Imperial College London

Lecture 9: Convolutional neural networks

Introduction to machine learning

Kevin Webster

Department of Mathematics
Imperial College London

Outline

Outline

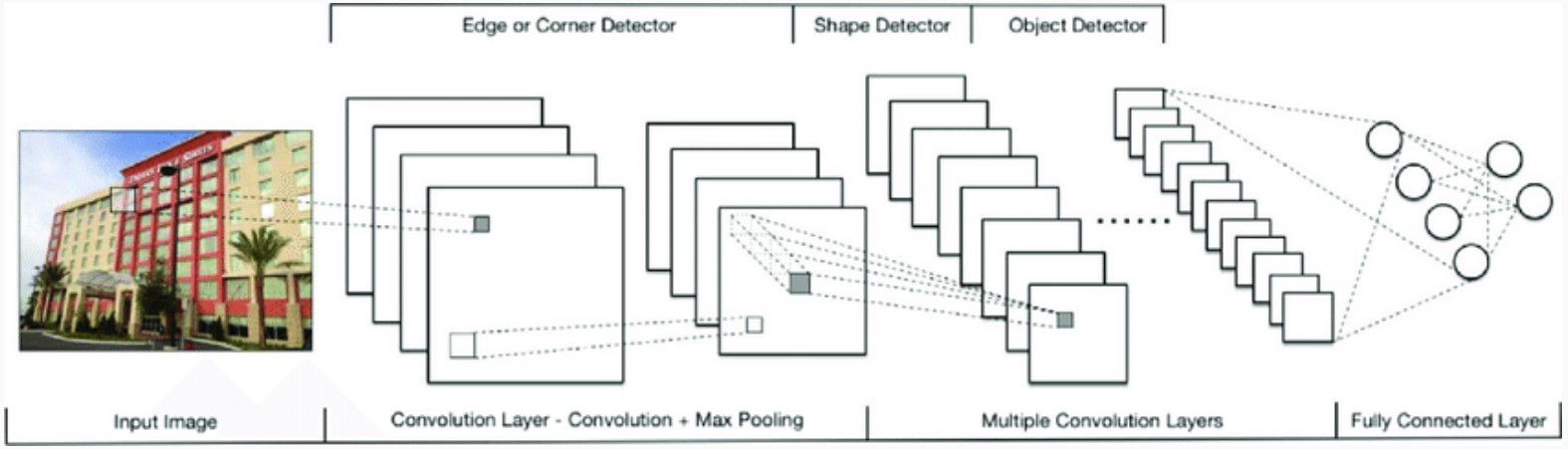
CNN fundamentals

- Convolution, padding, strides and pooling

CNN architectures

- AlexNet
- VGG
- GoogLeNet/Inception
- ResNet

Convolutional neural networks



- **Convolutional neural network**, **CNN** or **ConvNet** is a class of neural network with a special structure
- Breakthrough improvements in image processing
- Also used in NLP, audio waveform analysis and generation, and RL models for games

Convolution operation

The convolution operator

$$(f * w)(t) := \int_{-\infty}^{\infty} f(\tau) w(t - \tau) d\tau$$

can be described as a weighted average of the **input** f according to the weighting (or **kernel**) w at each point in time t.

The discrete convolution is given by

$$(f * w)(t) := \sum_{\tau=-\infty}^{\infty} f(\tau) w(t - \tau).$$

Note that when w has a finite support, a finite summation may be used.

Convolution operation

In convolutional neural networks with image inputs, a two dimensional convolution is used:

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

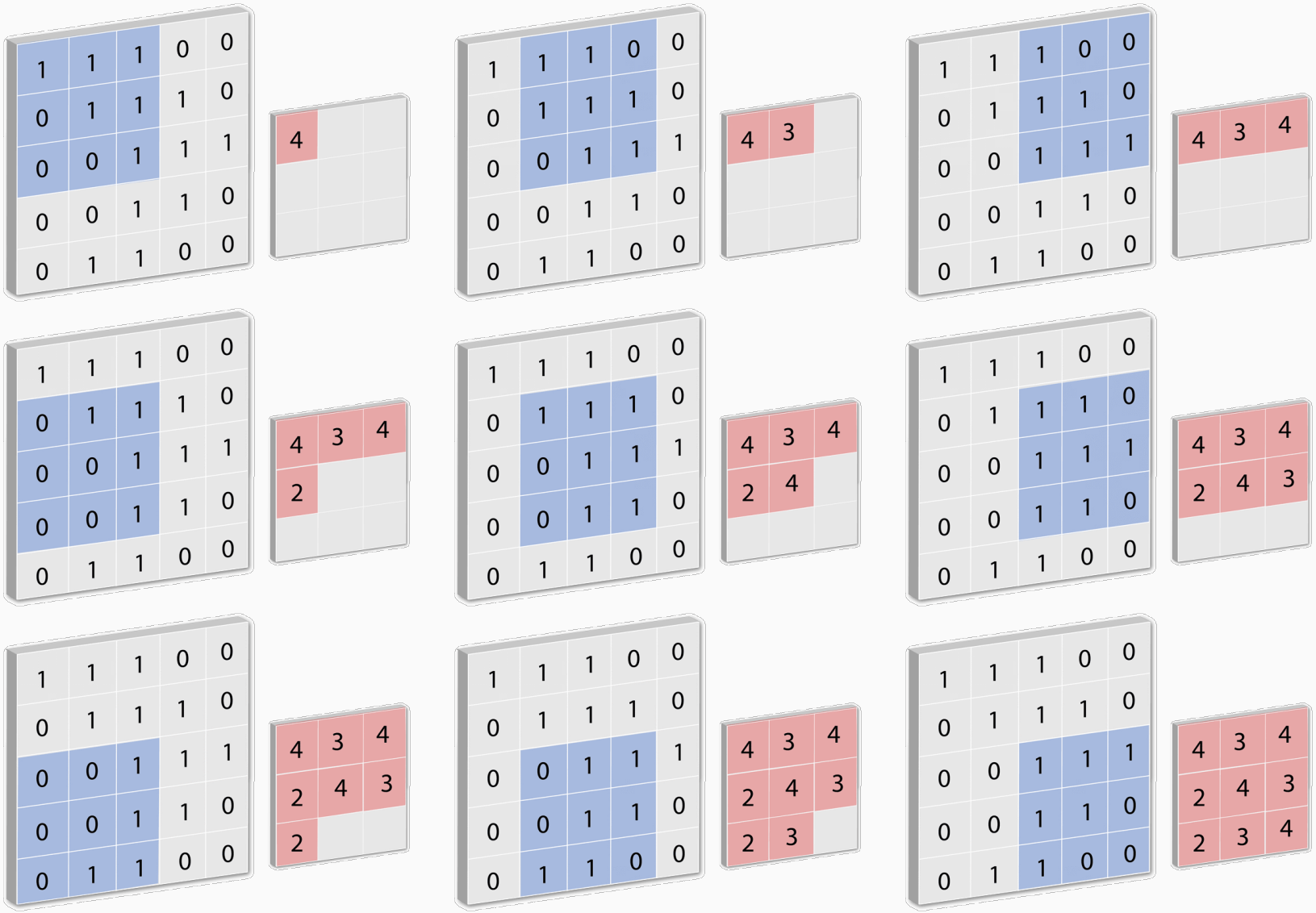
The kernel K will have a fixed size, outside which is can be assumed to be zero.

In practice, many libraries implement the cross-correlation operation, which is the same as above but changing the orientation of the arguments:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

We will consider the above operation in CNNs and refer to it as a **convolution**.

Convolution operation



Convolution operation

- The **kernel** is also referred to as a **filter**
- The support of the kernel defines its **receptive field**
- Output of a convolution is often referred to as a **feature map**
- There are usually many feature maps defined for each layer
- Within each layer, there are several **channels** (e.g. there are 3 channels in the input layer for an RGB image)
- In practice, the kernels for each feature map are usually three dimensional tensors (esp. for images) and the convolution operation is performed across all channels in the input layer:

$$S^l(i,j) = \sum_m \sum_n \sum_c I(i+m, j+n, c) K^l(m,n,c)$$

- The convolution is usually combined with a shared bias (one per channel) and passed through an activation function

Convolution properties

- Design inspired by visual processing systems
- **Sparse interactions**: a feedforward network connects every input to every output. The sparse connectivity of CNNs reduces number of parameters significantly and improves efficiency
- **Parameter sharing**, or tied weights, means that the CNN does not need to learn a separate set of parameters for each location, but reuses one set in every location
- **Equivariant features**: due to the parameter sharing used in CNNs, the feature maps are equivariant with respect to translations. (However note that convolutions are not equivariant with respect to other transformations such as rotation.)
- Also enables some flexibility in input size

Padding

In general, we can also define the convolution operation for N-dimensional inputs. The kernel/filter is then a tensor of shape $(n, m, k_1, \ldots, k_N)$, where

$$\begin{aligned} n &= \text{number of output maps} \\ m &= \text{number of input maps} \\ k_j &= \text{kernel size in dimension } j \end{aligned}$$

The output size o of a convolution along a dimension with input size i and kernel size k is given by

$$o = (i - k) + 1.$$

Padding

The output size is frequently controlled by padding the input with zeros, effectively increasing the input size.

- **'SAME'** (or half) padding adds $p = k - 1$ zeros to the input to constrain the output size to be the same for unit strides, $o = i$. For odd-sized kernels, add $p = \lfloor k/2 \rfloor$ zeros both sides of the input
- **'VALID'** padding is the standard terminology for when no padding is used
- **'FULL'** padding adds $p = 2(k - 1)$ zeros to the input ($k - 1$ zeros on both sides), resulting in an output size of $i + (k - 1)$

Strides

Convolutions may also use a **stride** s, which is the distance between consecutive positions of the kernel (the convolutions considered so far use $s = 1$).

In this case, the output size is given by

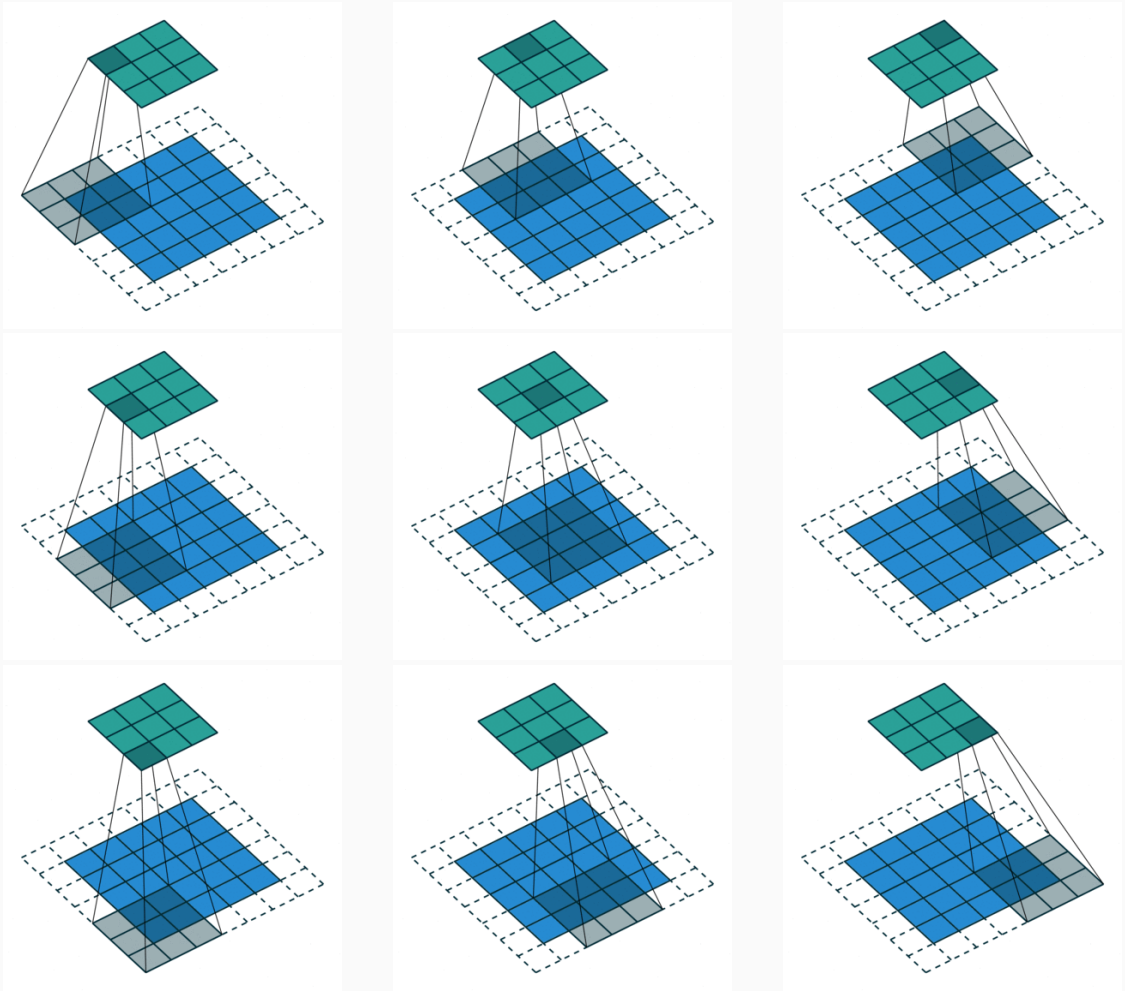
$$o = \left\lfloor \frac{i + p - k}{s} \right\rfloor + 1$$

Thus, strides are one way of *downsampling* within a convolutional neural network.

Note that when using strides $s > 1$, different input sizes can still lead to the same output size.

Strides

Example: 3×3 convolution with half ('SAME') padding $p = 2$, stride $s = 2$, $i = 5 \quad \Rightarrow \quad$ output size $o = 3$.



Pooling

In typical CNNs, convolutional layers are interleaved with **pooling** layers. These layers compute a summary statistic over regions of the input space identified by a sliding window.

It can be thought of as being similar to the convolution operation, where the linear transformation is replaced with a pooling operation.

- **Max pooling** computes the maximum activation per channel in the window
- **Average pooling** computes the average activation per channel in the window
- **L^2 norm** computes the 2-norm of activations per channel in the window

Pooling

- The pooling operation introduces a degree of *translation invariance* to the input
- This is appropriate when we do not need to know the exact location of a feature, just that it is there
- It can be thought of as a prior assumption that the learned function must have translation invariance
- Pooling is often used with strided windows, leading to downsampling of the input
- We can also pool over the outputs of separate feature maps, allowing the network to learn which transformations to become invariant to
- Pooling output can also be computed as $o = \left\lfloor \frac{i-k}{s} \right\rfloor + 1$
- Pooling is useful for handling inputs of varying size

Outline

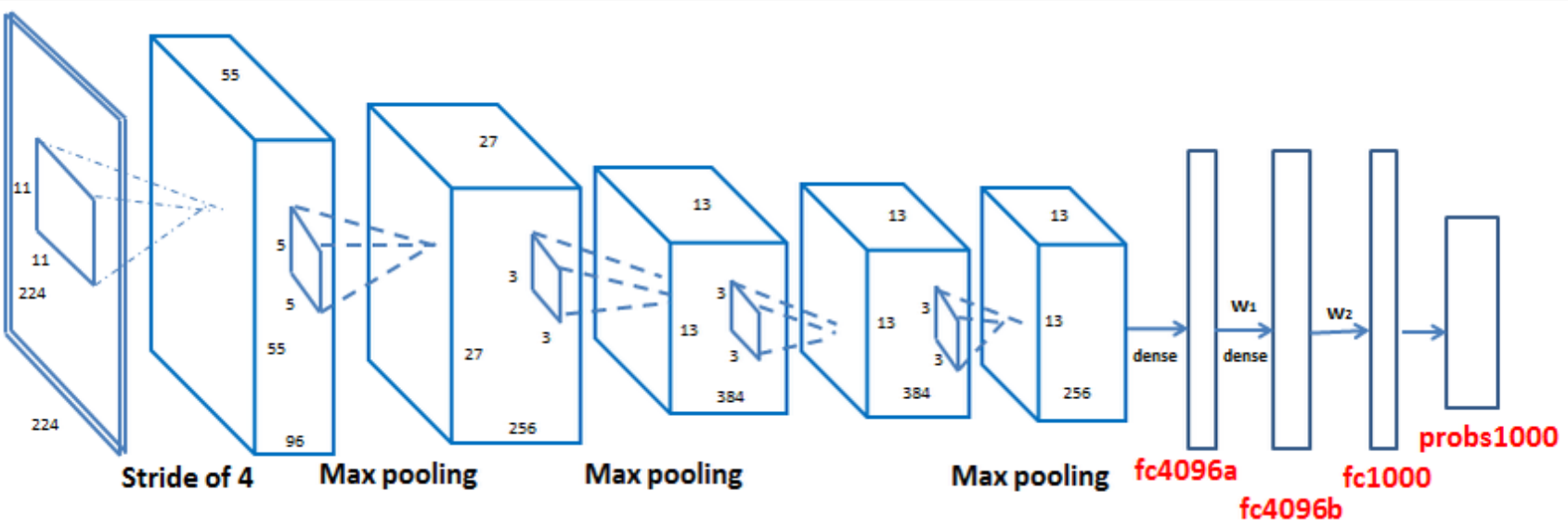
CNN architectures

- Certain CNN architectures have contributed to significant progress in image recognition
- ImageNet is a standard dataset to compare networks against each other
- ImageNet has been running an annual competition since 2010: the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)**
- 1.2 million images with 1000 classes for recognition

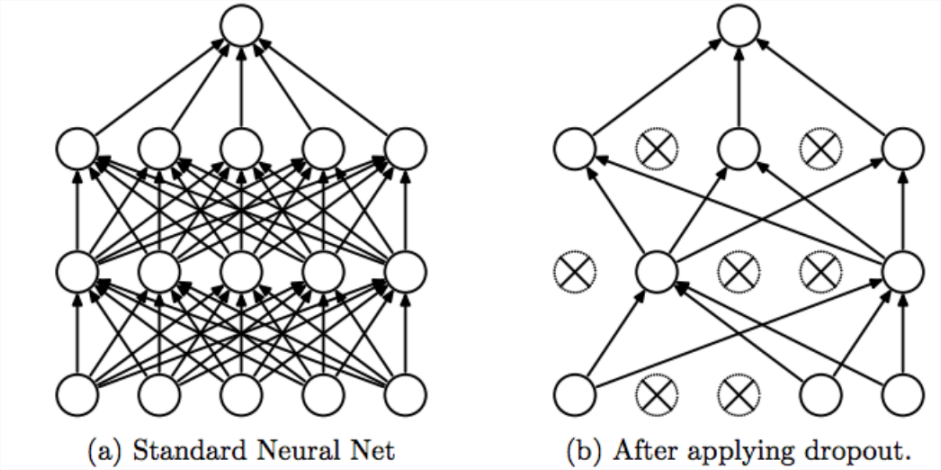
AlexNet

- AlexNet was published in 2012, and competed in the ImageNet competition in the same year
- Achieved top-5 error rate of 15.3%, more than 10.8% lower than that of the runner up
- One of the first deep networks to significantly outperform traditional methodologies in image classification
- Composed of 5 convolutional layers followed by 3 fully connected layers
- Used ReLU for the non-linear activations, instead of a tanh or sigmoid
- Reduced over-fitting by using a dropout layer after each fully connected layer

AlexNet



AlexNet architecture

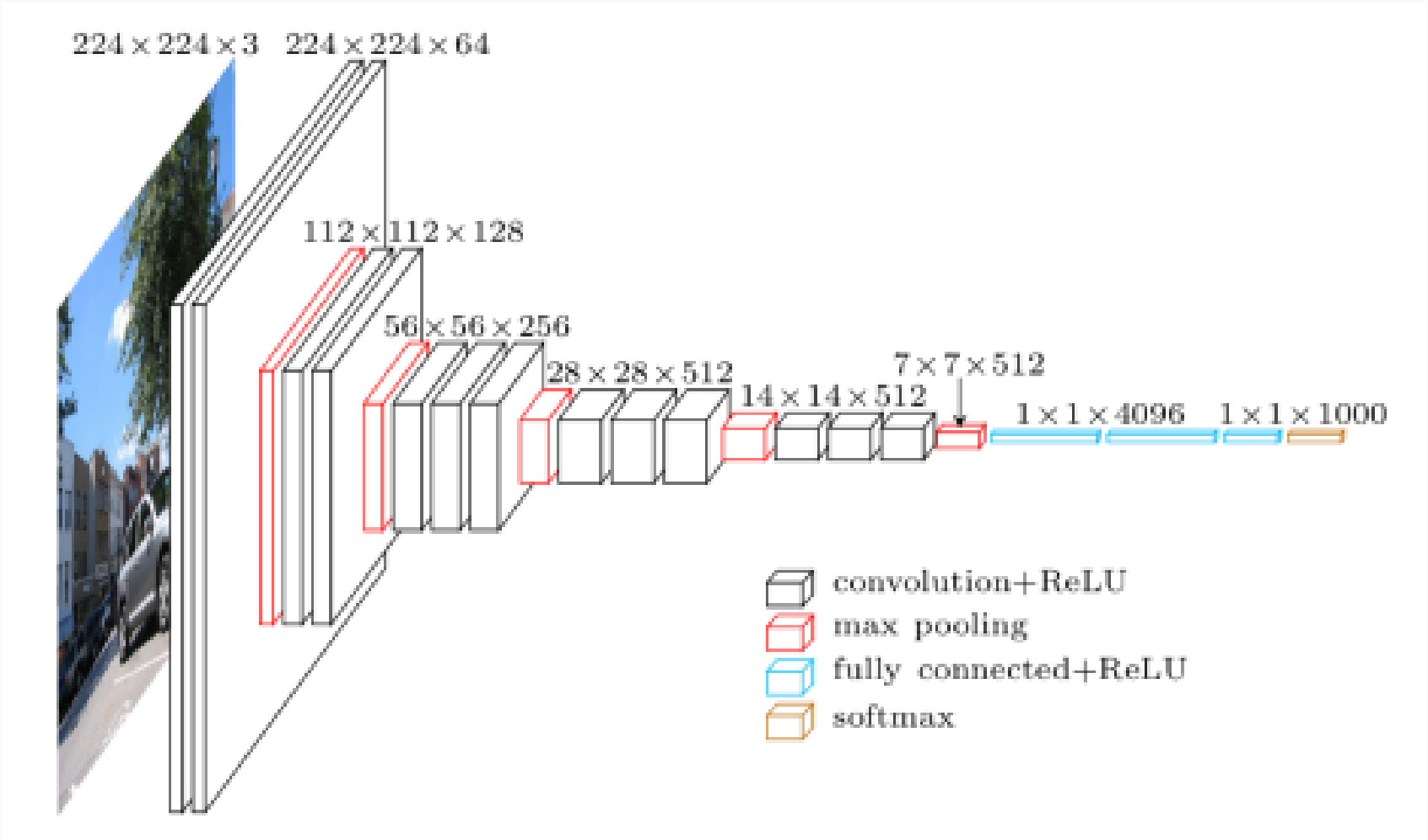


(a) Standard Neural Net (b) After applying dropout.

VGG

- Developed by the VGG (Visual Geometry Group) in Oxford
- Improves AlexNet by replacing the large filters in the first two layers by multiple 3x3 filters
- Multiple stacked smaller size kernels allows the network to learn more complex features
- Also reduces number of parameters
- VGG convolutional layers are followed by 3 fully connected layers
- Width of the network (number of channels) starts small at 64 and increases by a factor of 2 after every pooling layer
- Achieved the top-5 error rate of 7.3% on ImageNet in 2014

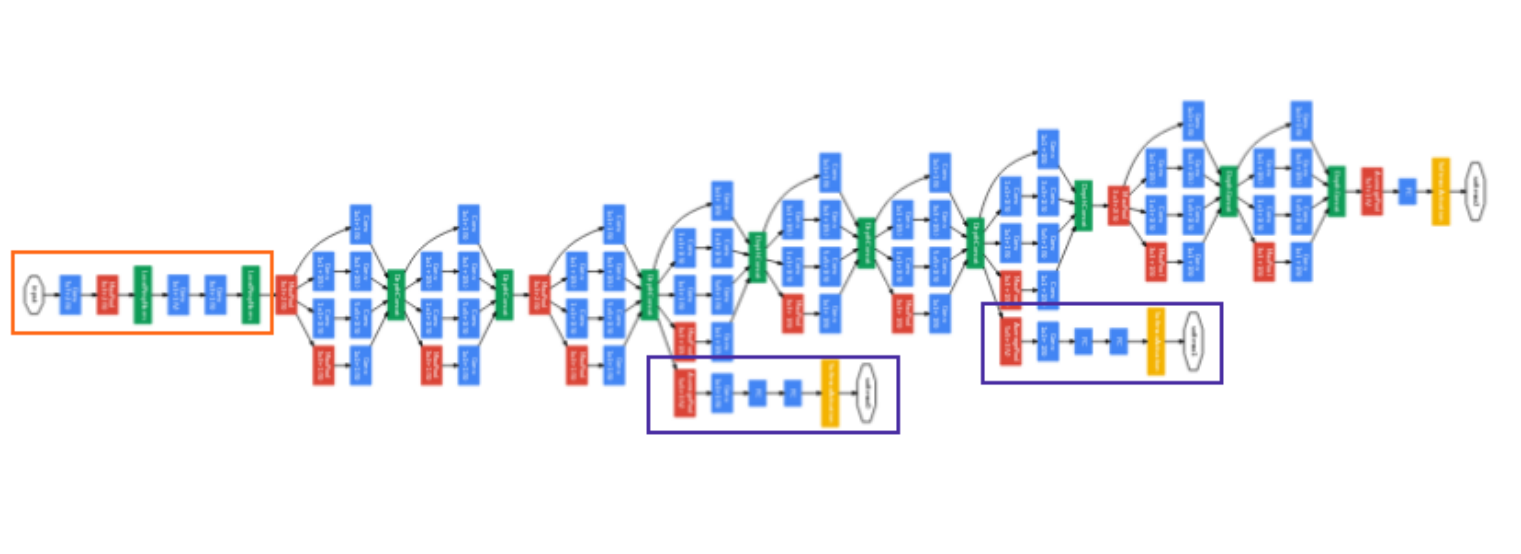
VGG-16


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

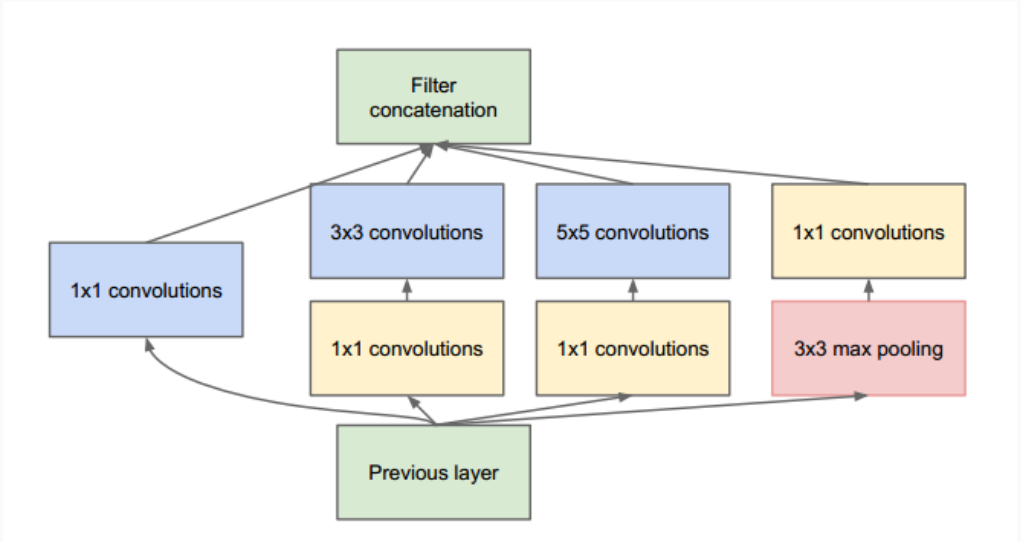
GoogLeNet/Inception

- VGG is computationally demanding due to large width of convolutional layers
- Insight of GoogLeNet is that many activations are zero or highly correlated
- Introduces the inception module that approximates a sparse CNN
- Uses convolutions of different sizes to capture details at various scales
- Replaces fully connected layers with global average pooling
- Achieves 6.67% top-5 error rate on ImageNet in 2014 and is much faster than VGG

GoogLeNet/Inception



GoogLeNet architecture, including auxiliary losses to mitigate vanishing gradients



Inception module

Residual networks

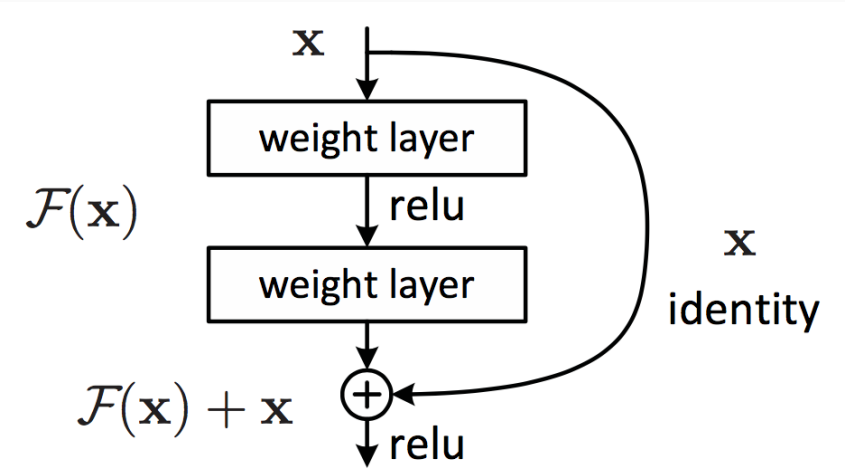
- Previous developments show increased depth increases capacity
- Backpropagated signal becomes vanishingly small with large depth
- Increased parameters also present optimisation problems
- Proposed solution is to introduce **residual blocks** into the architecture, that use shortcut connections to propagate the signal
- Assumption is that the optimal learned function is closer to identity than to zero
- Similar to GoogLeNet, uses a global average pooling followed by the classification layer
- ResNets were learned with network depth of as large as 152
- Similar to the VGGNet, consisting mostly of 3X3 filters
- ResNet-152 achieved 3.6% top-5 error rate on ImageNet in 2015, surpassing human performance

Residual block

- Residual blocks add the input to the output:

$$\mathbf{y} = \sigma(\mathcal{F}(\mathbf{x}) + \mathbf{x}),$$

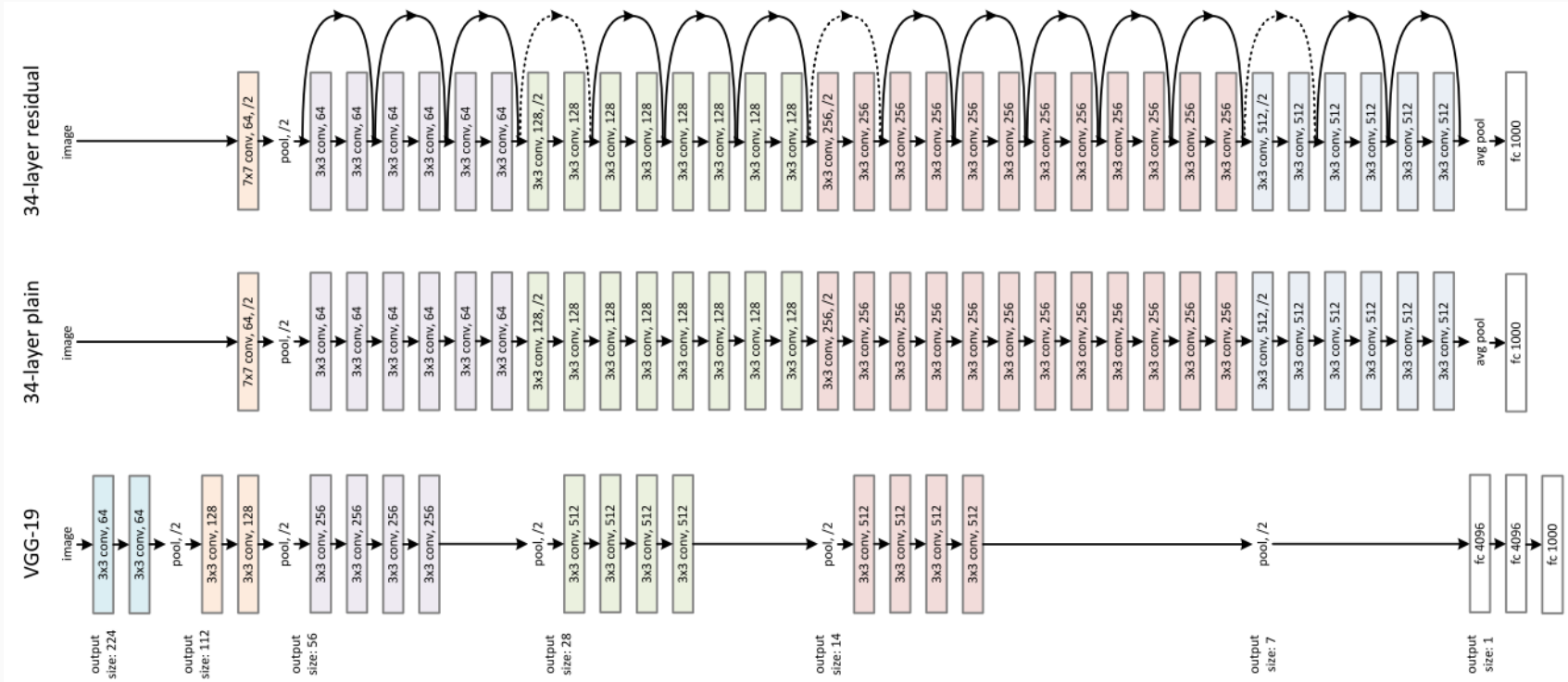
where $\mathcal{F}(\mathbf{x}) = W_2\sigma(W_1\mathbf{x} + b_1) + b_2$



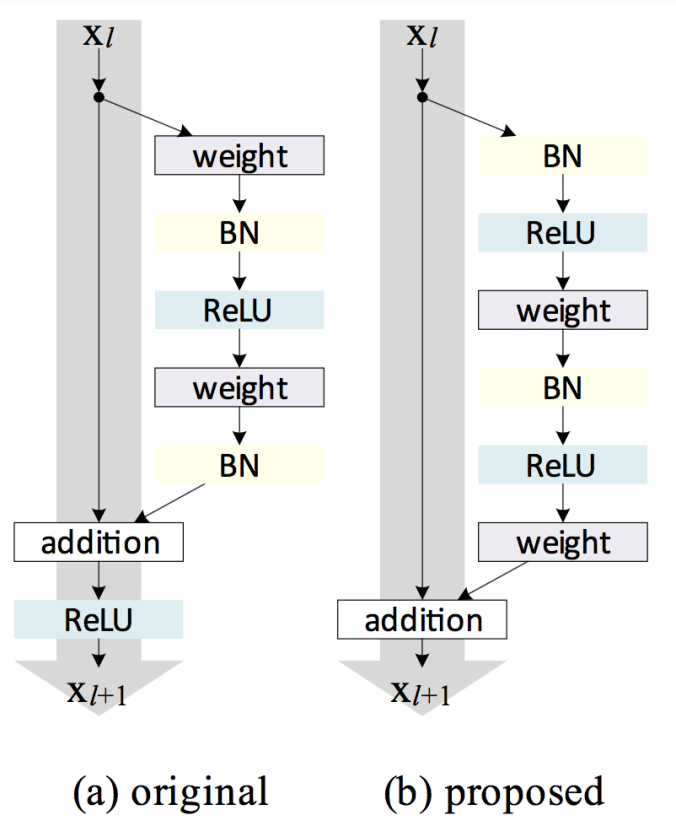
- Two layers are included inside the block, since one layer would be too close to a linear transformation:

$$\tilde{\mathbf{y}} = \sigma(W_1\mathbf{x} + b_1 + \mathbf{x})$$

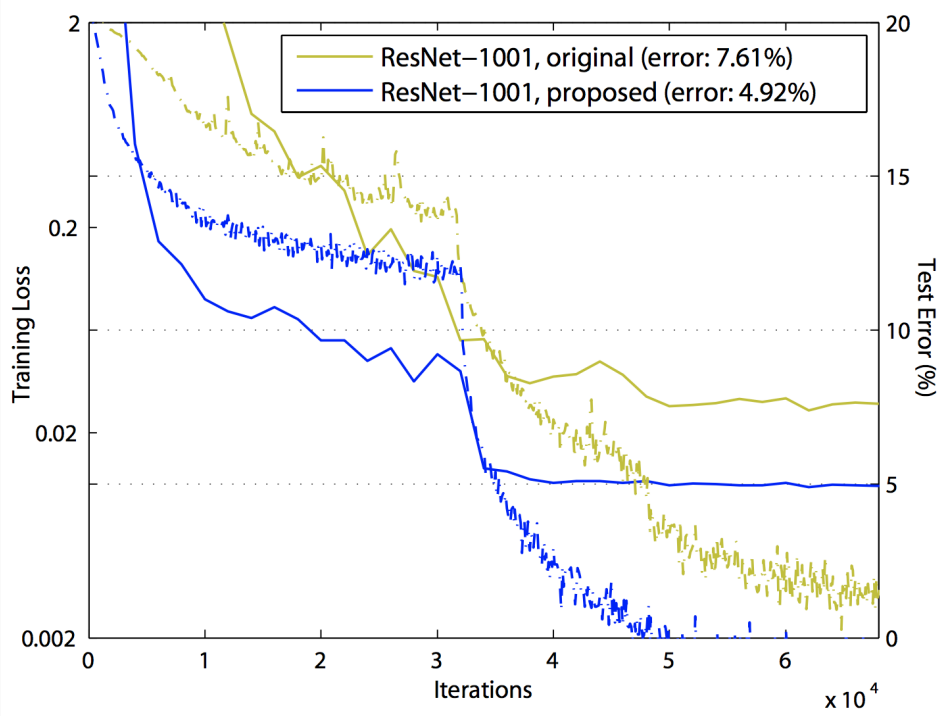
ResNet, plain, VGG-19



Revised ResNet (2016)



Original and revised ResNet blocks



Training curves on CIFAR-10 of 1001-layer ResNets. Solid lines denote test error, and dashed lines denote training loss

CNN architectures and ILSVRC

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

ImageNet competition progress