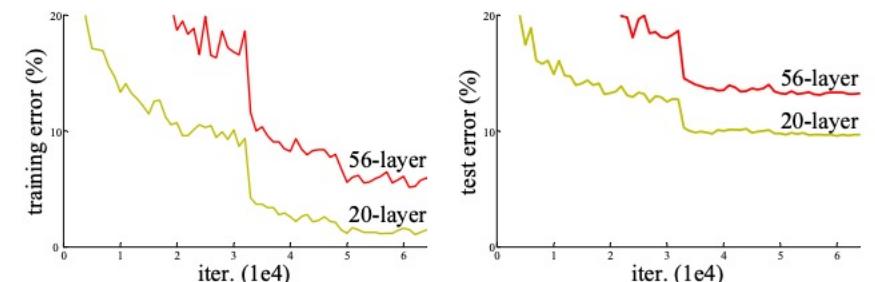
2 important discoveries, made deep networks trainable

- Batch normalization
- Residual block



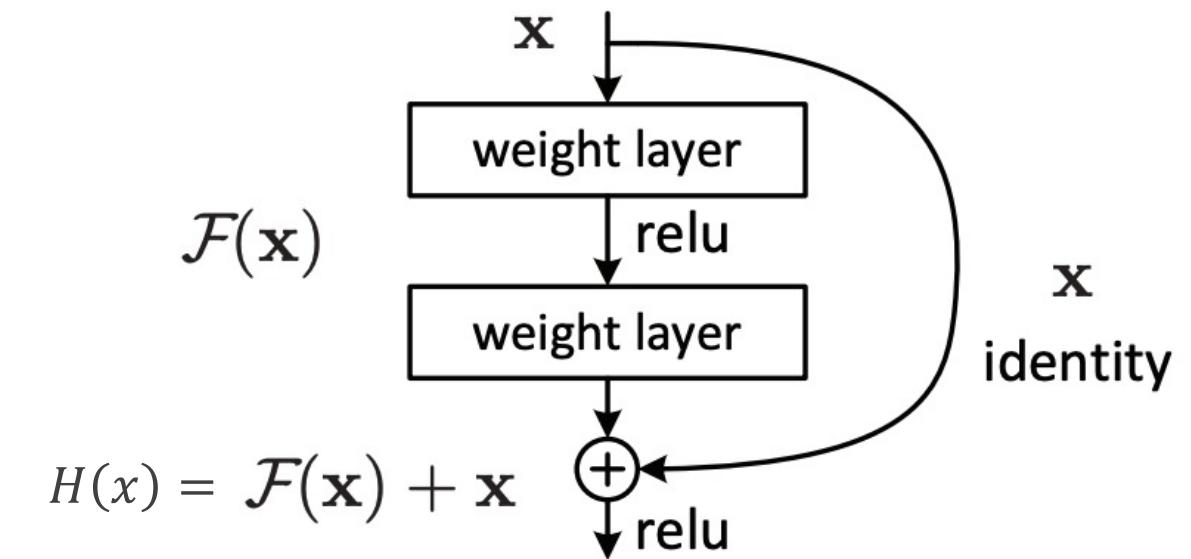
RESNET

- What they noticed: a deeper net (56 layers) was performing worse than a shallower net (20 layers), but not because of overfitting (because it was performing worse even on the training set)
 - Hypothesis: it is because deeper net is a harder optimization problem
- Idea: don't learn the mapping, but learn the residual
 - Residual: what you need to change from the previous representation to achieve the mapping



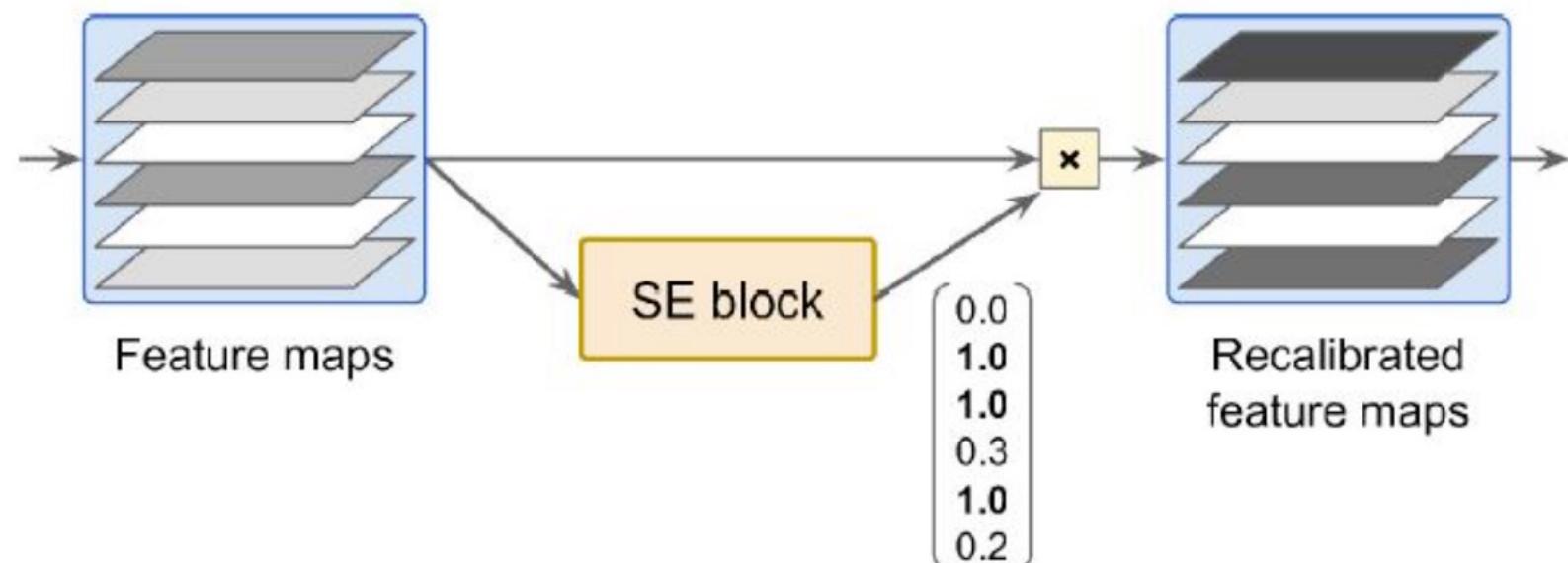
RESNET

- Learn the residual $F(x)$ of the mapping, instead of the complete mapping $H(x)$
- Desirable properties of skip connection :
 - Lets gradient flow back to earlier layers more easily
 - Residual block can easily learn the identity transformation
 - If all weights are zero -> identity transformation



SENET – SQUEEZE-AND-EXCITE NETWORK

The SE block analyzes which features of the output are most likely to be activated together, and adjusts the feature map according to the analysis results, as shown in the figure:

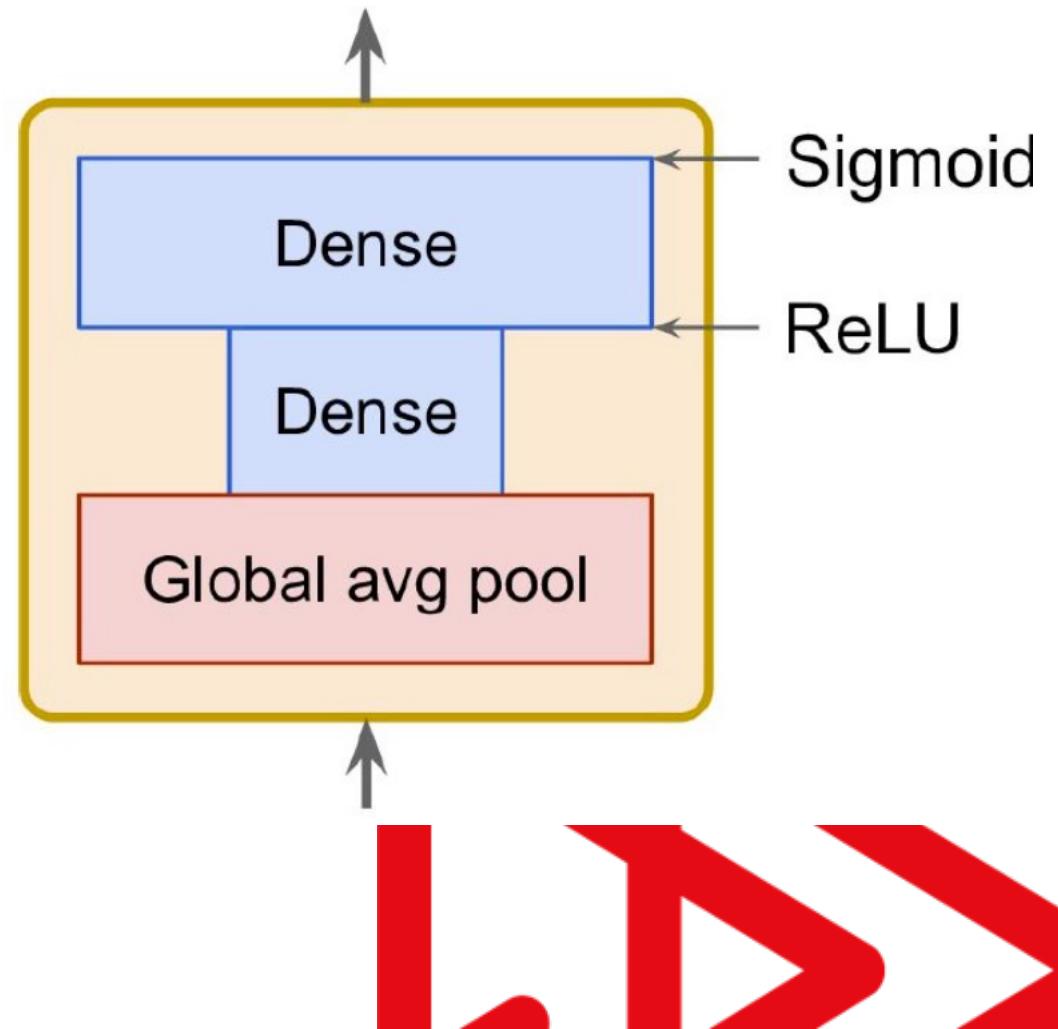


SENET

Squeeze-and-excite network

- **Global average pooling layer:** Calculate the average value of each feature map, and output x values for x feature maps, e.g. 256
- **Dense layer with Relu (squeeze):** Reduces the features by a fraction (e.g. one-sixteenth). E.g. 256 feature maps are compressed to 16.
- **Dense layer with Sigmoid (excite):** The number of neurons is the same as the number of feature maps in the global average pooling layer, and the output is a vector with a value between 0 and 1

The feature map before the global average pooling layer is multiplied by the output vector. Result: items that often occur together , will be enhanced. E.g: usually you see an eye and a nose together. If you doubt about the nose (because of noise), it probably is a nose.



TRANSFER LEARNING

TRANSFER LEARNING

- Sometimes you overfit, because you don't have enough data
- Idea: Pretrain model on a large generic dataset, then refine on your problem specific data set
- Motivation: the features that the model learned from pretraining will be generic, and useful for your data set

TRANSFER LEARNING

Strategies

- Pretrain and freeze
 - Freeze weights, except from last layer(s)
 - Train last layer(s) with your specific dataset
- Pretrain and refine (can be a second step)
 - Don't freeze weights, just refine with your data
 - Keep the **learning rate low**, for refining training

The less data you have the smaller part of the model to refine (for very few data, only the last layer)

Transfer learning (pretraining) nowadays is a default, not an exception

DATA AUGMENTATION

- Idea: randomly transform the image
- Reasoning:
 - Free way of generating new training data
 - Acts as a regularization
 - NN doesn't see the same pixel combination over and over again
- Transformations
 - Flip
 - Scale
 - Crops
 - Change contrast, brightness
 - Colour jittering
 - Salt and paper
 - Cutout (cover parts of the image with black rectangle)

