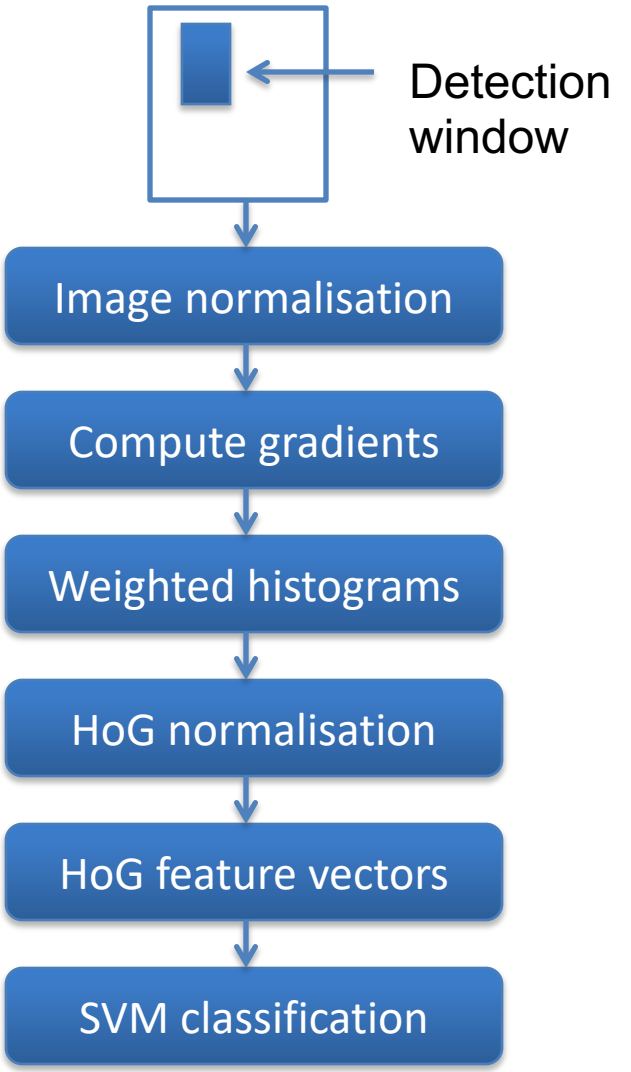
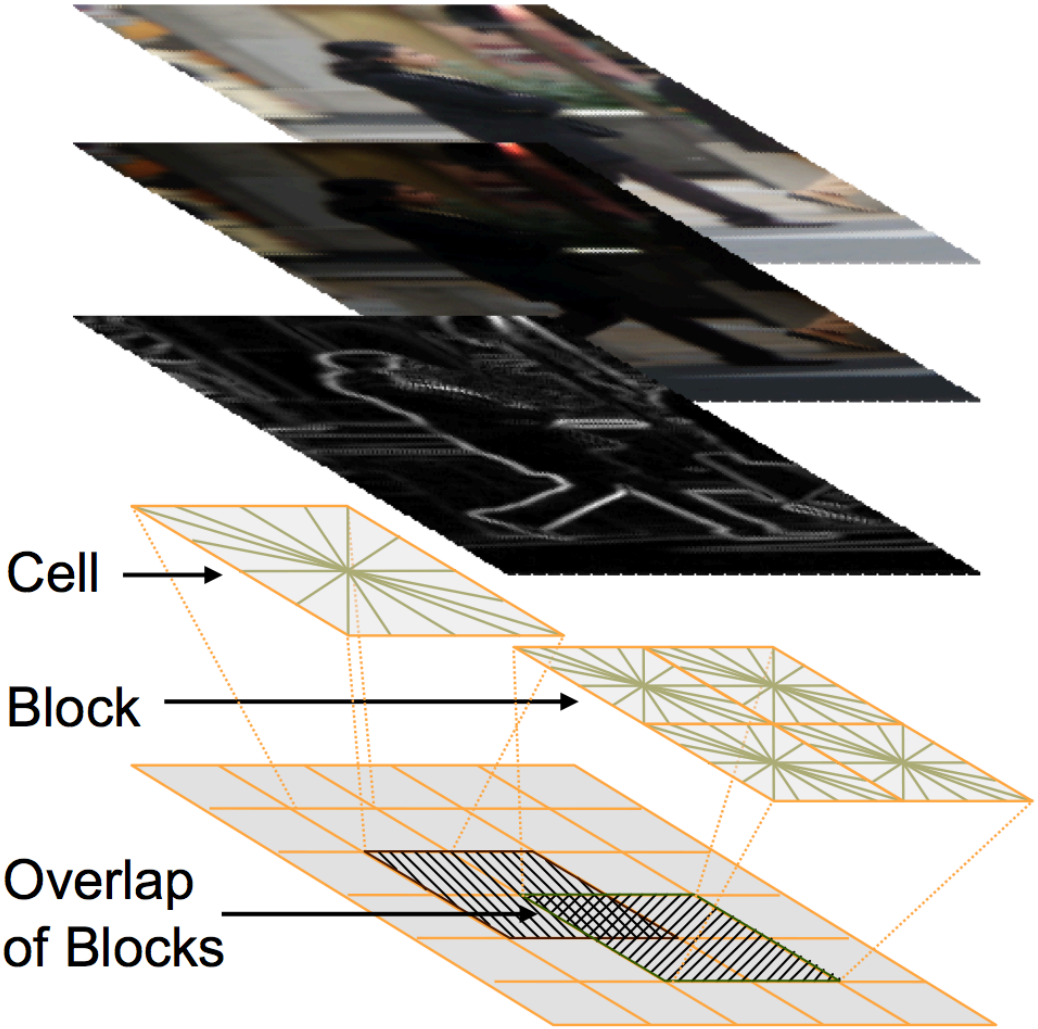
CSCM77
Convolutional Neural Network

Prof. Xianghua Xie
x.xie@swansea.ac.uk
http://csvision.swan.ac.uk

Histogram of Oriented Gradients

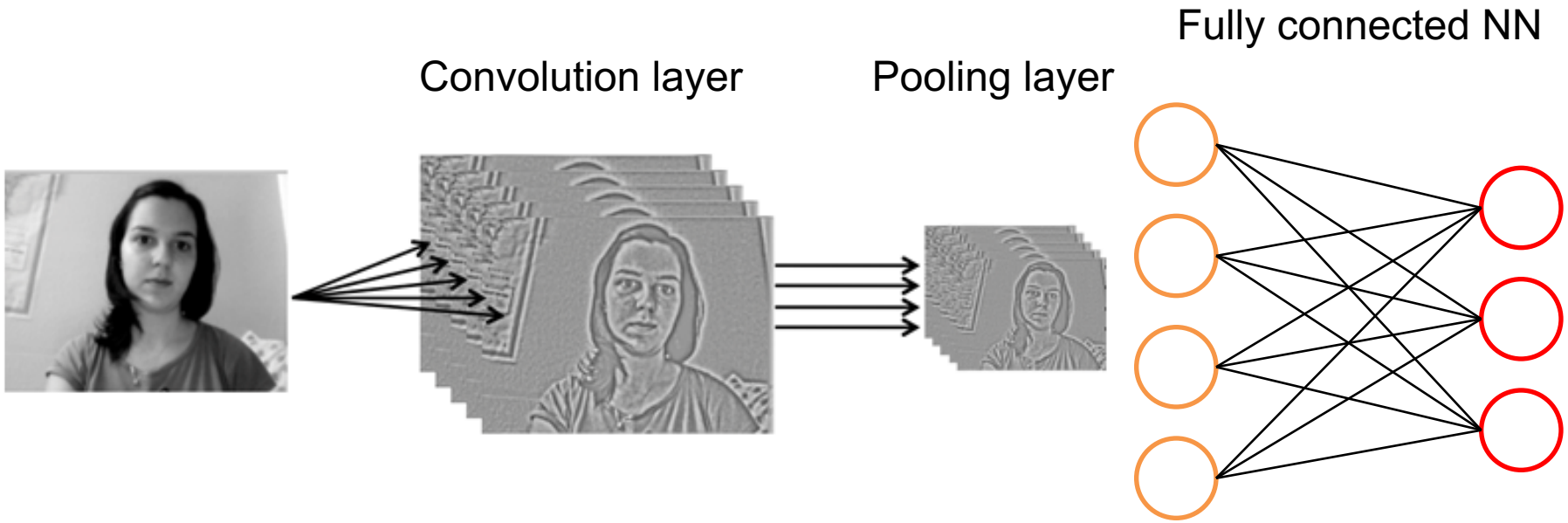
- Overview





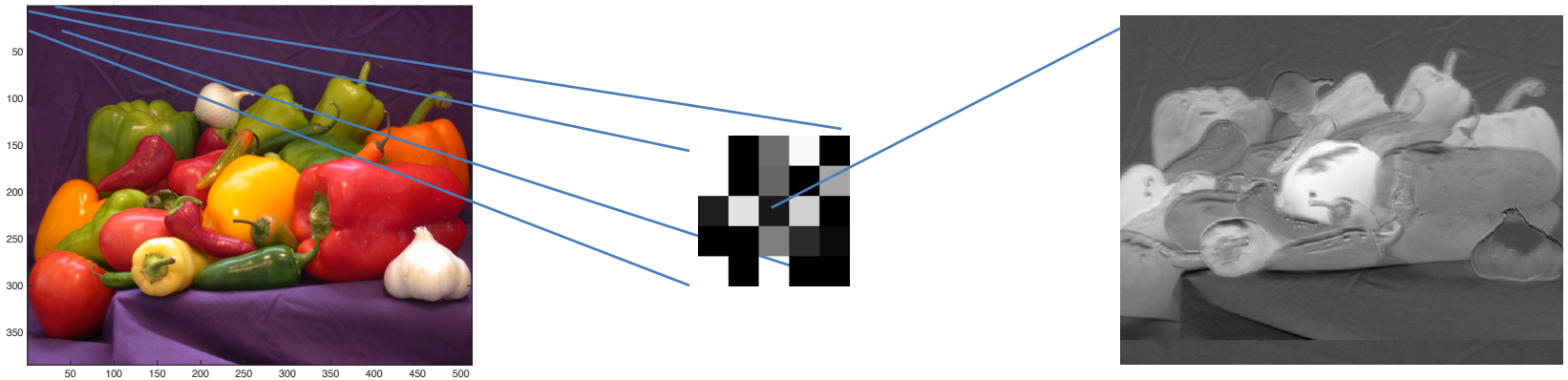
CNN: Convolutional Neural Network

- CNN
  - Filter kernel identifies features within a receptive region of input
  - Convolution layer generates feature map of input layer
  - Pooling generalises feature maps



CNN: Convolutional Neural Network

- NNs disadvantages:
  - non-localised features,
  - many parameters to learn,
  - doesn't scale well (difficult to train with several hidden layers)
- Image domain features are often locally aggregated



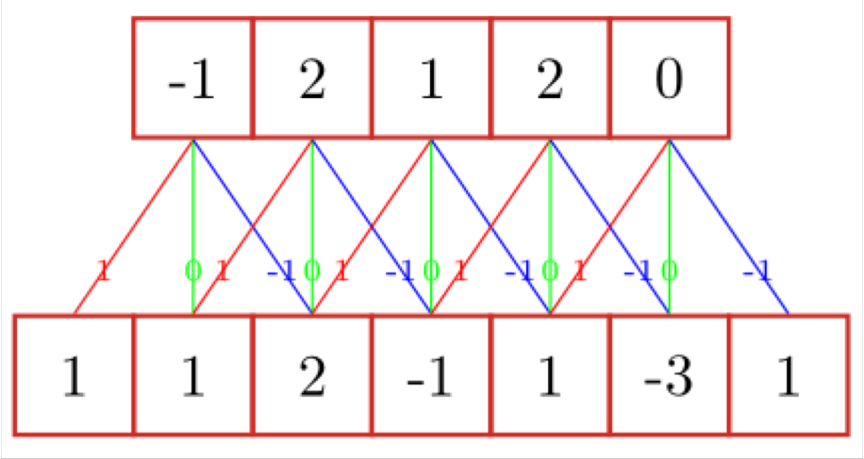
Input image | Convolution kernel | Output feature map

Convolution

- 1D convolution

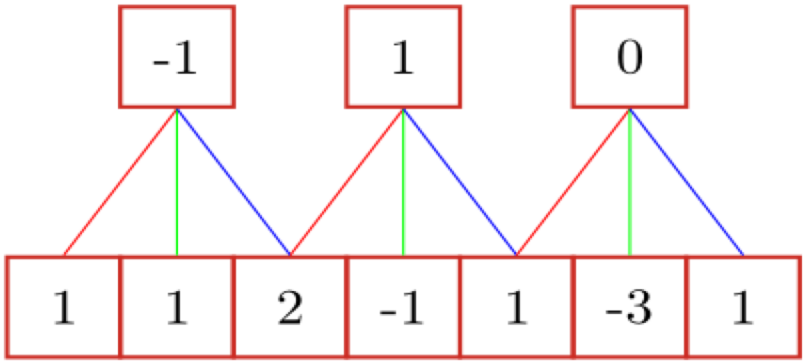
$$g(x) = \sum_{m=-a}^{a} w(m)I(x-m)$$

Filter: [-1,0,1]

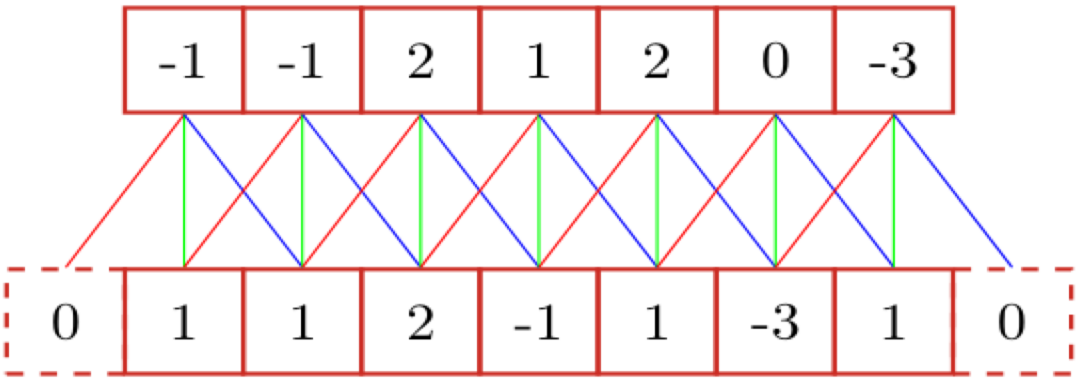


Convolution

- 1D convolution: stride & padding



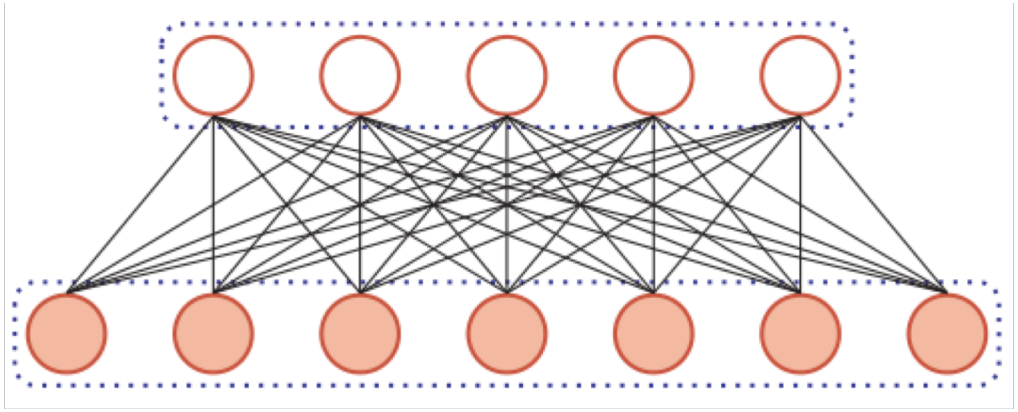
S=2



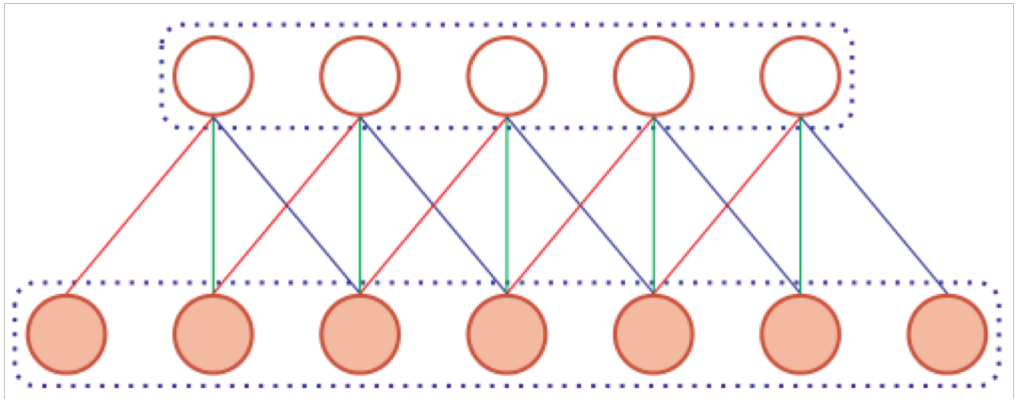
Zero padding

Convolution in NN

- Convolution to replace fully connected



Fully connected

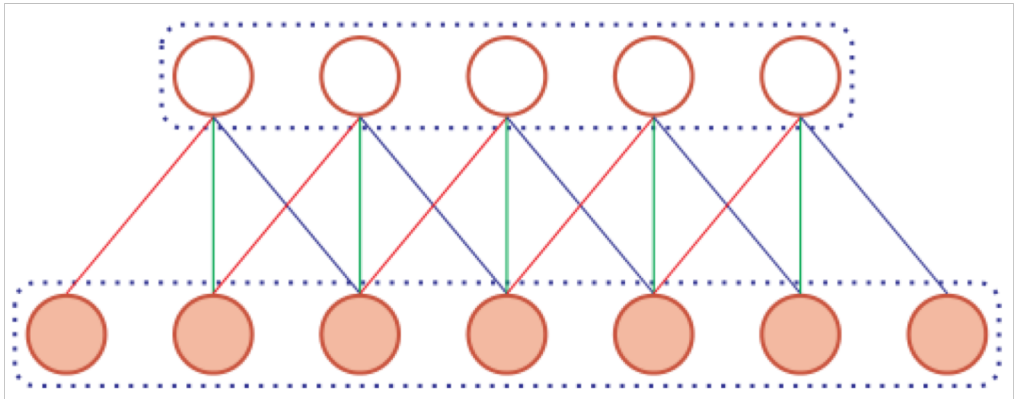


convolution

Convolution in NN

- Convolution to replace fully connected
  - (colour) edges represent the elements in the kernel
  - Values in the kernel = weights on the edges
  - Shared kernel = shared weights
  - Connections are local, instead of global

What have we missed, in order for the convolution result to be useful?

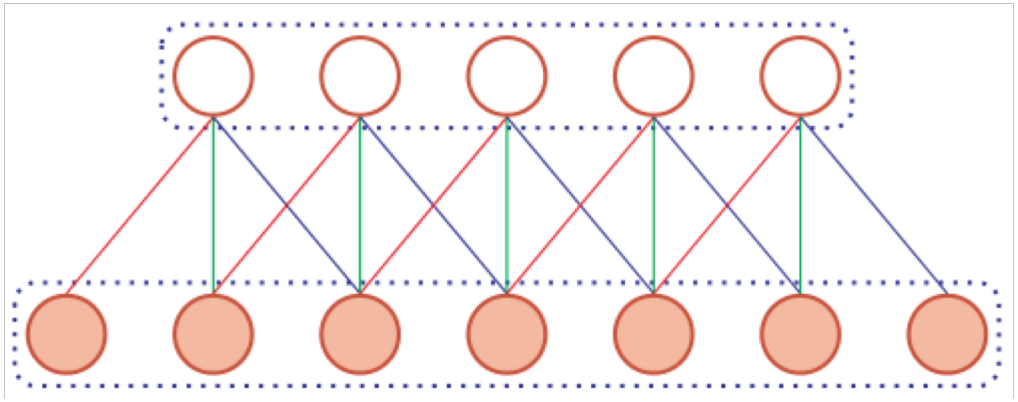


convolution

Convolution in NN

- Nonlinear transformation: activation
  - Convolution is linear calculation
  - Linear transformation is not powerful enough to solve complex problem
  - As in fully connected NN, a nonlinear transformation through activation function, e.g. sigmoid, is necessary (more on this later)
- Bias unit for each filter

- How many parameters here?
- Compare this to FCN.

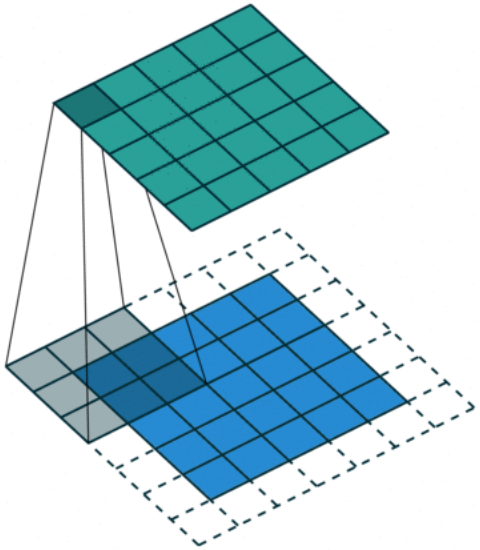


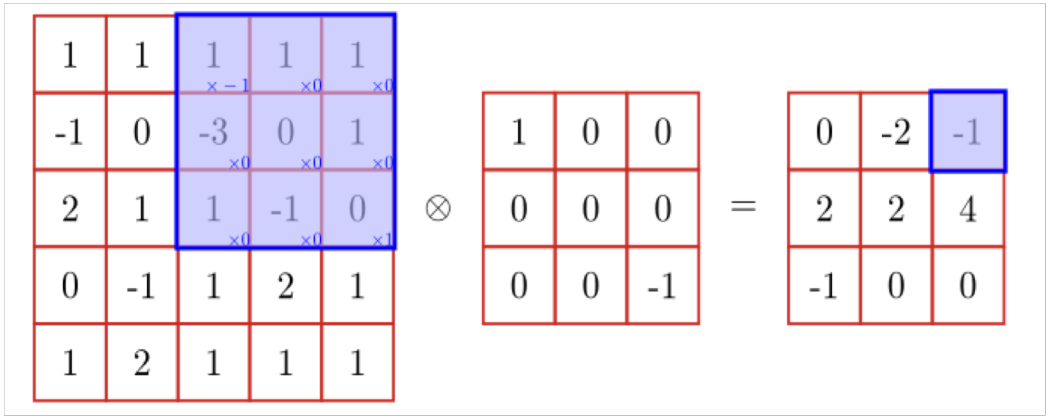
convolution

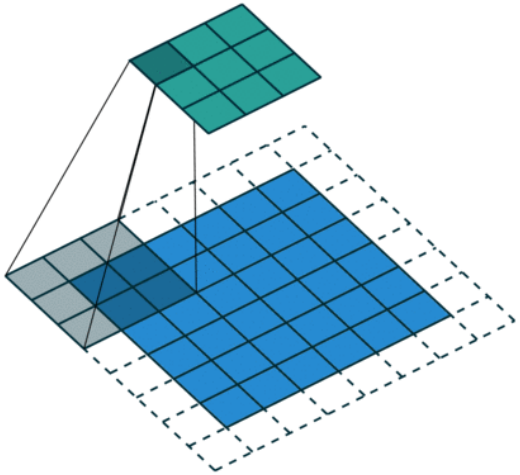
Convolution

- 2D convolution

$$g(x,y) = w * I = \sum_{m=-a}^{a} \sum_{n=-b}^{b} w(m,n) I(x-m, y-n)$$



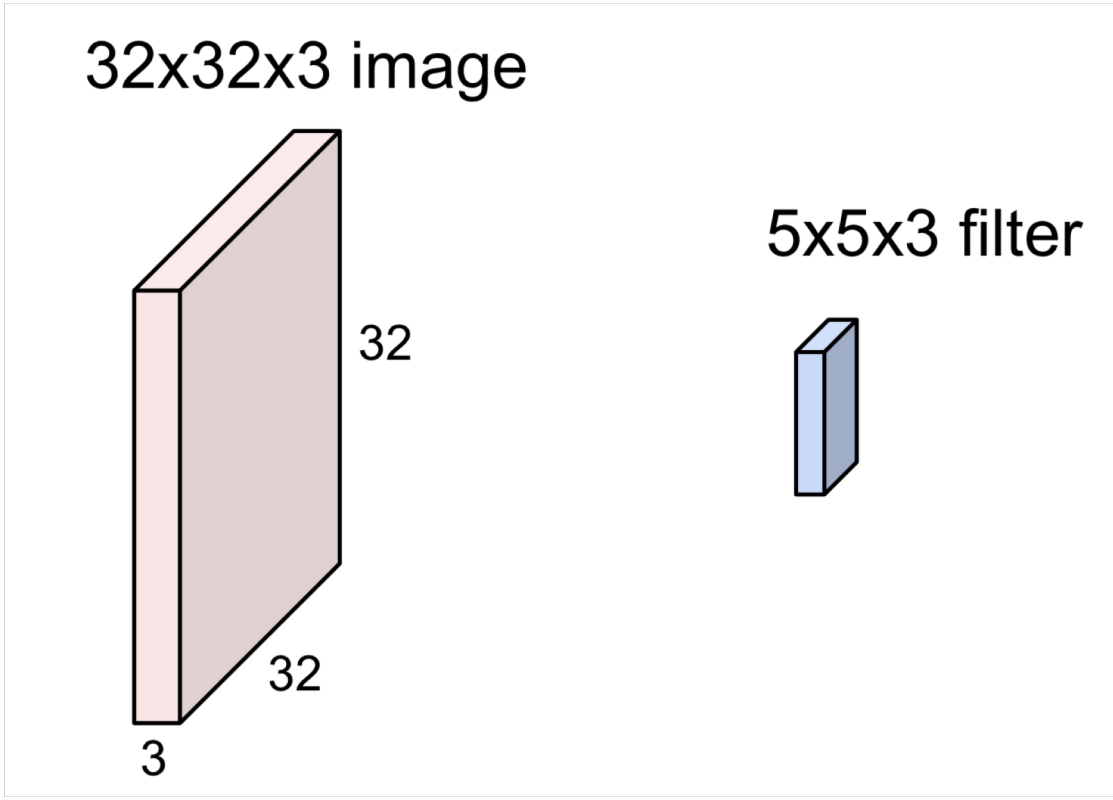




- However, for the purpose of machine learning, there is no difference if do not flip the kernel.
- This means we just need to compute a dot product

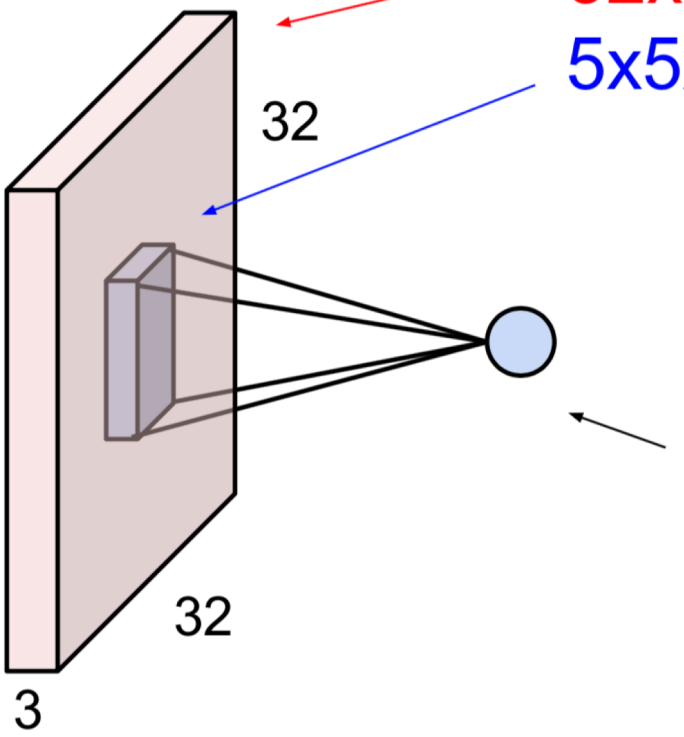
Convolution

- Convolution with multiple channels



Convolution

- Convolution with multiple channels
  - Taking into account the bias term for NN
  - Bias shared by filter

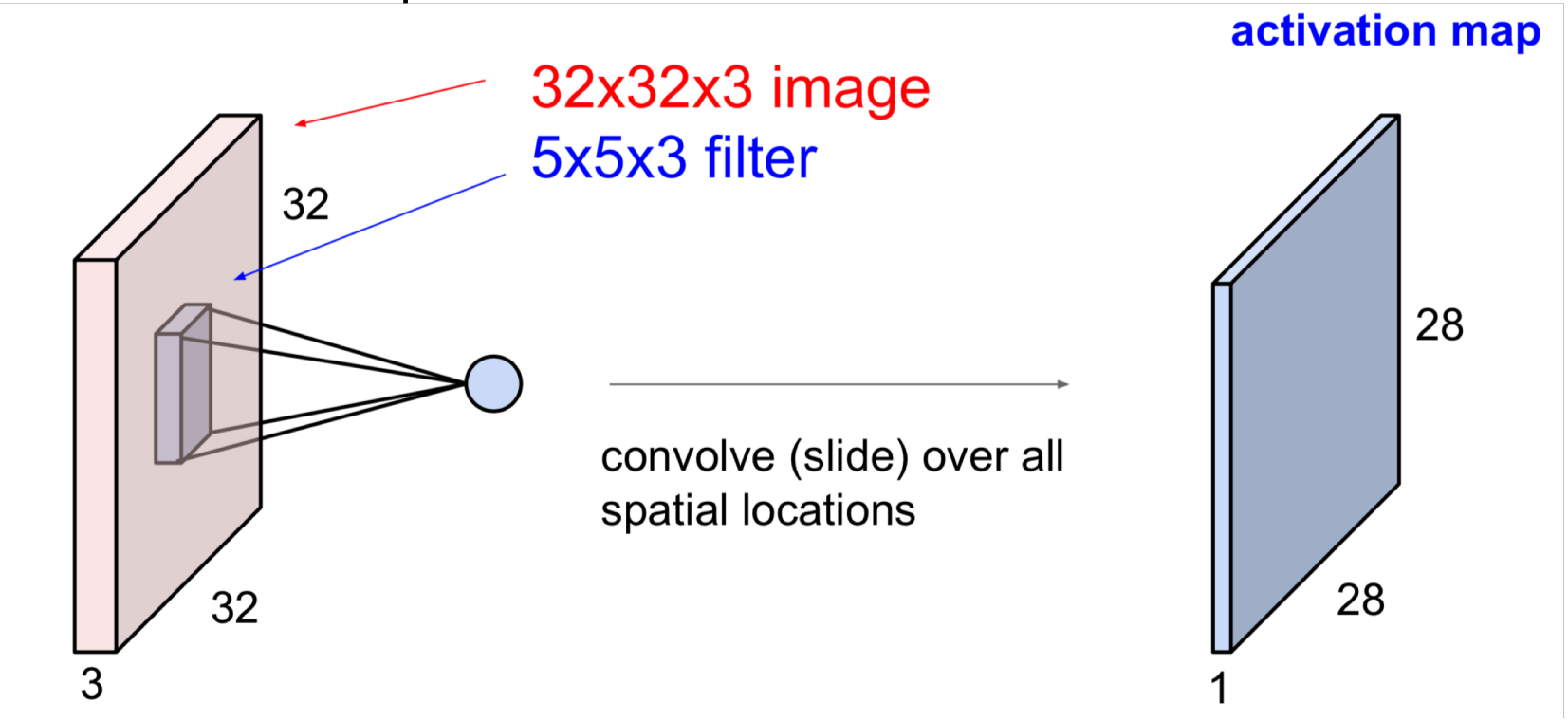


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolution

- Convolution is a linear operation which not useful, directly, for complex problems
- Nonlinear transformation is necessary, as in conventional NN
  - Apply activation function (nonlinear) to produce activation map or feature map

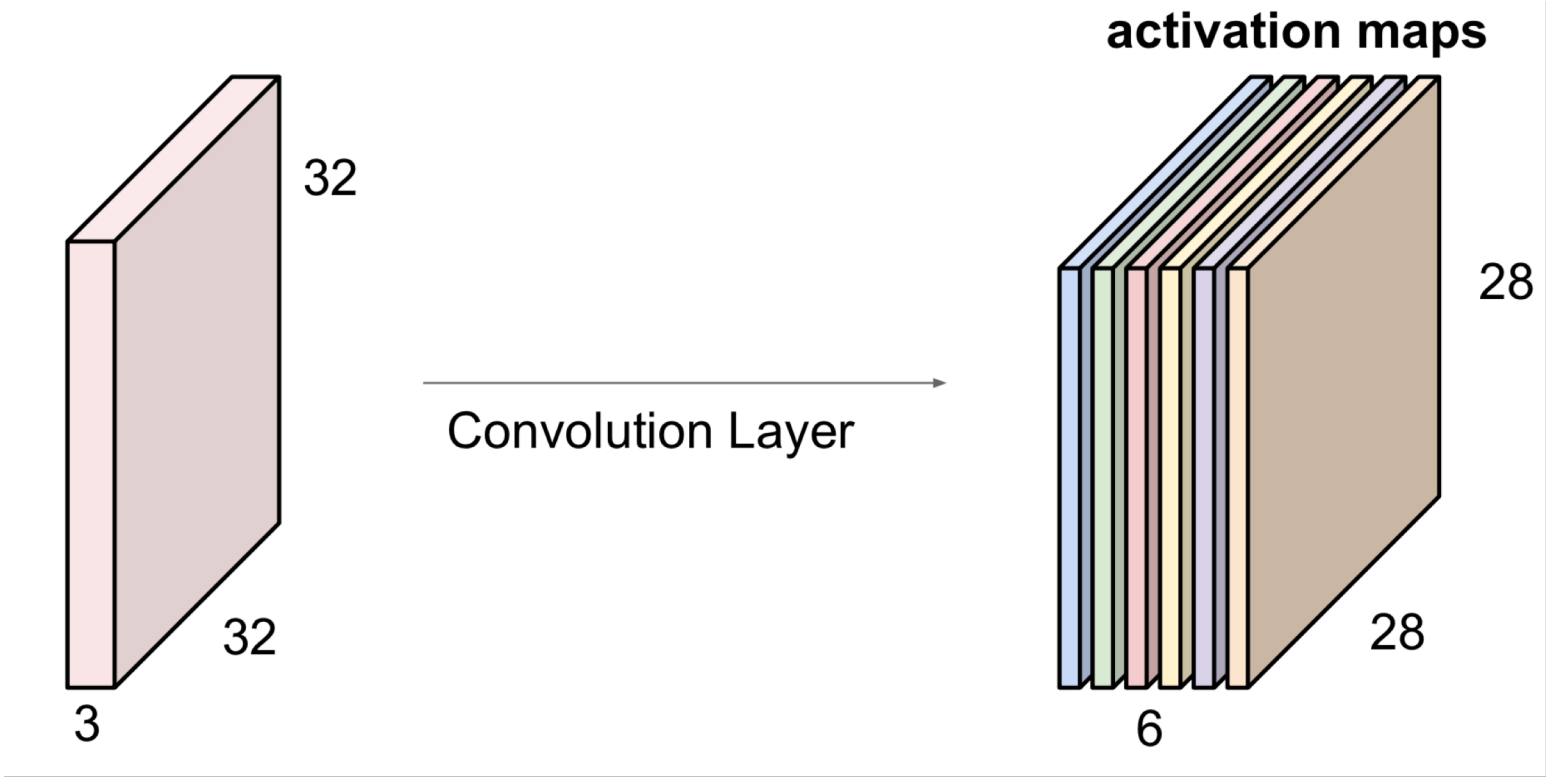


Convolution

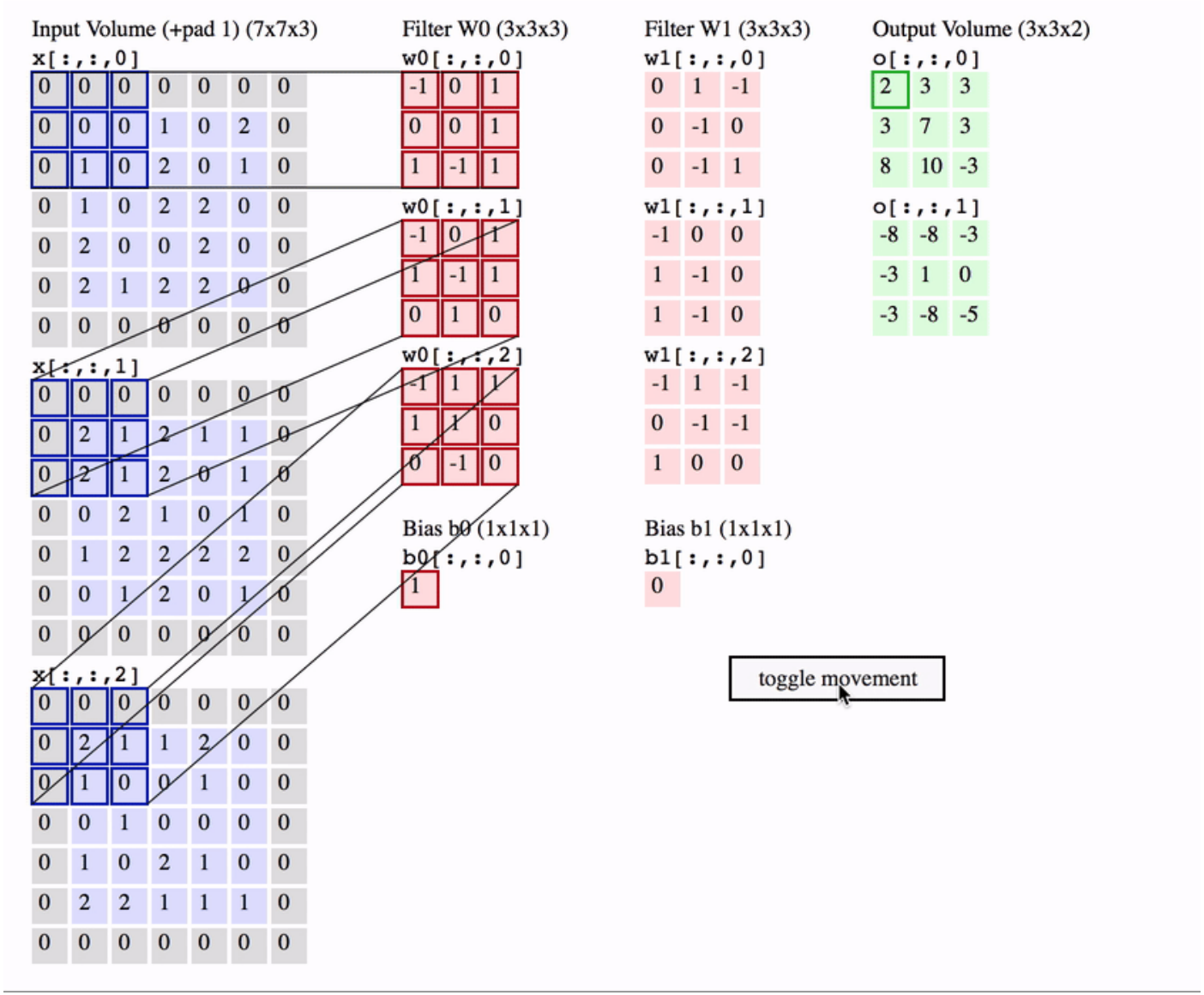
- Consider second filter (green)

Convolution

- Convolution with multiple filters
  - Filters are randomly initialised
  - Together, the filters mine (complementary) features

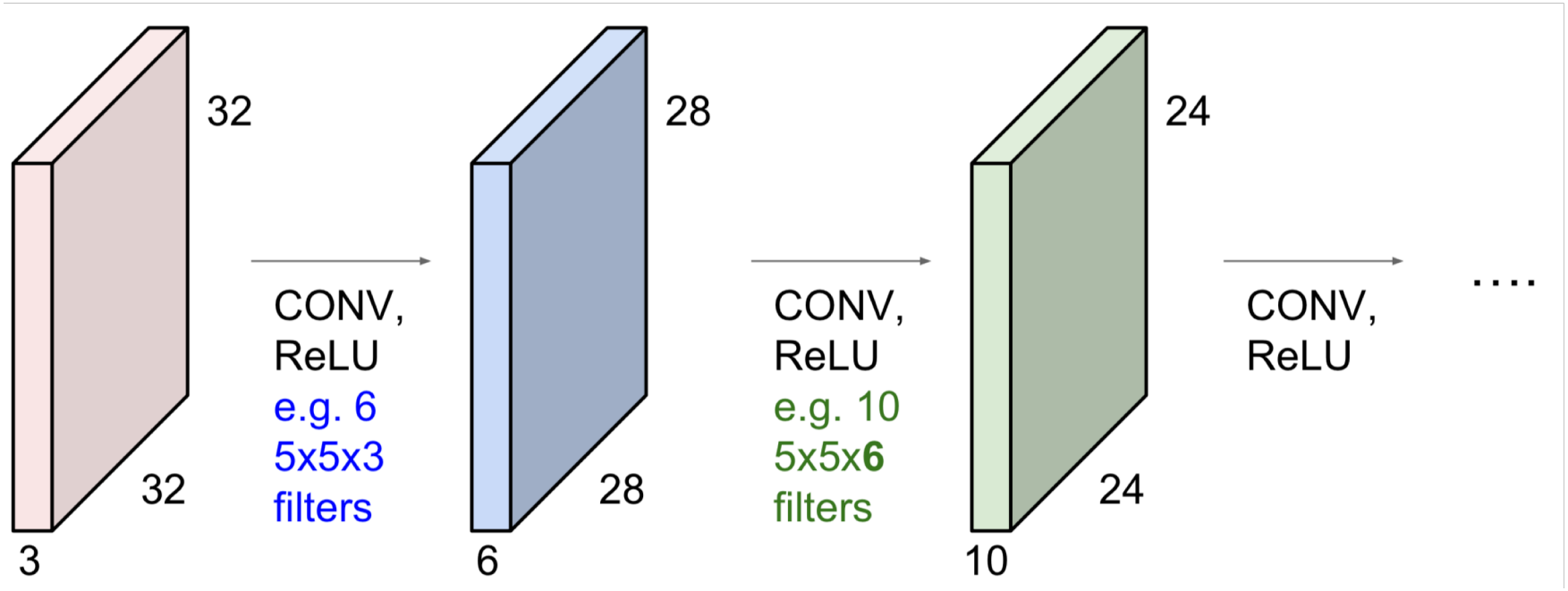


Convolution



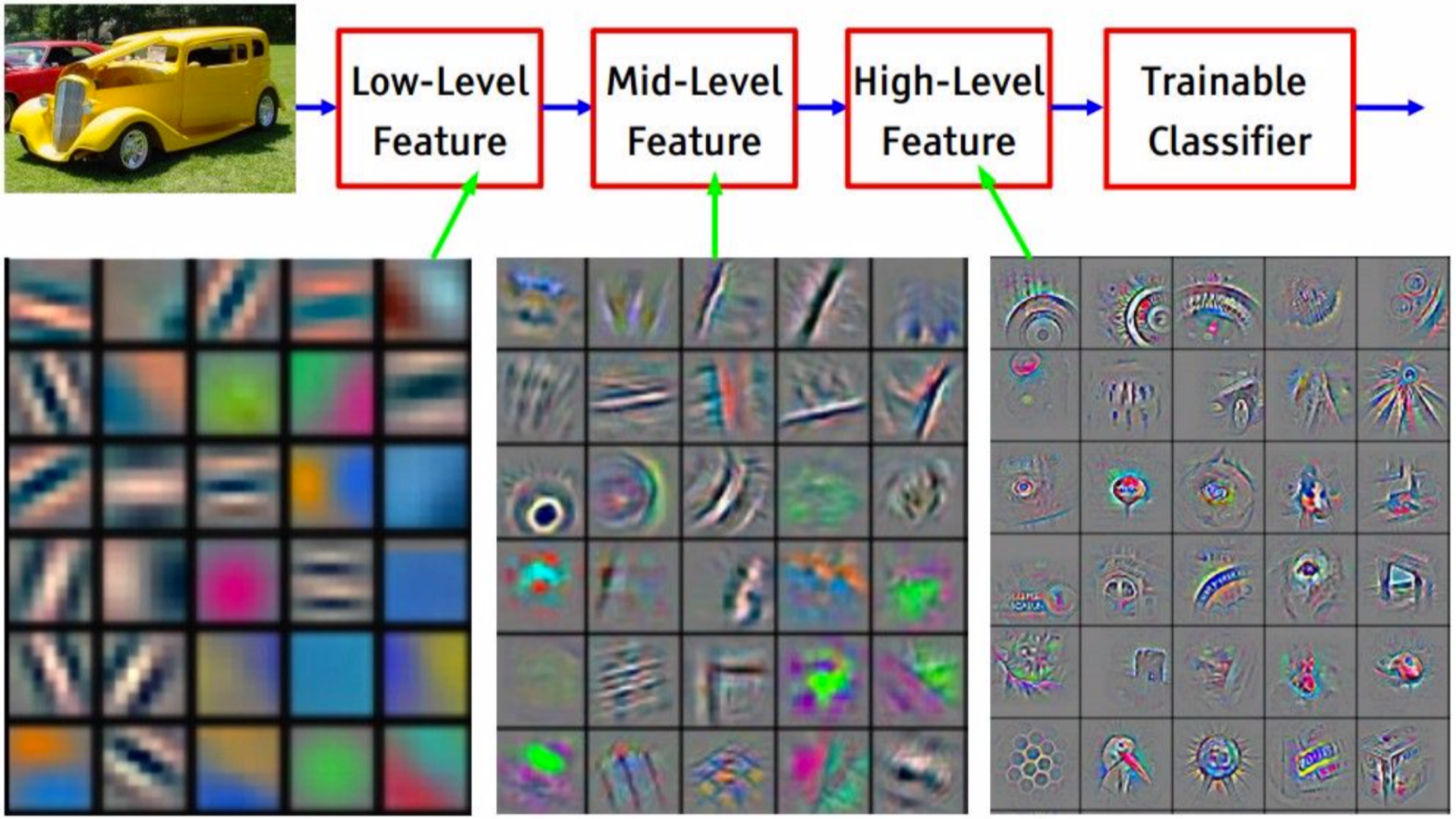
Convolution

- It is possible to perform consecutive convolutions
  - Linear combinations of lower level features to produce higher level features
  - This is similar to fully connected NN, but features are localised in spatial domain: in FCN, features spatially far away from each other can be linearly combined



Convolution

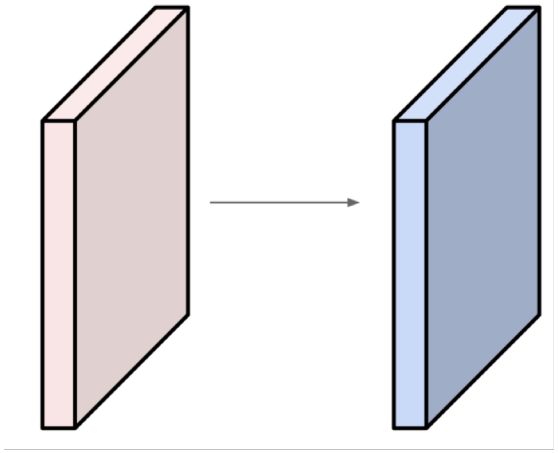
- Visualise convolutional features



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Convolution

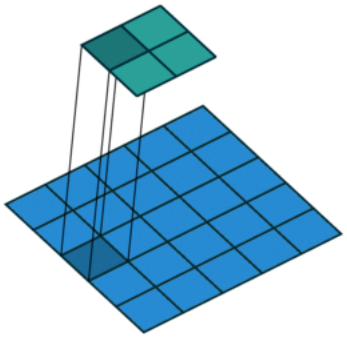
- Zero padding commonly applied to avoid volume shrinking when generating the feature maps/activation maps
- With stride=1, feature map spatial size is the same as input (not the depth)



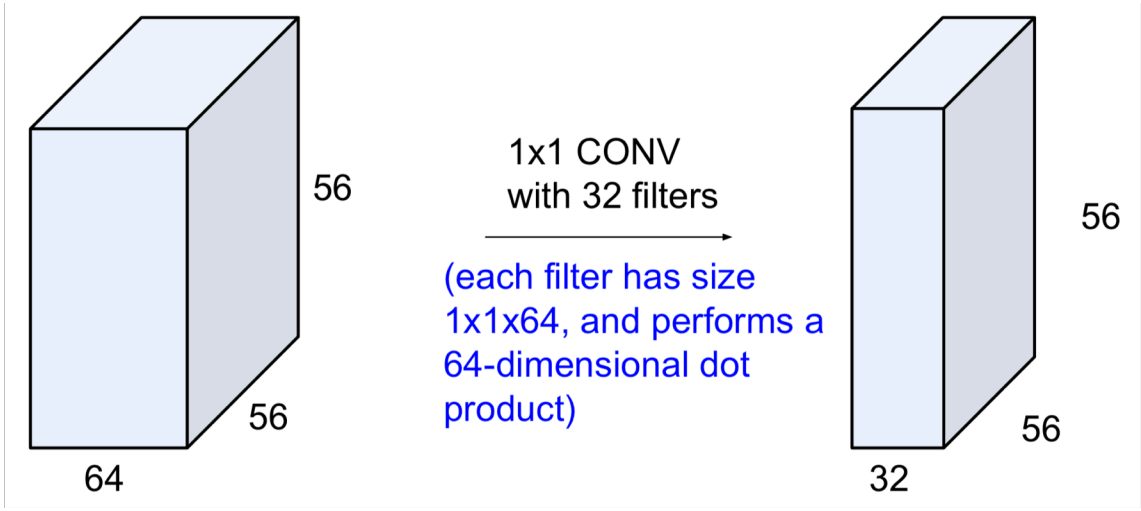
Input volume: 32x32x3, convolution using ten 5x5 filters with stride=1 and zero padding:
- How many parameters?

Convolution

- Convolution with 1x1 filter
  - 1x1 convolution with strides leads to data reduction

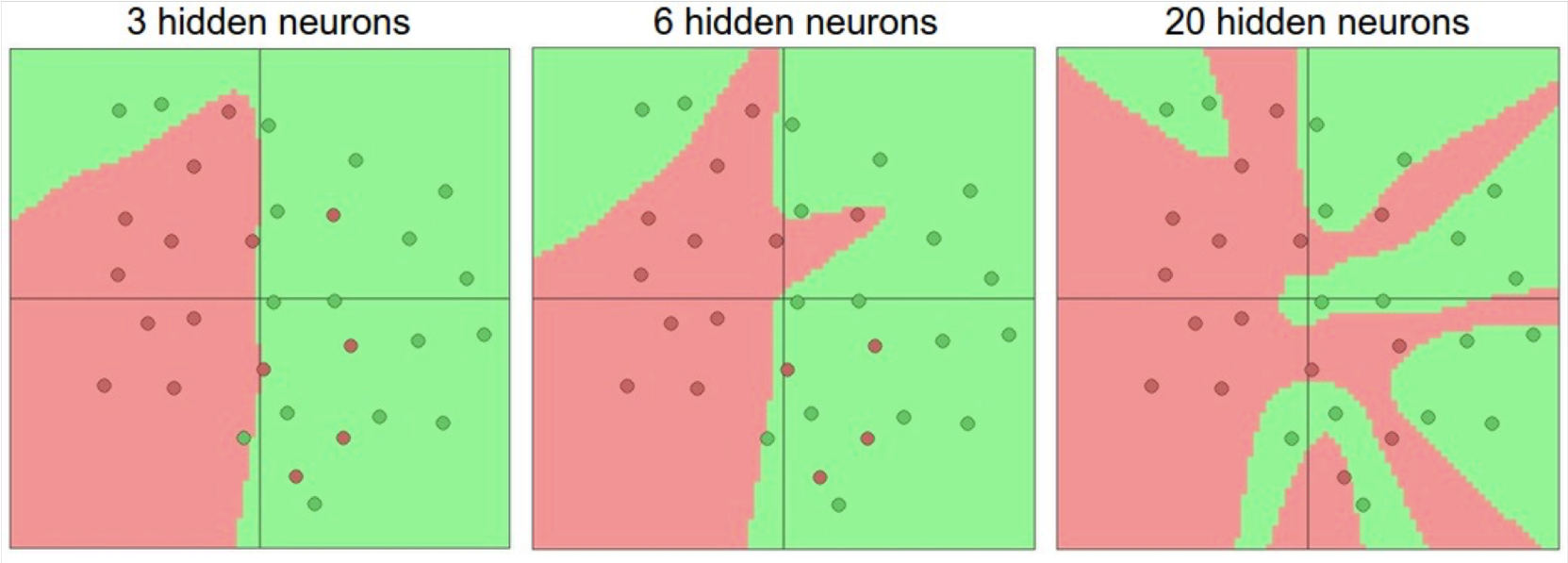


  - Feature dimensionality reduction



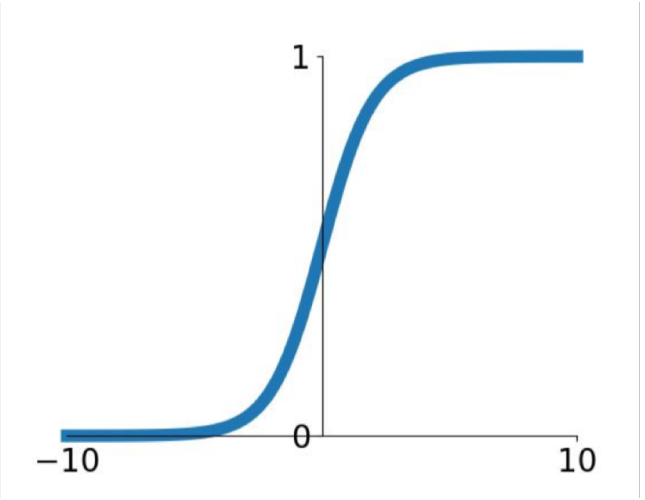
Activation function

- Nonlinearity is necessary to learn complex representation of data
- Activation transforms linear convolution operation to nonlinear



Activation

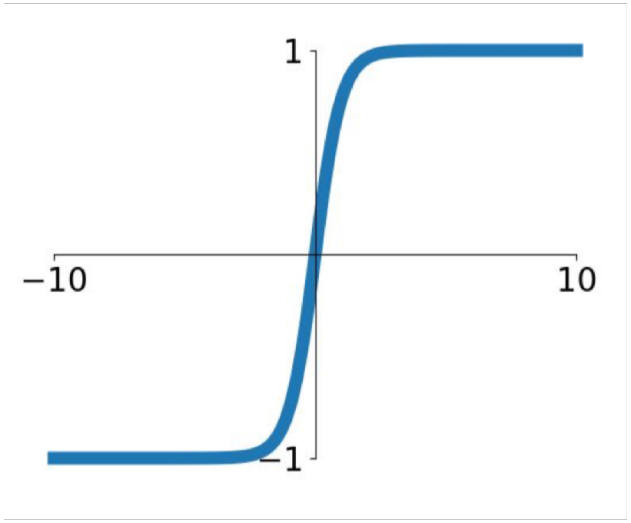
- Sigmoid function
  - Output is within (0,1)
  - Continuous function, differentiable everywhere



  - Exponential function is somewhat expensive to compute
  - Output is not zero centred
  - More importantly for deep learning, gradient vanishes quickly from zero region: stops parameter updating

Activation

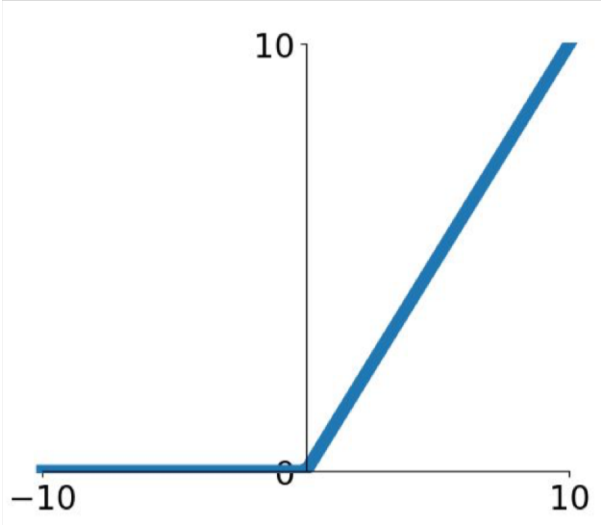
- tanh function
  - Output within (-1, 1)
  - Zero centred



  - Sufferers from the same vanishing gradient problem as sigmoid function

Activation

- Rectified linear unit (ReLU)
  - $f(x) = max(0,x)$

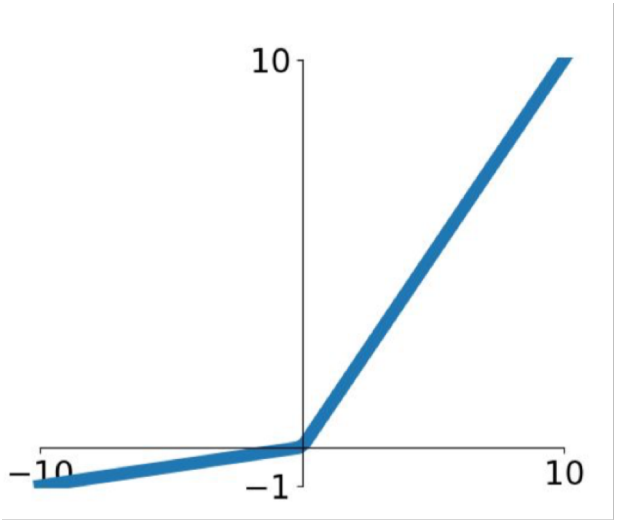


  - Does not saturate in the positive region
  - Very efficient
  - Converges faster than sigmoid

  - Not zero centred
  - Negative region: no gradient to update which leads to dead activation units (initialise with small positive bias)

Activation

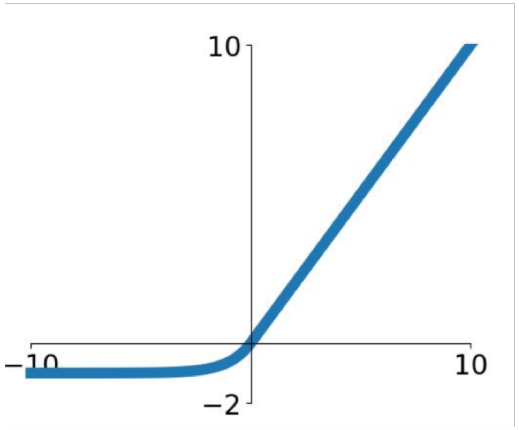
- Leaky ReLU
  - $f(x) = max(0.01x, x)$
  - Does not saturate
  - Equally efficient
  - Fast convergence
  - No dead activation units
  - Not zero centred



Activation

- Exponential linear unit (ELU)

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

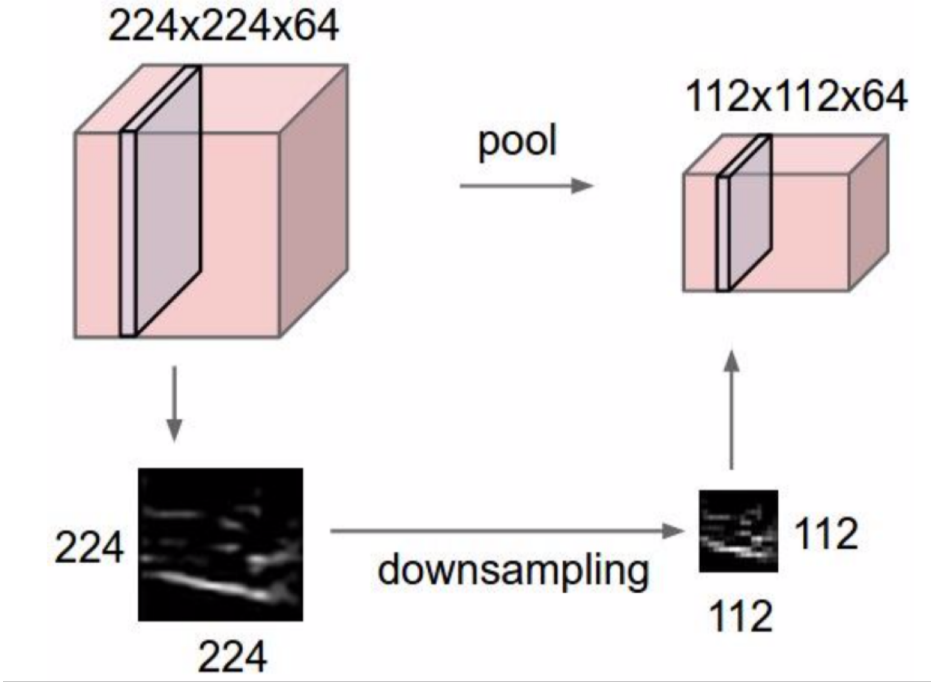


  - Fast convergence
  - It can take on negative values, so the average output of the neurons in any given layer is typically closer to 0 than when using the ReLU activation function never outputs negative values). This helps alleviate the vanishing gradients problem.
  - It always has a nonzero derivative, which avoids the dying unit
  - It is smooth everywhere; abrupt change can slow down Gradient Descent because it will bounce around f = 0.

  - Computationally requires exponential function

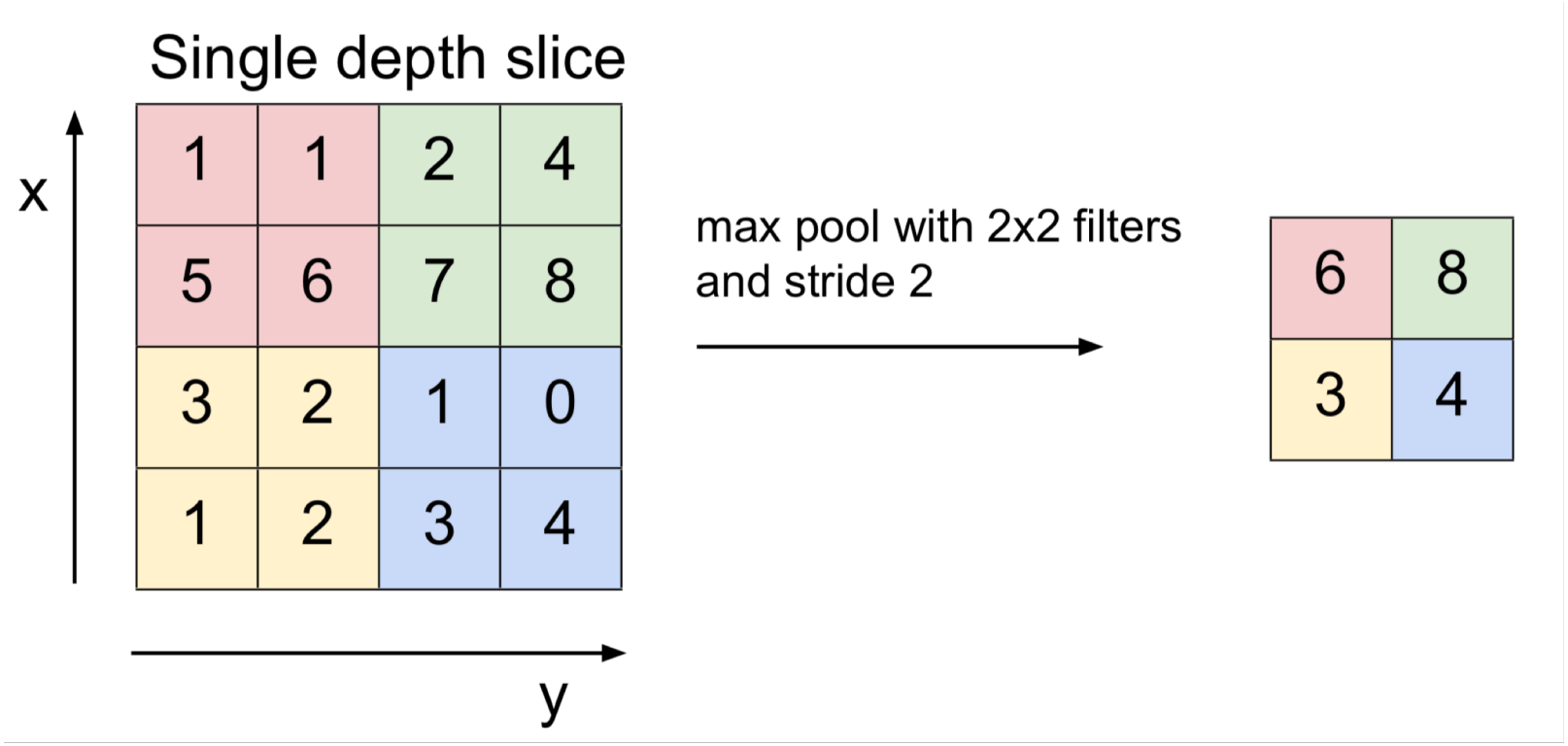
Pooling

- Pooling makes the representation more compact
- Computationally more manageable as the network goes deeper
- Allows multi-scale analysis
- Operates on each activation map independently



Pooling

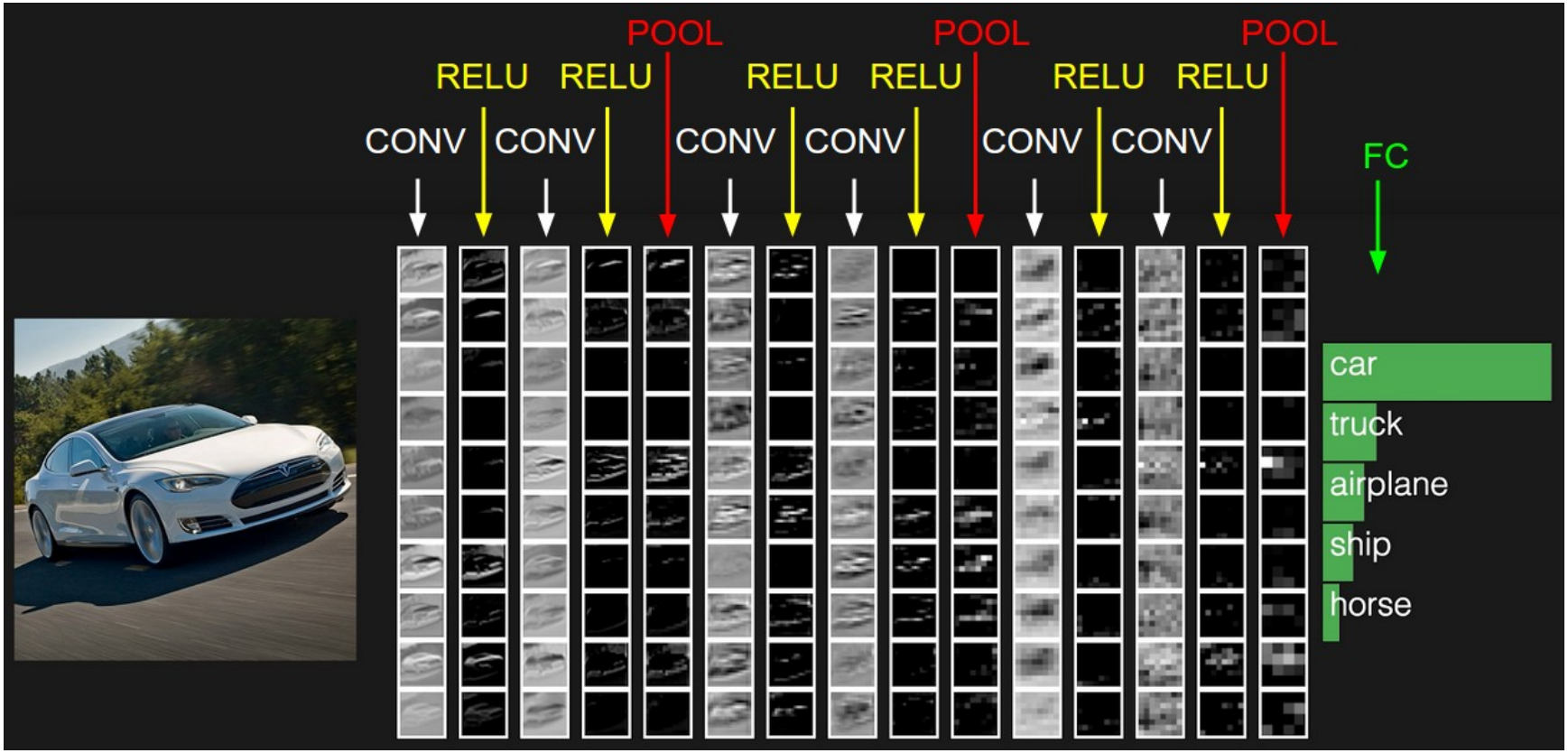
- Max pooling



- Min-pooling
- Mean-pooling

CNN for classification

- Adding fully connected (FC) layers



Softmax

- The output from, e.g., ReLU can no longer be interpreted as probability
- To interpret raw classifier scores as probabilities, we use softmax classifier

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

cat **3.2**

car 5.1

frog -1.7

Softmax

- The output from, e.g., ReLU can no longer be interpreted as probability
- To interpret raw classifier scores as probabilities, we use softmax

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

		exp		normalize	
			Probabilities must be >= 0		Probabilities must sum to 1
cat	3.2		24.5		0.13
car	5.1	→	164.0	→	0.87
frog	-1.7		0.18		0.00
			unnormalized probabilities		probabilities

Softmax

- Can be used as a classifer
- Training can be treated similar to logistic regression

$$s = f(x_i; W)$$

$$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax Function

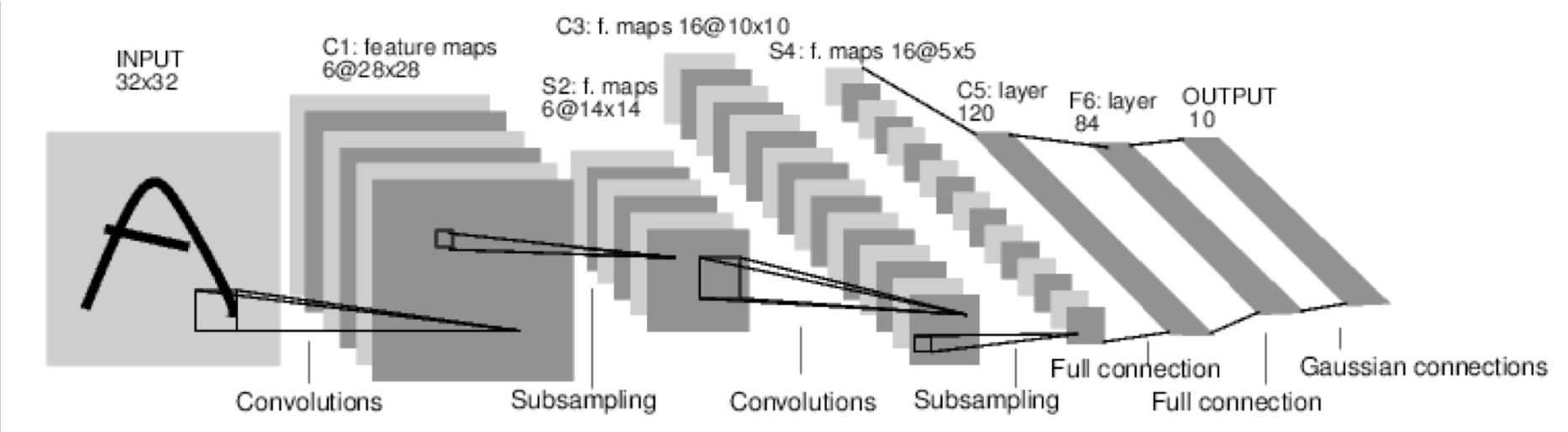
cat	**3.2**	exp →	**24.5**	normalize →	**0.13**	compare ← **1.00**
car	5.1		164.0		0.87	0.00
frog	-1.7		0.18		0.00	0.00
			unnormalized probabilities		probabilities	Correct probs

Probabilities must be >= 0

Probabilities must sum to 1

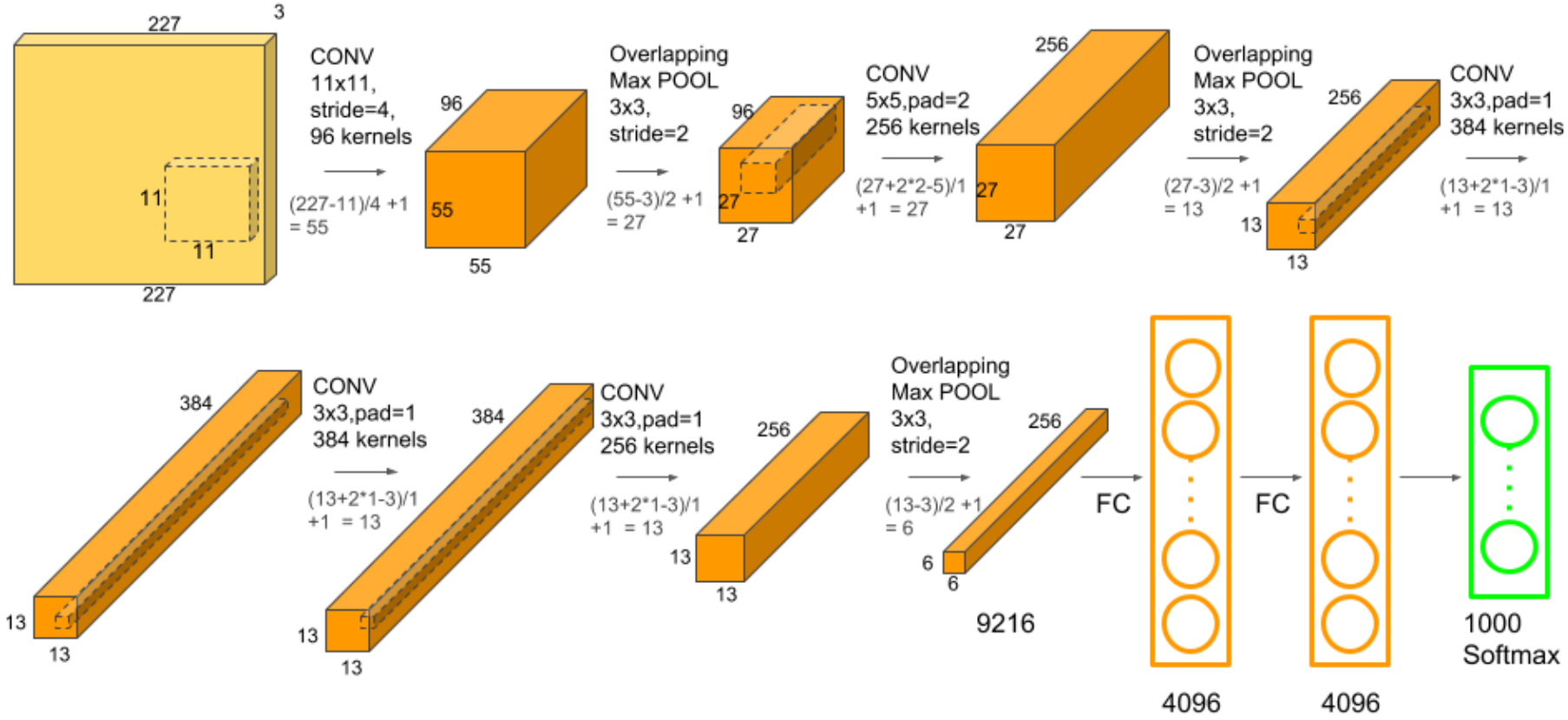
LeNet5 CNN Architecture:

- Input: 32*32*1 pixel intensities
- 1C - **in**: 32*32, **filter**: 5*5*1*6, **stride**: 1, **pad**: 1, **out**: 6@28*28
- 2P - **in**: 28*28**, filter**: 2*2, **stride**: 5, **pad**: 5, **out**: 6@14*14
- 3C - **in**: 14*14, **filter**: 5*5*1*16, **stride**: 1, **pad**: 1, **out**: 16@10*10
- 4P - **in**: 10*10, **filter**: 2*2, **stride**: 5, **pad**: 5, **out**: 16@5*5
- 5C - **in**: 5*5, **filter**: 5*5*120, **stride**: 1, **pad**: 1, **out**: 120@1*1
- 6F - **in**: 1*1, **fully connected NN**, **out**: 10@1*1



AlexNet

- Initially developed to tackle image classification (imageNet)
  - 1000 classes
  - Input 227x227 colour images



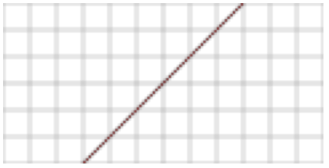
Tips on Architectural Design

- Typical architectures look like

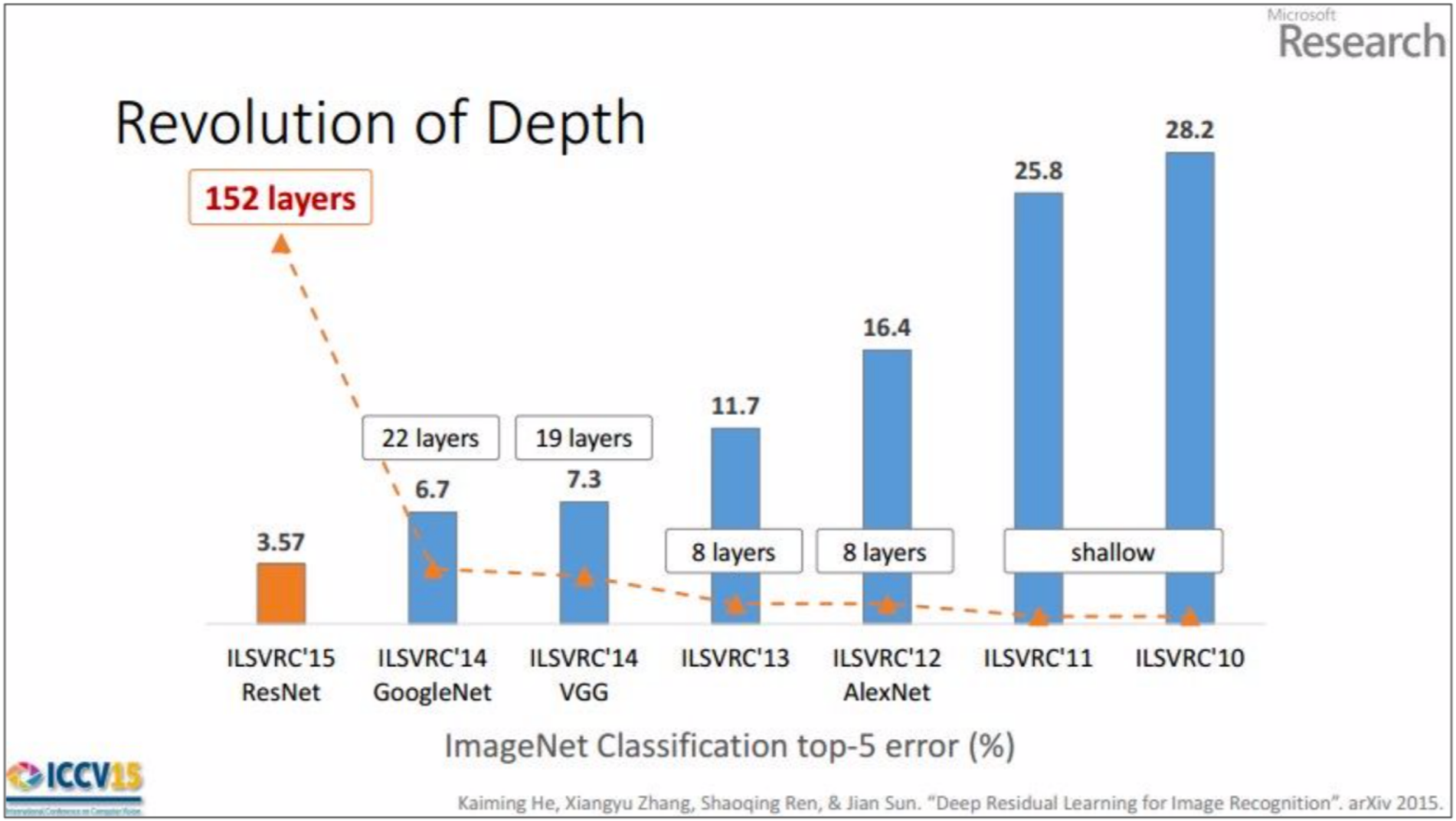
[(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX

- where N is usually up to ~5, M is large, 0 <= K <= 2.

Cost function

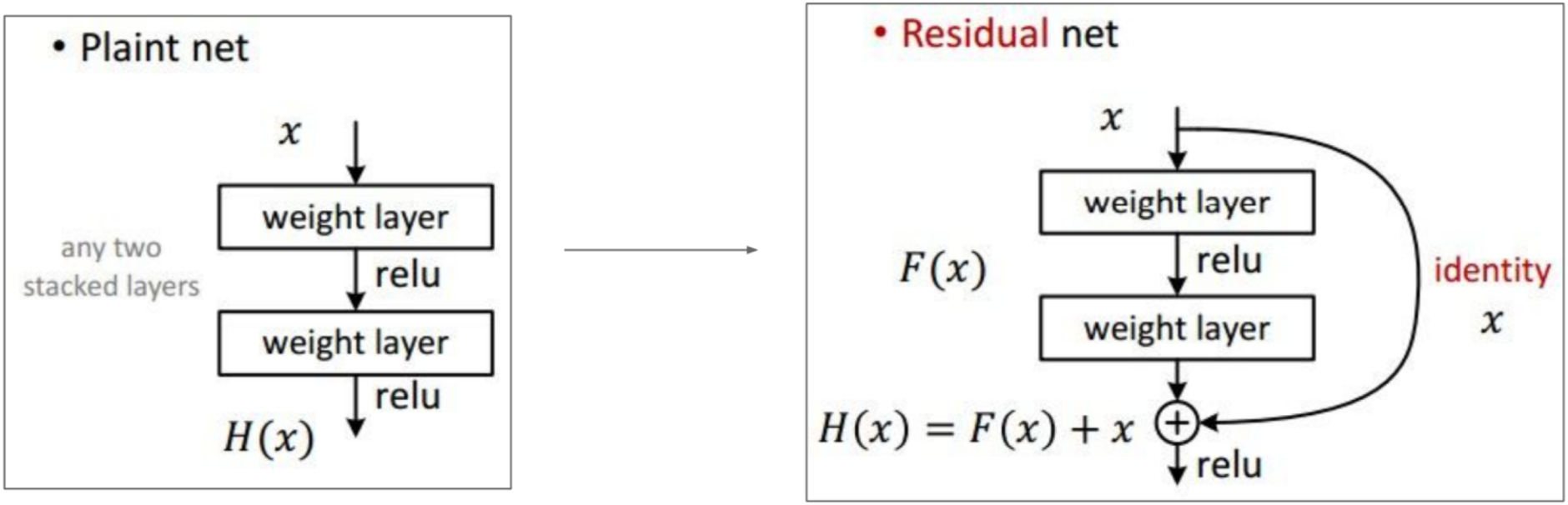
	Classification	Regression
Output Layer	Soft-max [map to a probability distribution] $P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\top \mathbf{w}_k}}$	Linear (Identity) or Sigmoid 
Cost (loss) function	Cross-entropy $J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\left[y_k^{(i)} \log \hat{y}_k^{(i)} + \left(1 - y_k^{(i)}\right) \log\left(1 - \hat{y}_k^{(i)}\right)\right]$	Mean Squared Error $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$ Mean Absolute Error $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left\lvert y^{(i)} - \hat{y}^{(i)}\right\rvert$

Deep structures



Deep structures

- Deep models have the potential to provides features with semantic meanings
- Simply stacking layers causes vanishing gradient in backpropagation
- ResNet introduces shortcut connections for training



Overfitting

- Overfitting problem
  - Huge amount of parameters, e.g. billions
  - Various techniques to minimise overfitting
    - More data if possible
    - Data augmentation, e.g. rotate, scale, perturb images
    - Change learning rate as validation error plateaus
    - Reduce network complexity
    - Applying dropout: random delete nodes in (only) training