Convolutional Neural Networks

Training course: Machine learning for weather prediction

Ana Prieto Nemesio

ECMWF Bonn

ana.prietonemesio@ecmwf.int

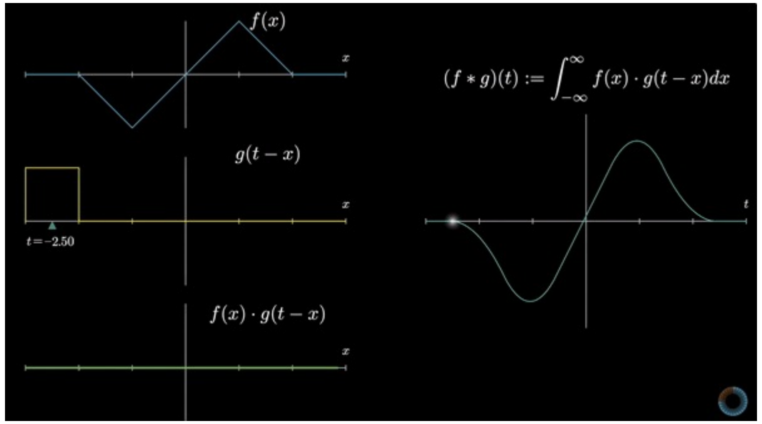
ECMWF

Outline

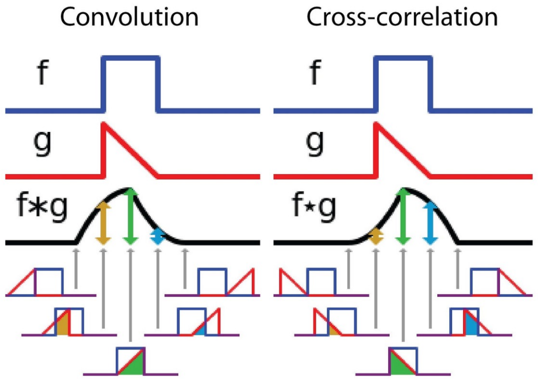
- What is a convolution?
- Advantages of convolutional layer when using spatial/multidimensional data
- Convolution’s arithmetic
- Building a Convolutional Neural Network
    - Key concepts
- Popular CNN-based architectures - ResNets, U-nets

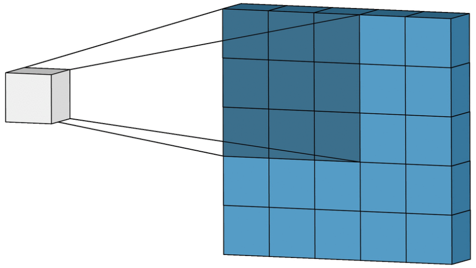
What is a convolution?

What is a convolution?



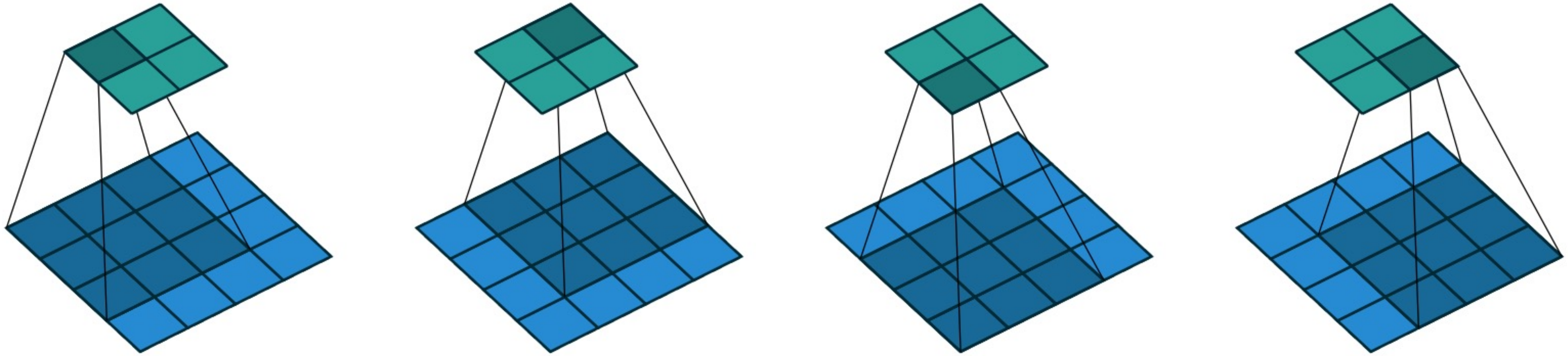
But what is a convolution?
3Blue1Brown





Intuitively Understanding Convolutions for Deep Learning
Towards Data Science

What is a convolution?



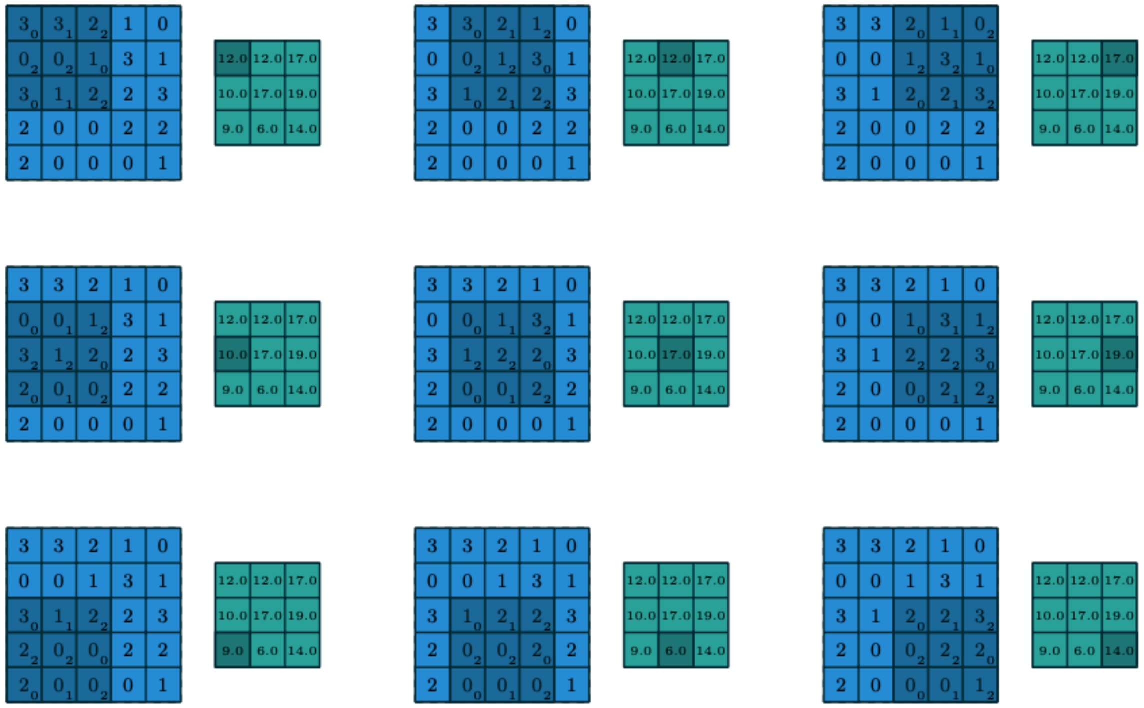
A guide to convolution arithmetic for deep Learning
Dumoulin V., Visin. F, 2018, arXiv:1603.07285

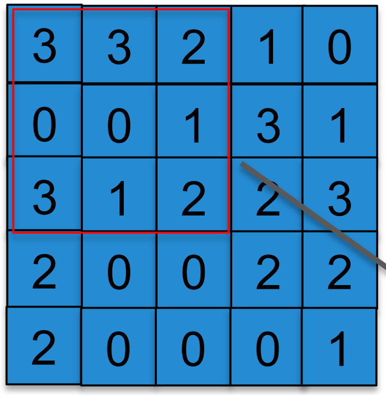
What is a convolution?

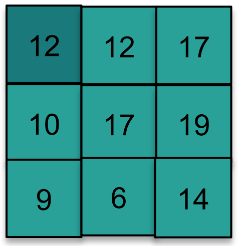
Kernel or Filter

Input Feature map

Output Activation Map





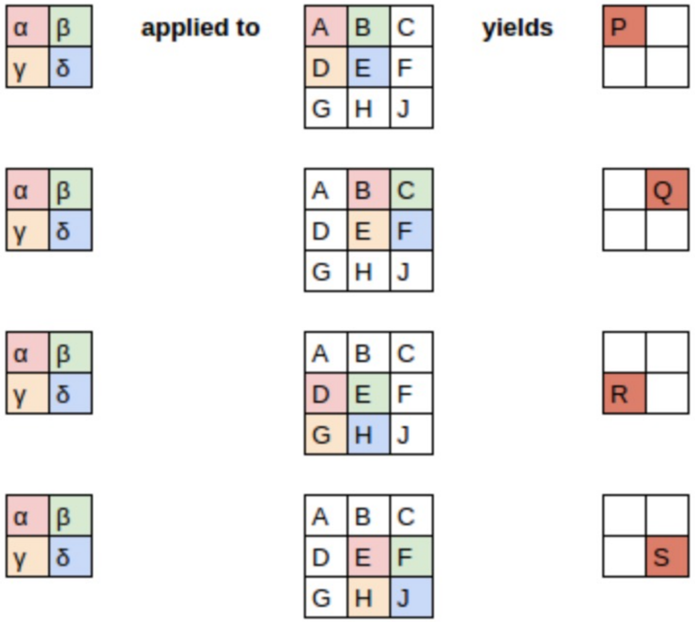


$$
\begin{aligned}
&= 0 * 3 + 1 * 3 + 2 * 2 \\
&+ 2 * 0 + 2 * 0 + 0 * 1 \\
&+ 0 * 3 + 1 * 1 + 2 * 2 \\
&= 12
\end{aligned}
$$

Convolve a filter with the image = spatially sliding it over the image and computing the dot product

What is a convolution?

Convolutions are still linear transforms



$\alpha * A + \beta * B + \gamma * D + \delta * E + b = P$

$\alpha * B + \beta * C + \gamma * E + \delta * F + b = Q$

$\alpha * D + \beta * E + \gamma * G + \delta * H + b = R$

$\alpha * E + \beta * F + \gamma * H + \delta * J + b = S$

$\alpha A+\beta B+\gamma D+\delta E+b = P$

$\alpha B+\beta C+\gamma E+\delta F+b = Q$

$\alpha D+\beta E+\gamma G+\delta H+b = R$

$\alpha E+\beta F+\gamma H+\delta J+b = S$

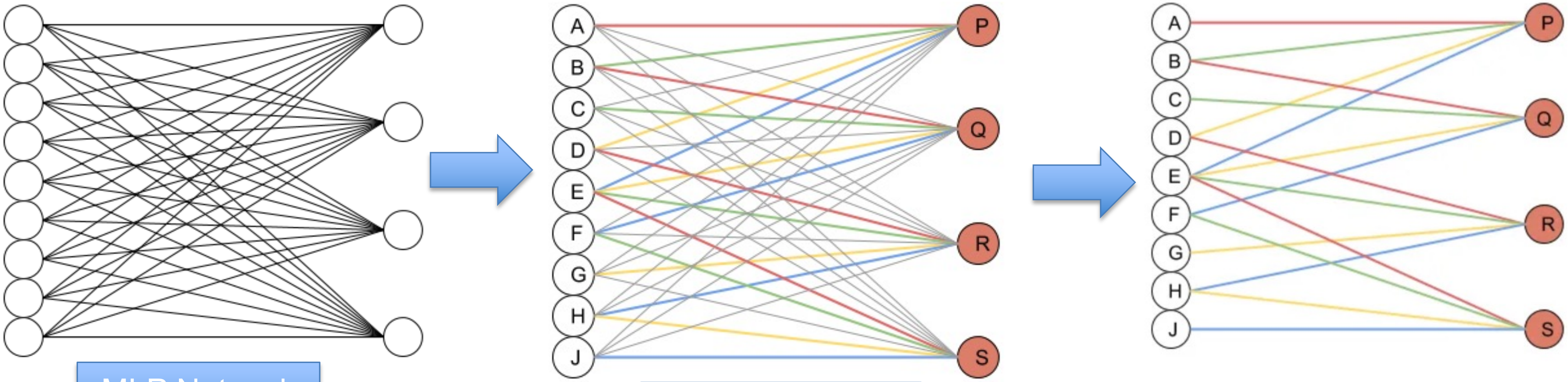
$$Y = \sum (weight * input) + bias$$

$$\begin{array}{c} \begin{bmatrix} \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta & 0 \\ 0 & 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta \end{bmatrix} \\ \begin{matrix} A & B & C & D & E & F & G & H & J \end{matrix} \end{array} * \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ J \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+0C+\gamma D+\delta E+0F+0G+0H+0J+b \\ 0A+\alpha B+\beta C+0D+\gamma E+\delta F+0G+0H+0J+b \\ 0A+0B+0C+\alpha D+\beta E+0F+\gamma G+\delta H+0J+b \\ 0A+0B+0C+0D+\alpha E+\beta F+0G+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+\gamma D+\delta E+b \\ \alpha B+\beta C+\gamma E+\delta F+b \\ \alpha D+\beta E+\gamma G+\delta H+b \\ \alpha E+\beta F+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} P \\ Q \\ R \\ S \end{bmatrix}$$

CNN from different viewpoints
Matt Kleinsmith

Convolutions can be seen as *special* type of matrix multiplication

What is a convolution?



MLP Network

MLP Network
(for previous example)

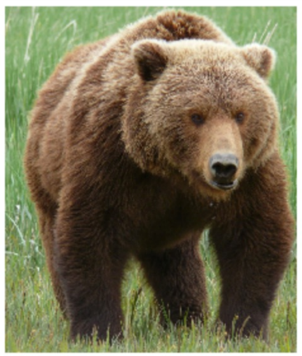
CNN Network
(for previous example)

sparsity

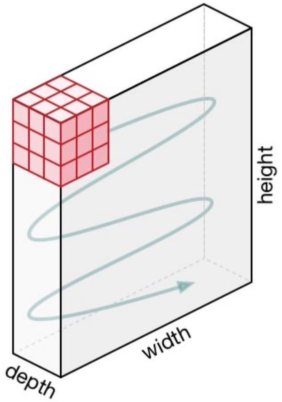
weight-sharing

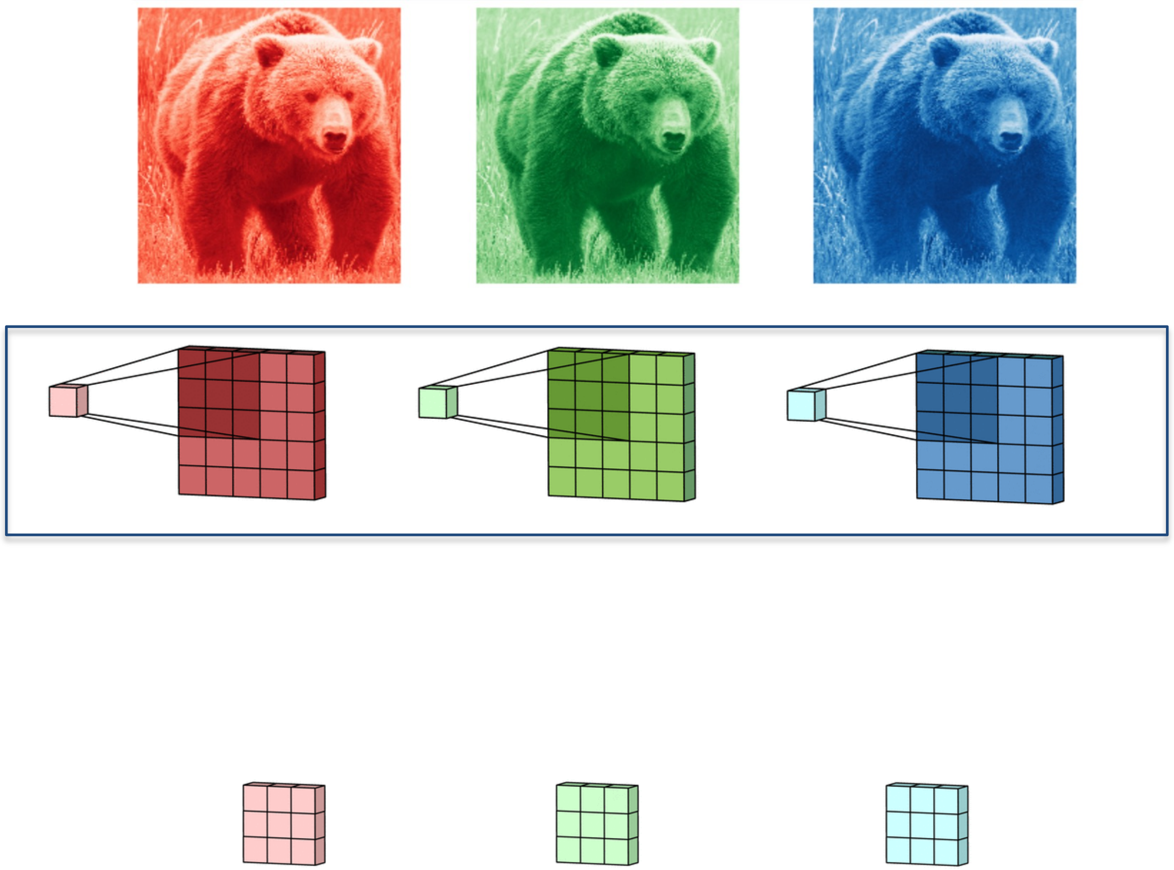
CNN from different viewpoints
Matt Kleinsmith

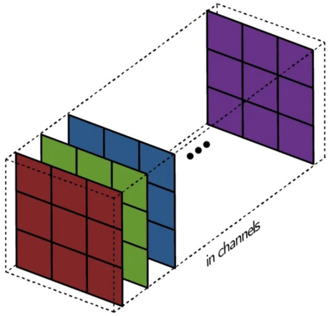
What is a convolution?



RGB Image



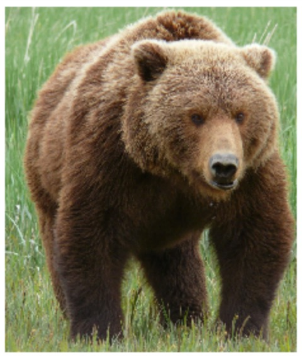




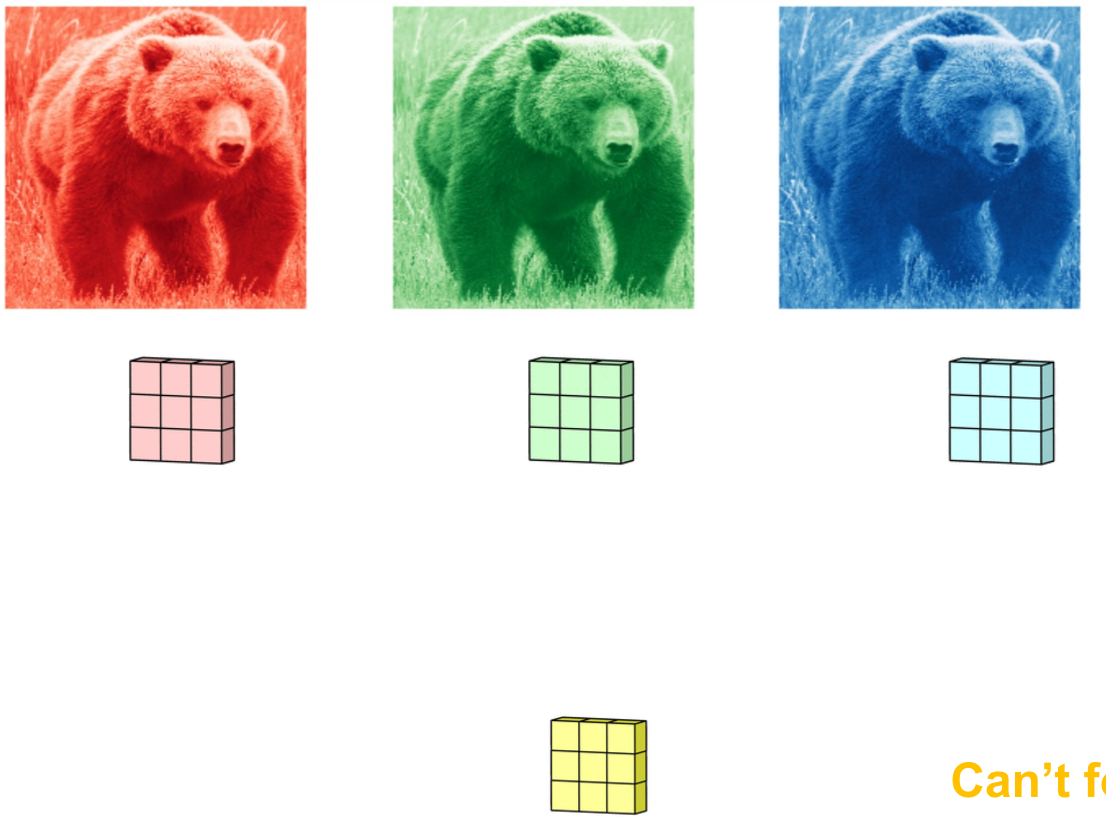
1 filter with
3 kernels

Intuitively Understanding Convolutions for Deep Learning
Towards Data Science

What is a convolution?



RGB Image



Can't forget the Bias term!

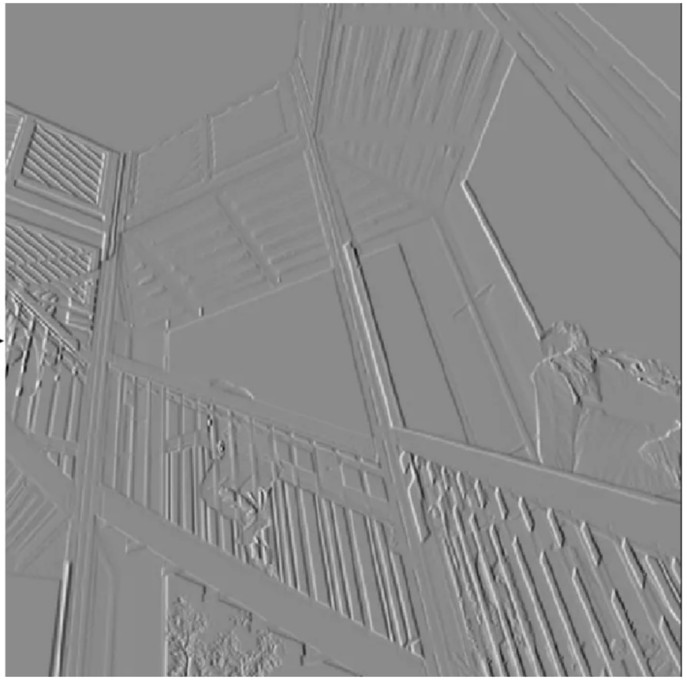
Intuitively Understanding Convolutions for Deep Learning
Towards Data Science

What is a convolution?

Automatic Feature Extraction

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Horizontal Sobel kernel



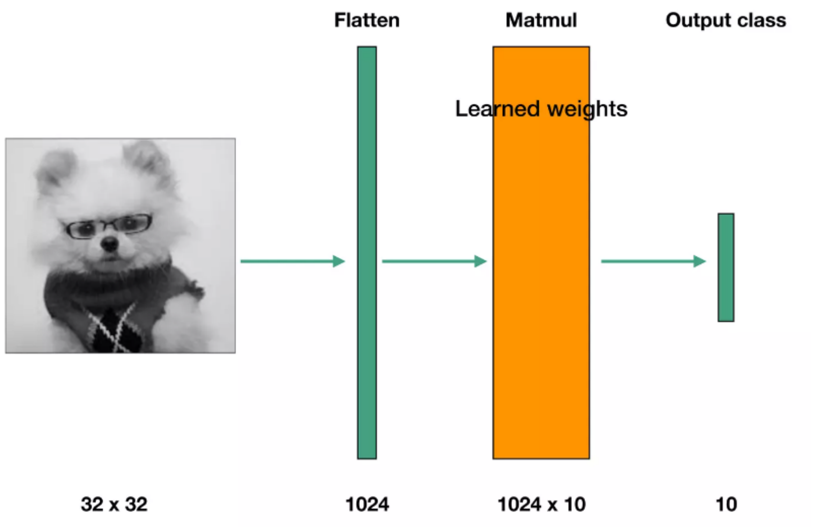
Applying a vertical edge detector kernel

https://setosa.io/ev/image-kernels/

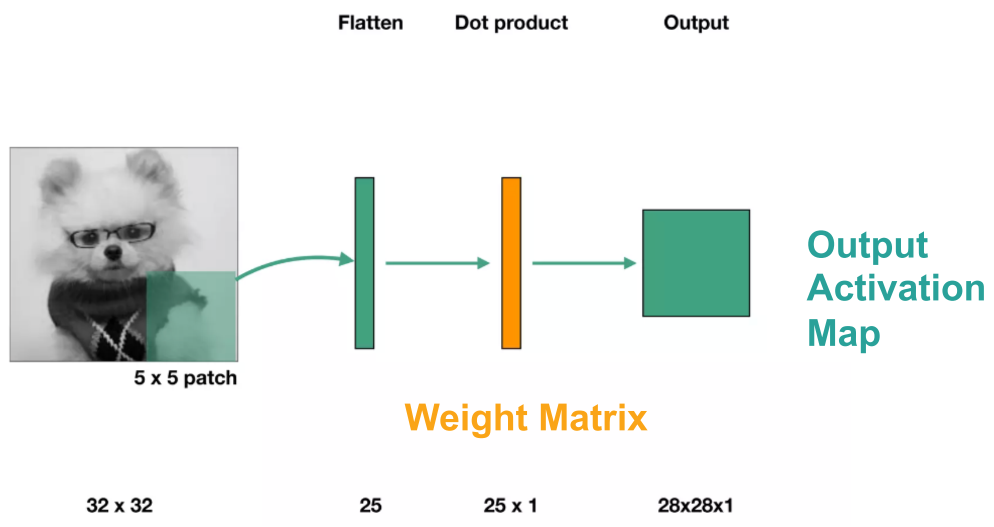
Advantages of Convolutional Layers when using multidimensional data

Advantages of Convolutional Layers when using multidimensional data

MLP(FC) Architecture



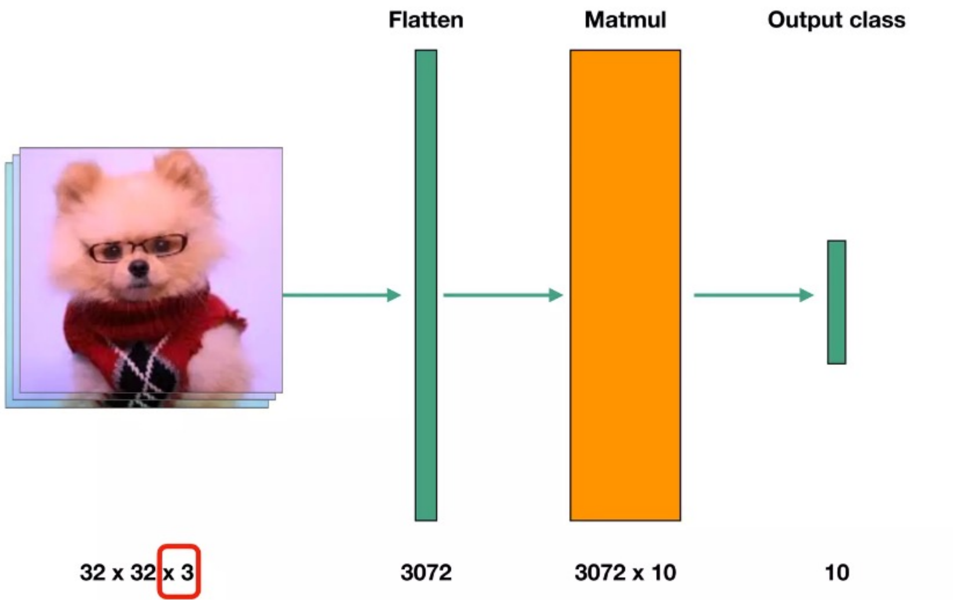
CNN Architecture



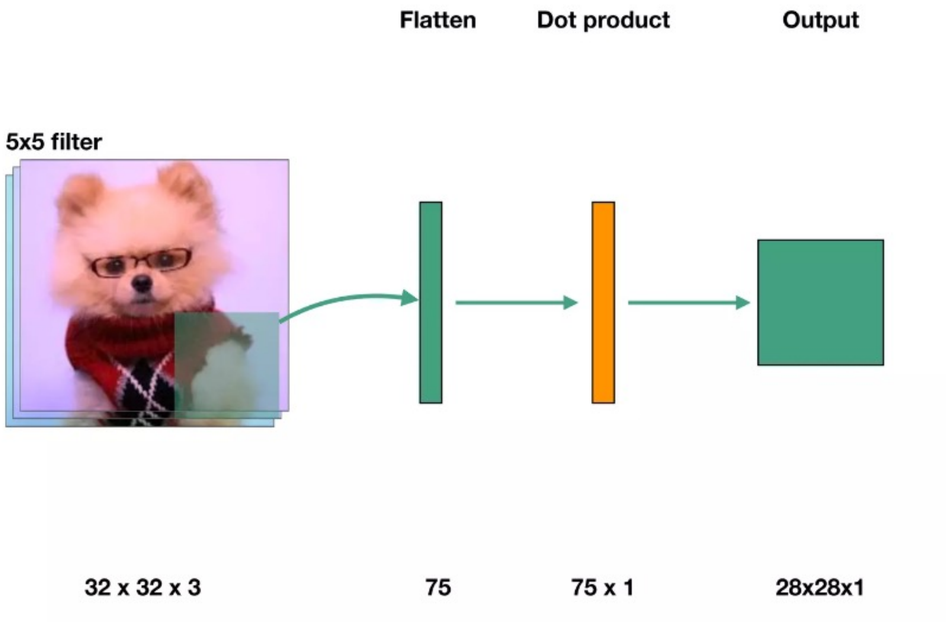
- Poor scaling with image size
- Inefficient weight use – no "weight sharing"
- FC do not provide translation invariance nor equivariance

Advantages of Convolutional Layers when using multidimensional data

MLP (FC) Architecture



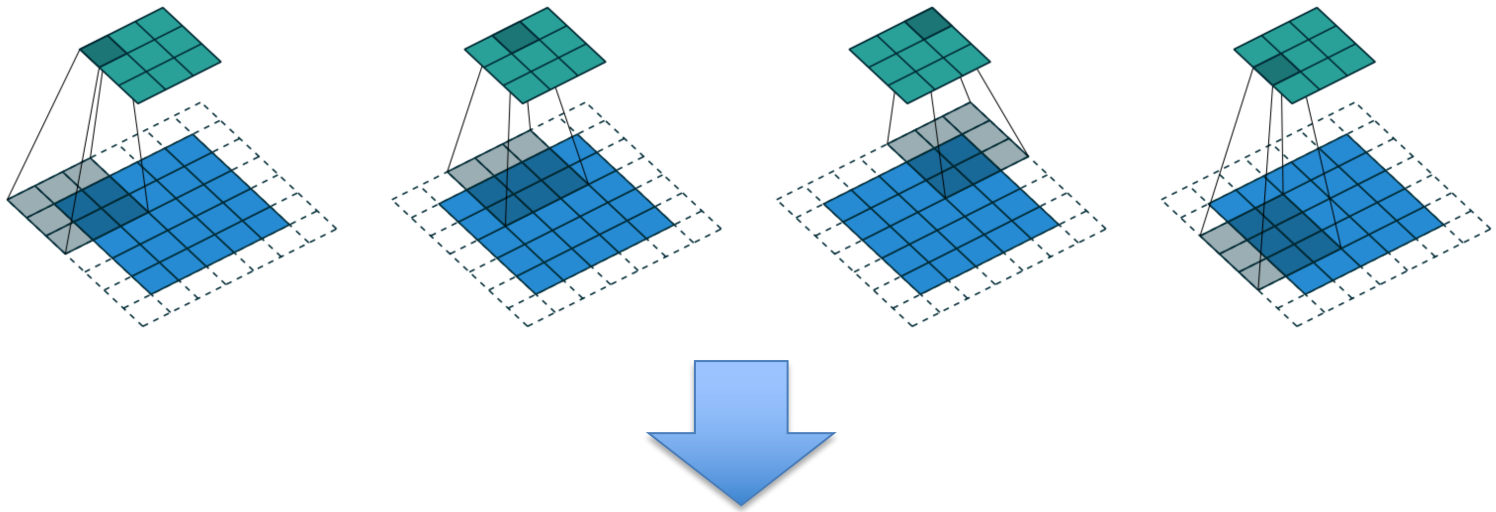
CNN Architecture

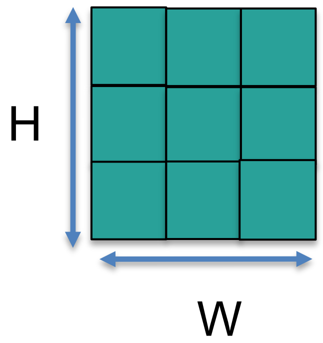


Lecture 2A: Convolutional Neural Networks (Full Stack Deep Learning - Spring 2021)

Convolution’s Arithmetic

Convolution's arithmetic



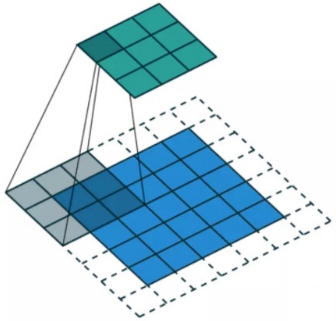


The shape of the output feature map (W,H) is defined based on:

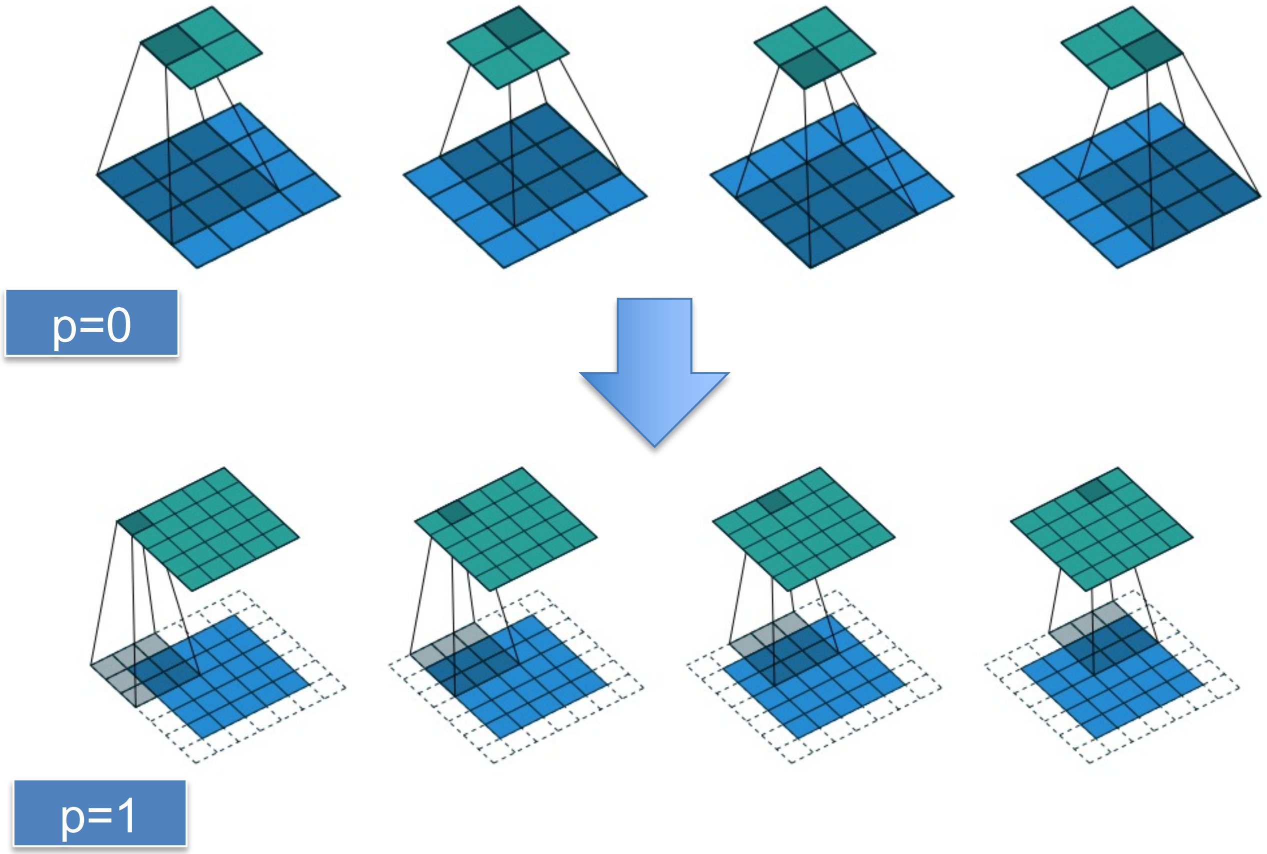
- Shape of the input feature map (W,H)
- The Kernel size (w,h)
- The stride (s)
- The padding (p)

Convolution's arithmetic

Padding



- Padding solves the problem of filters running out of image
- Done by adding extra rows/cols to the input (usually set to 0)



Convolution’s arithmetic

Padding

Padding mode:

See `torch.nn.CircularPad2d`, `torch.nn.ConstantPad2d`, `torch.nn.ReflectionPad2d`, and `torch.nn.ReplicationPad2d` for concrete examples on how each of the padding modes works. Constant padding is implemented for arbitrary dimensions. Circular, replicate and reflection padding are implemented for padding the last 3 dimensions of a 4D or 5D input tensor, the last 2 dimensions of a 3D or 4D input tensor, or the last dimension of a 2D or 3D input tensor.

Replication Padding

5	5	0	8	7	8	1	1
5	**5**	**0**	**8**	**7**	**8**	**1**	1
1	**1**	**9**	**5**	**0**	**7**	**7**	7
6	**6**	**0**	**2**	**4**	**6**	**6**	6
9	**9**	**7**	**6**	**6**	**8**	**4**	4
8	**8**	**3**	**8**	**5**	**1**	**3**	3
7	**7**	**2**	**7**	**0**	**1**	**0**	0
7	7	2	7	0	1	0	0

Reflection Padding

9	1	9	5	0	7	7	7
0	**5**	**0**	**8**	**7**	**8**	**1**	8
9	**1**	**9**	**5**	**0**	**7**	**7**	7
0	**6**	**0**	**2**	**4**	**6**	**6**	6
7	**9**	**7**	**6**	**6**	**8**	**4**	8
3	**8**	**3**	**8**	**5**	**1**	**3**	1
2	**7**	**2**	**7**	**0**	**1**	**0**	1
3	8	3	8	5	1	3	1

Circular Padding

0	7	2	7	0	1	0	7
1	**5**	**0**	**8**	**7**	**8**	**1**	5
7	**1**	**9**	**5**	**0**	**7**	**7**	1
6	**6**	**0**	**2**	**4**	**6**	**6**	6
4	**9**	**7**	**6**	**6**	**8**	**4**	9
3	**8**	**3**	**8**	**5**	**1**	**3**	8
0	**7**	**2**	**7**	**0**	**1**	**0**	7
1	5	0	8	7	8	1	5

Pytorch Ref

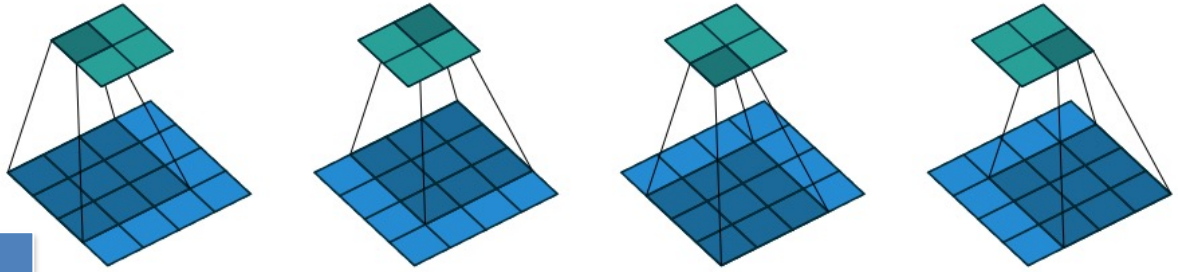
- **padding**: string, either `"valid"` or `"same"` (case-insensitive). `"valid"` means no padding. `"same"` results in padding evenly to the left/right or up/down of the input. When `padding="same"` and `strides=1`, the output has the same size as the input.

Tensorflow Ref

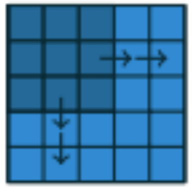
Convolution's arithmetic

Input Feature map 4x4

Stride

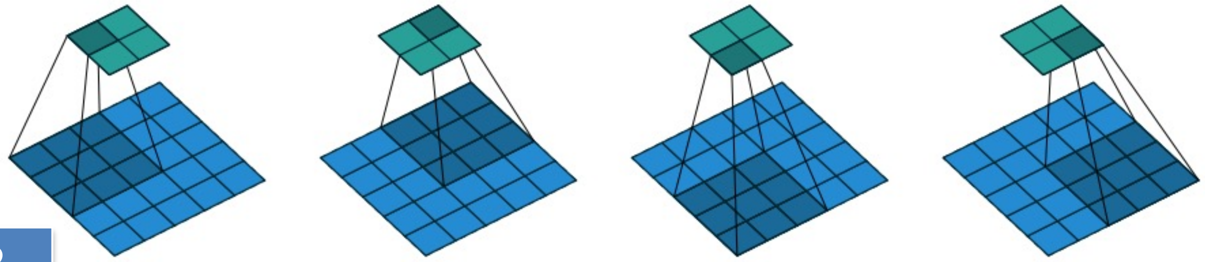


s=1



- Convolutions can subsample the image by jumping across some locations — this is called 'stride'

Input Feature map 5x5



s=2

Convolution's arithmetic

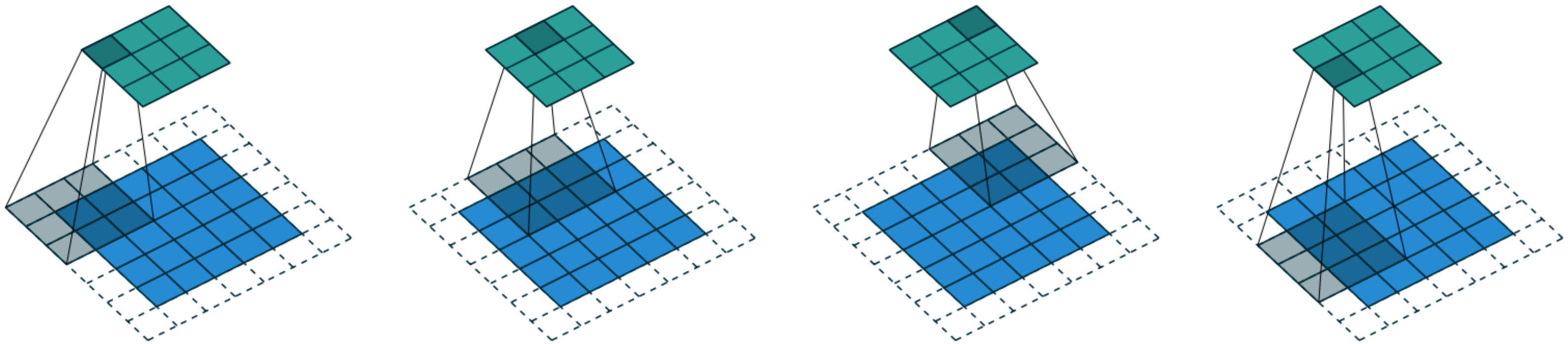
```
import torch.nn as nn
m = nn.Conv2d(in_channels=16, out_channels=33, kernel_size=3, stride=1,padding=1)
m
✓ 0.0s

Conv2d(16, 33, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2\ x\ padding[0] - kernel\_size[0]}{stride[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2\ x\ padding[1] - kernel\_size[1]}{stride[1]} + 1 \right\rfloor$$



Convolution's arithmetic

Padding + Stride

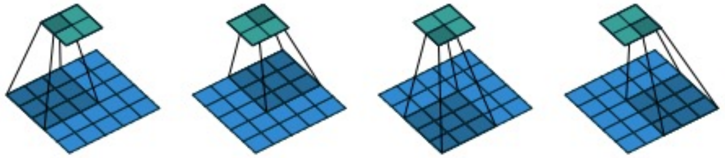


Figure 2.5: (No zero padding, arbitrary strides) Convolving a 3×3 kernel over a 5×5 input using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 0$).

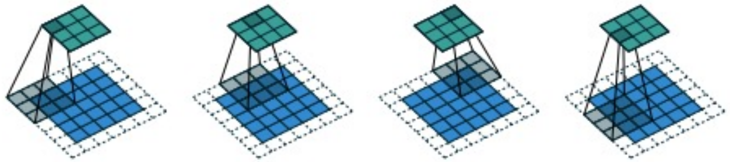


Figure 2.6: (Arbitrary padding and strides) Convolving a 3×3 kernel over a 5×5 input padded with a 1×1 border of zeros using 2×2 strides (i.e., $i = 5$, $k = 3$, $s = 2$ and $p = 1$).

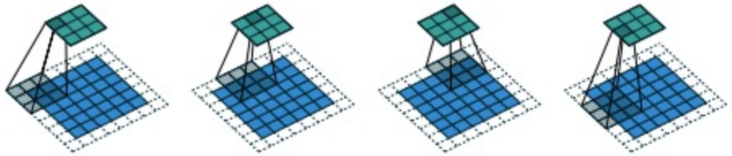


Figure 2.7: (Arbitrary padding and strides) Convolving a 3×3 kernel over a 6×6 input padded with a 1×1 border of zeros using 2×2 strides (i.e., $i = 6$, $k = 3$, $s = 2$ and $p = 1$). In this case, the bottom row and right column of the zero padded input are not covered by the kernel.

A guide to convolution arithmetic for deep Learning
Dumoulin V., Visin. F, 2018, arXiv:1603.07285

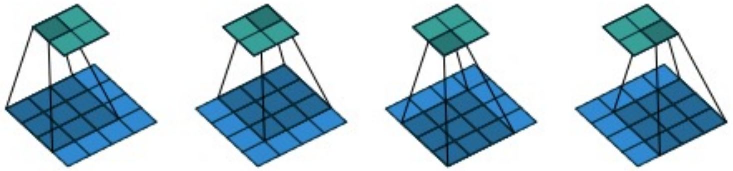


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

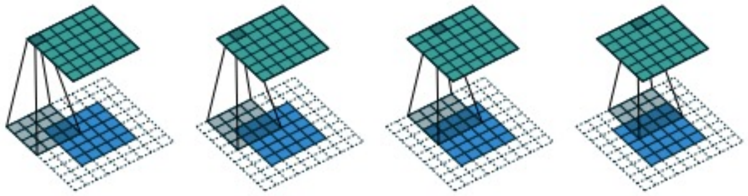


Figure 2.2: (Arbitrary padding, unit strides) Convolving a 4×4 kernel over a 5×5 input padded with a 2×2 border of zeros using unit strides (i.e., $i = 5$, $k = 4$, $s = 1$ and $p = 2$).

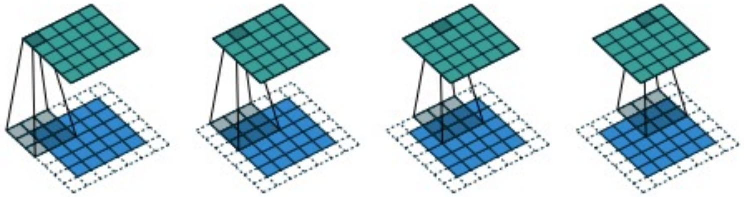


Figure 2.3: (Half padding, unit strides) Convolving a 3×3 kernel over a 5×5 input using half padding and unit strides (i.e., $i = 5$, $k = 3$, $s = 1$ and $p = 1$).

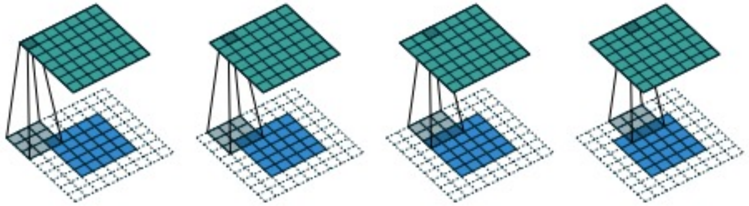
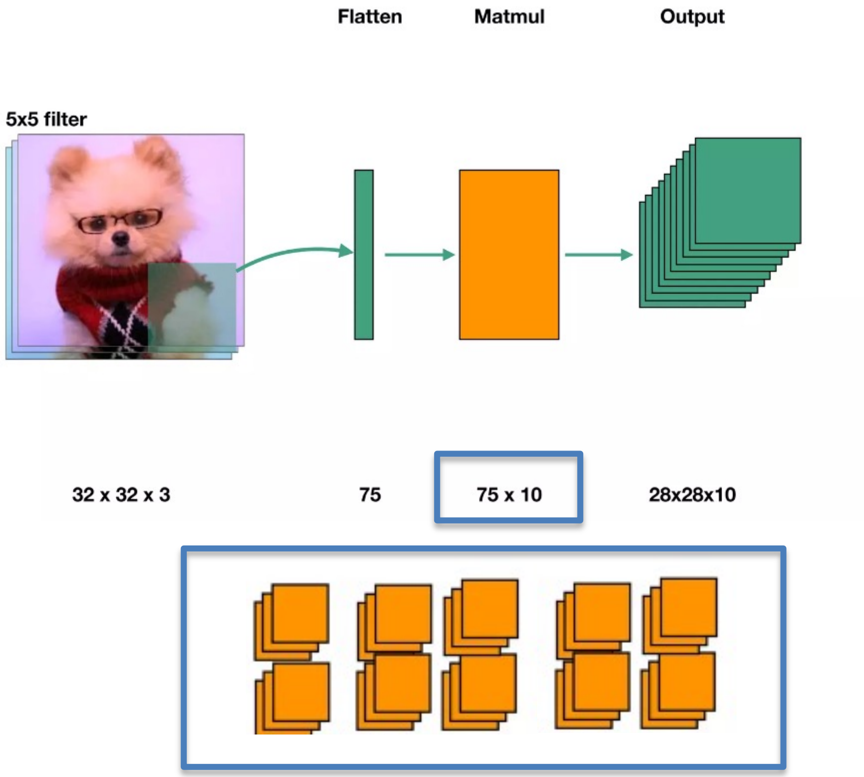


Figure 2.4: (Full padding, unit strides) Convolving a 3×3 kernel over a 5×5 input using full padding and unit strides (i.e., $i = 5$, $k = 3$, $s = 1$ and $p = 2$).

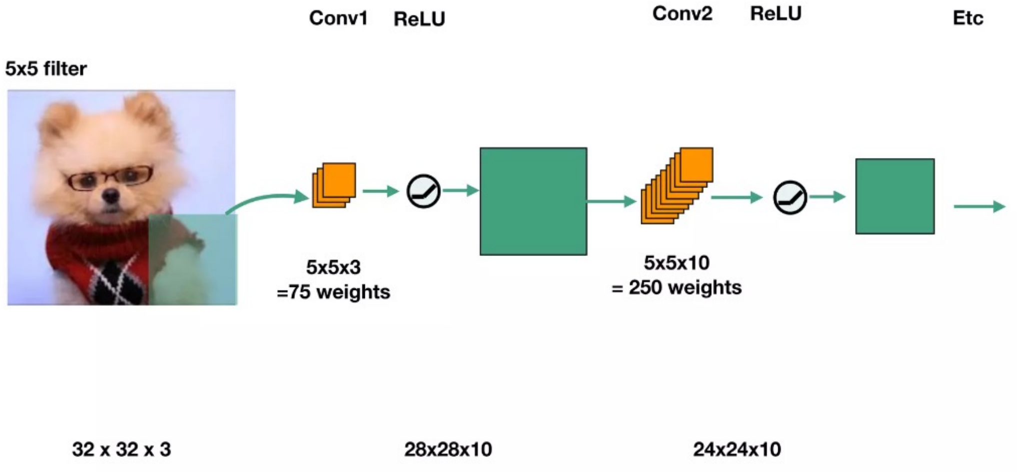
Building a Convolutional Neural Network (CNN)

Building a CNN

Multiple Channel Outputs

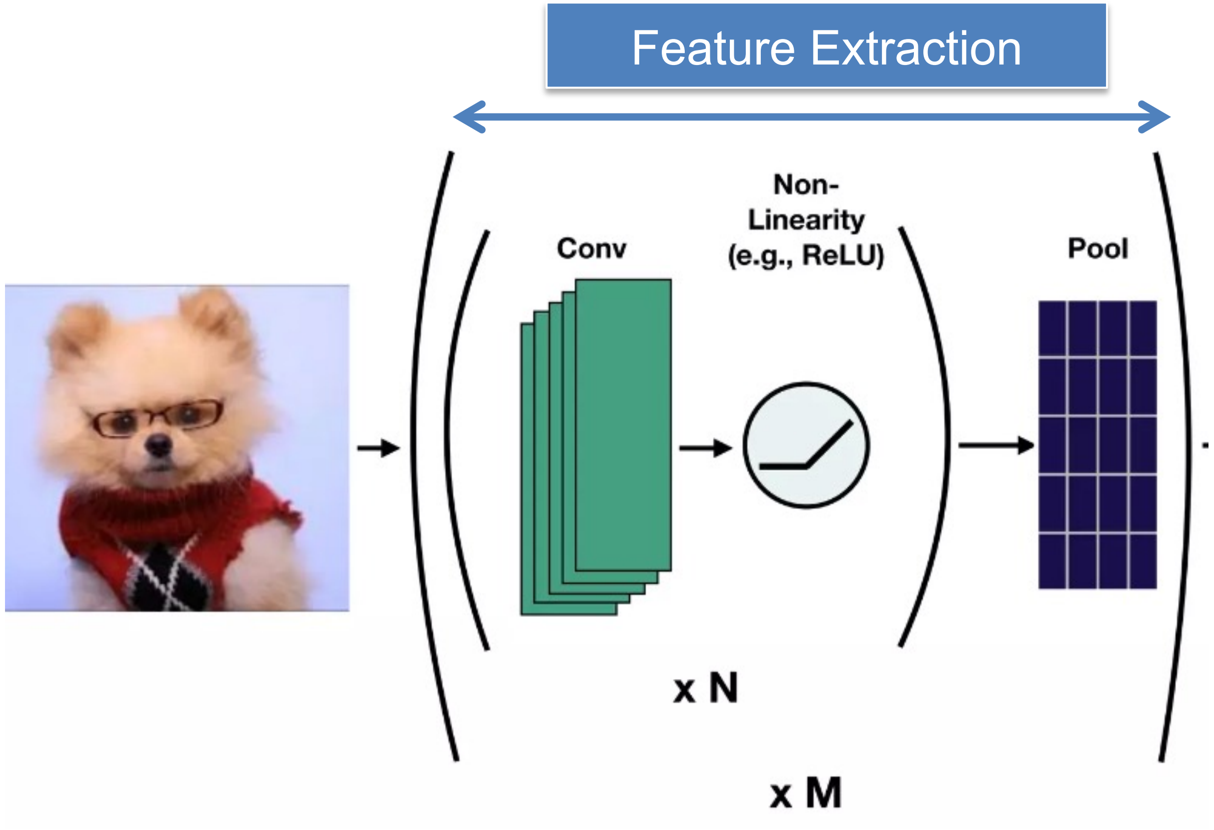


Stacking of Conv Layers



Lecture 2A: Convolutional Neural Networks
(Full Stack Deep Learning - Spring 2021)

Building a CNN

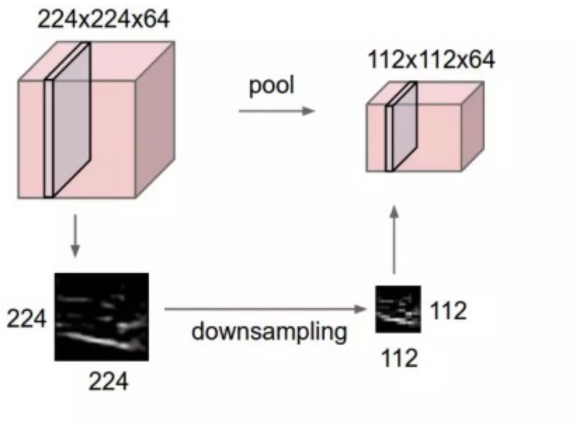


1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

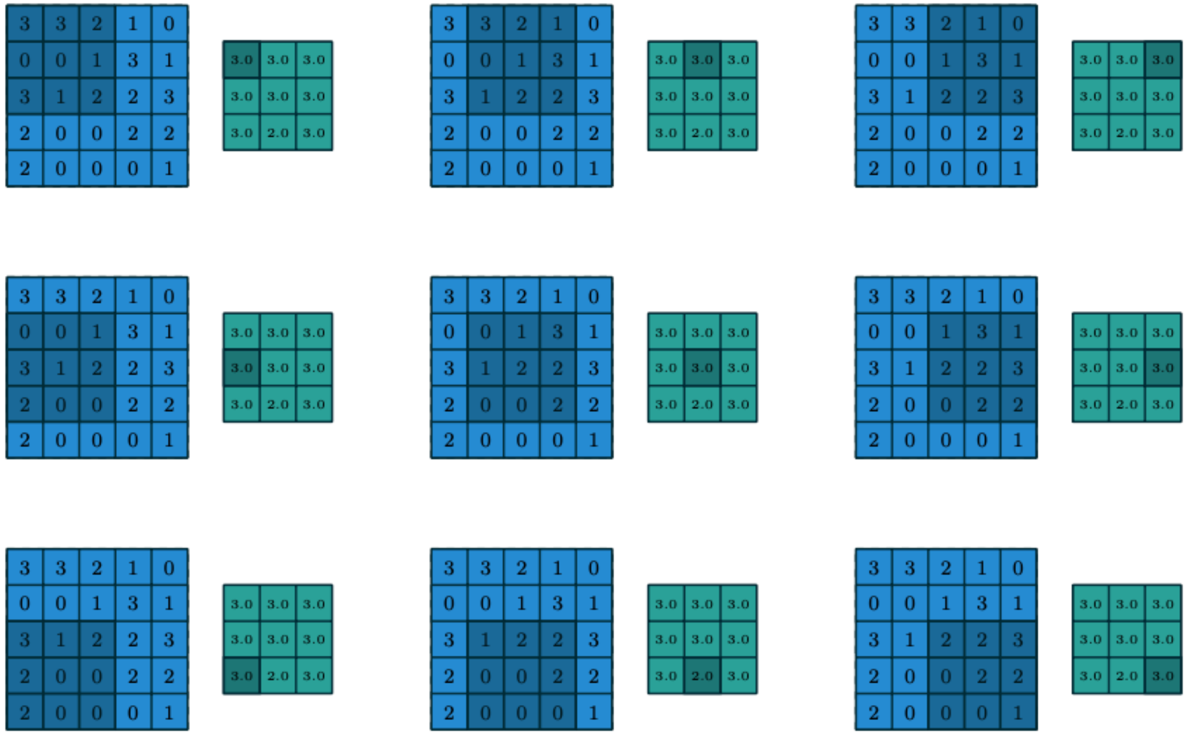
Lecture 2A: Convolutional Neural Networks
(Full Stack Deep Learning - Spring 2021)

Building a CNN

Pooling



Pooling works very much like a discrete convolution, but replaces the linear combination described by the kernel with some other function.



A guide to convolution arithmetic for deep Learning
Dumoulin V., Visin. F, 2018, arXiv:1603.07285

Building a CNN

Pooling → Translation Invariance

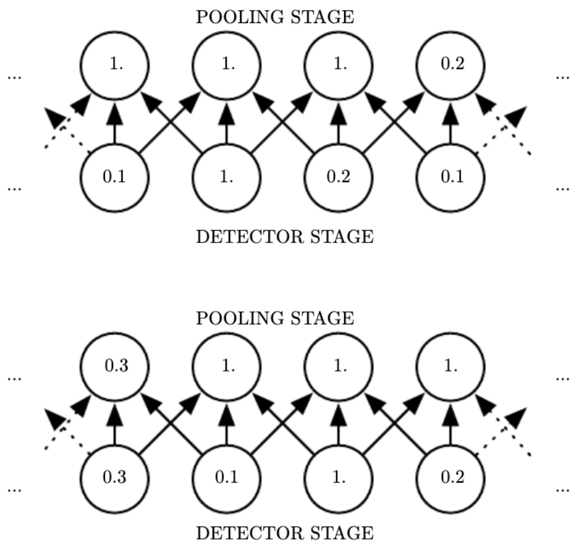
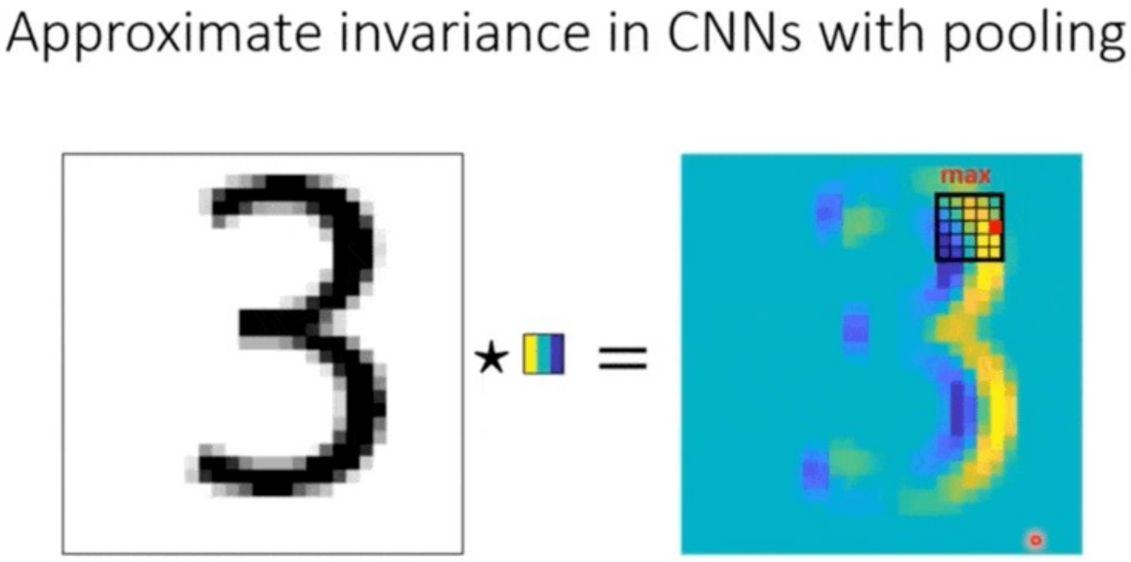


Figure 9.8: Max pooling introduces invariance. *(Top)*A view of the middle of the output of a convolutional layer. The bottom row shows outputs of the nonlinearity. The top row shows the outputs of max pooling, with a stride of one pixel between pooling regions and a pooling region width of three pixels. *(Bottom)*A view of the same network, after the input has been shifted to the right by one pixel. Every value in the bottom row has changed, but only half of the values in the top row have changed, because the max pooling units are sensitive only to the maximum value in the neighborhood, not its exact location.

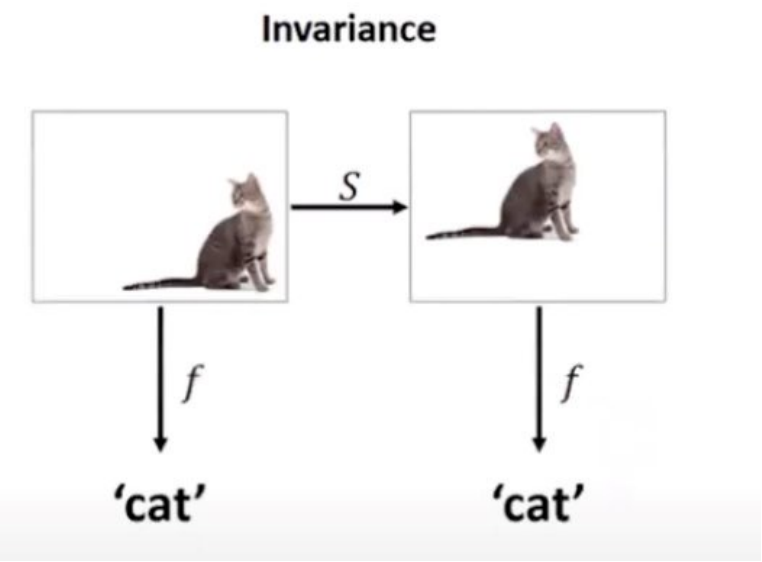


Output of convolutional layer+max pooling (~shift invariant)

05 Imperial's Deep learning course: Equivariance and Invariance
Bernhard Kainz
https://www.youtube.com/watch?v=a4Quhf9NhMY

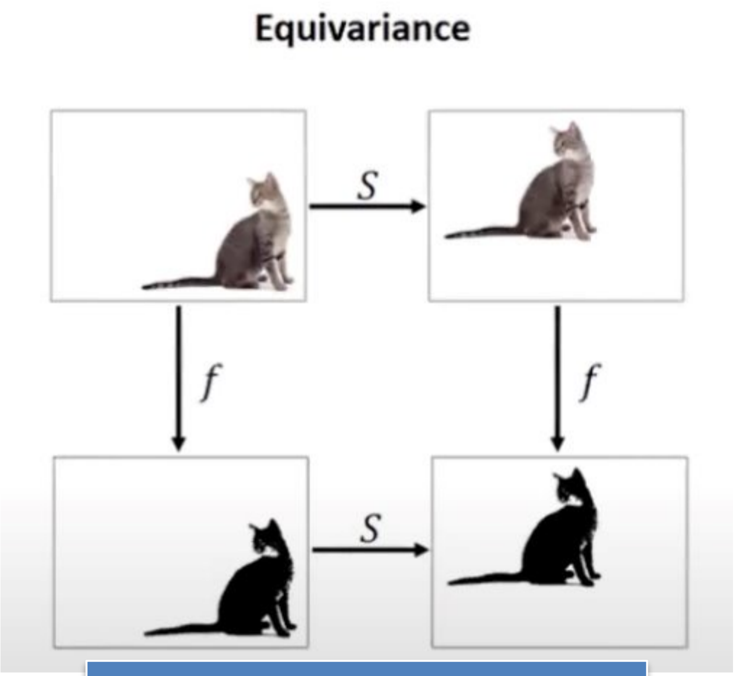
Building a CNN

Translation Invariance



$$f(x) = f(S_v x)$$

Translation Equivariance

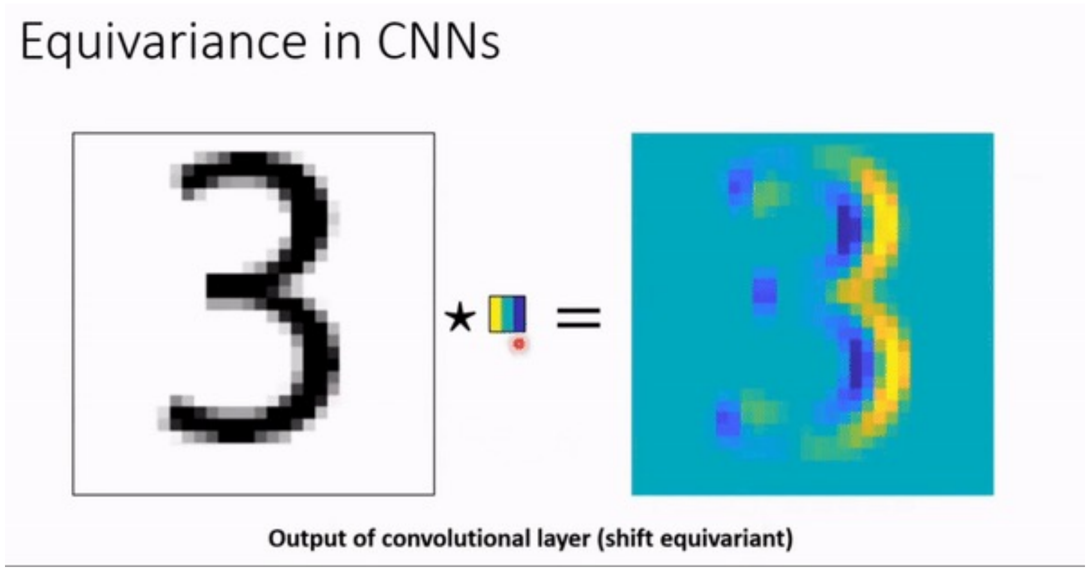


$$S_v f(x) = f(S_v x)$$

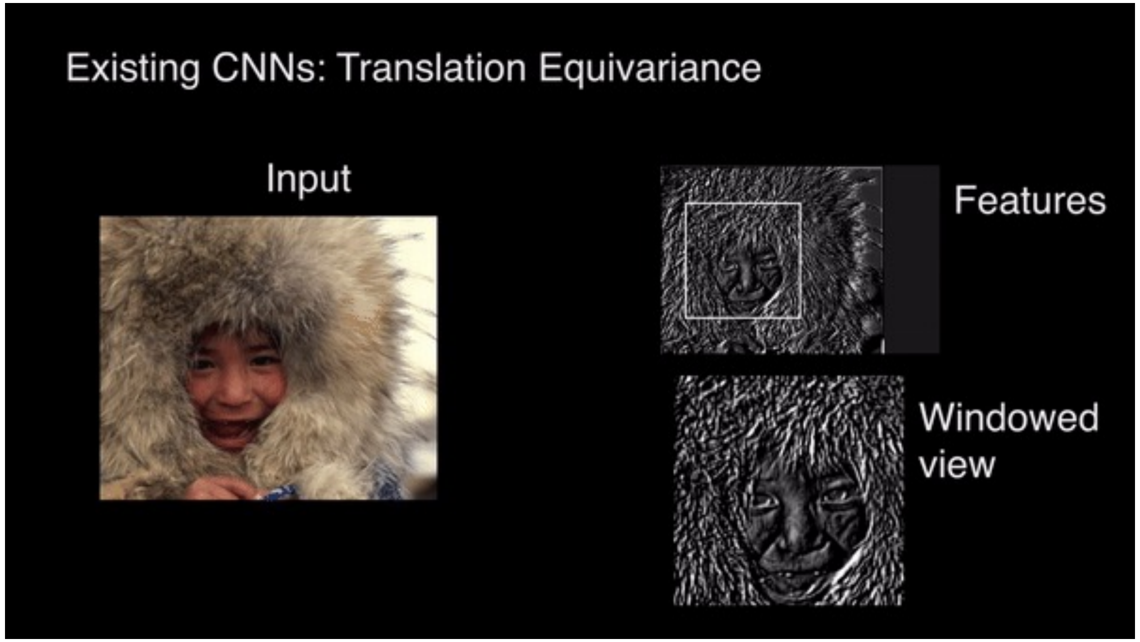
05 Imperial's Deep learning course: Equivariance and Invariance
Bernhard Kainz
https://www.youtube.com/watch?v=a4Quhf9NhMY

Building a CNN

Parameter Sharing → Translation Equivariance



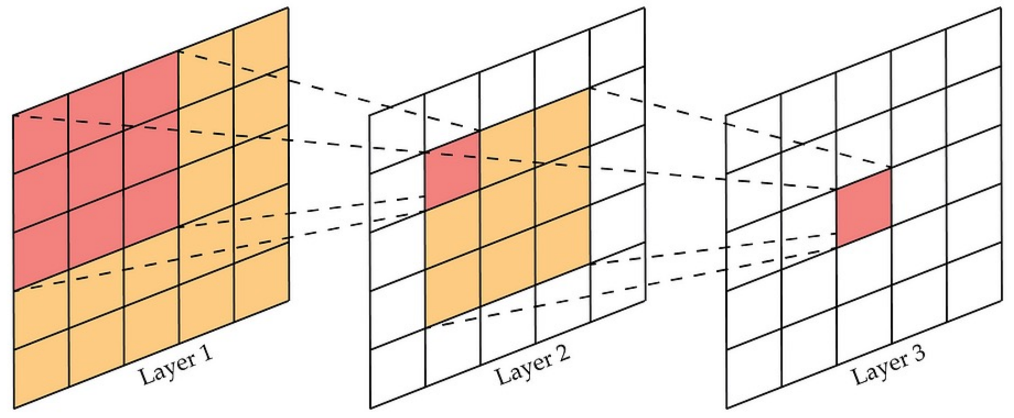
05 Imperial's Deep learning course: Equivariance and Invariance
Bernhard Kainz
https://www.youtube.com/watch?v=a4Quhf9NhMY



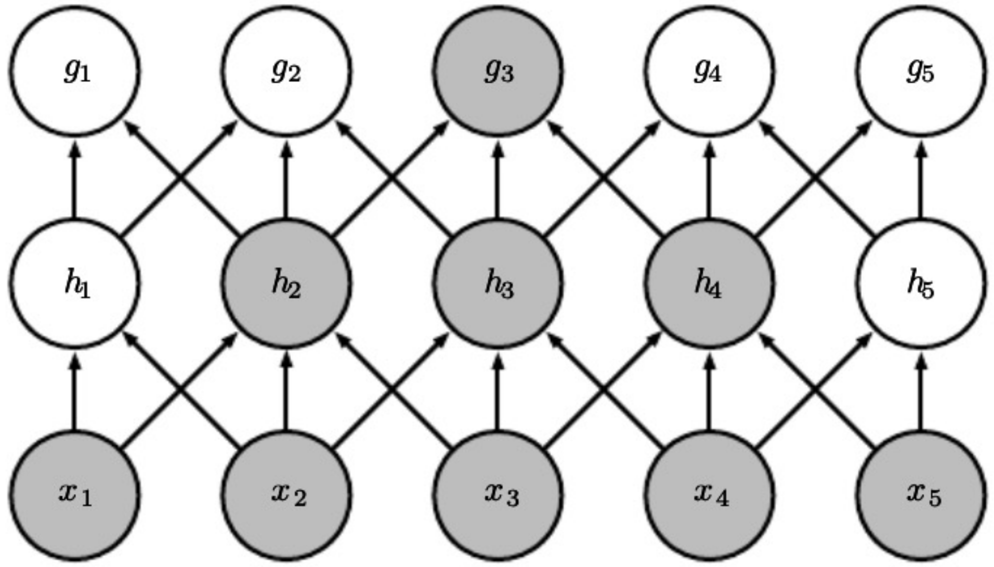
https://www.youtube.com/watchv=a4Quhf9NhMY&t=944s

Building a CNN

Receptive Field

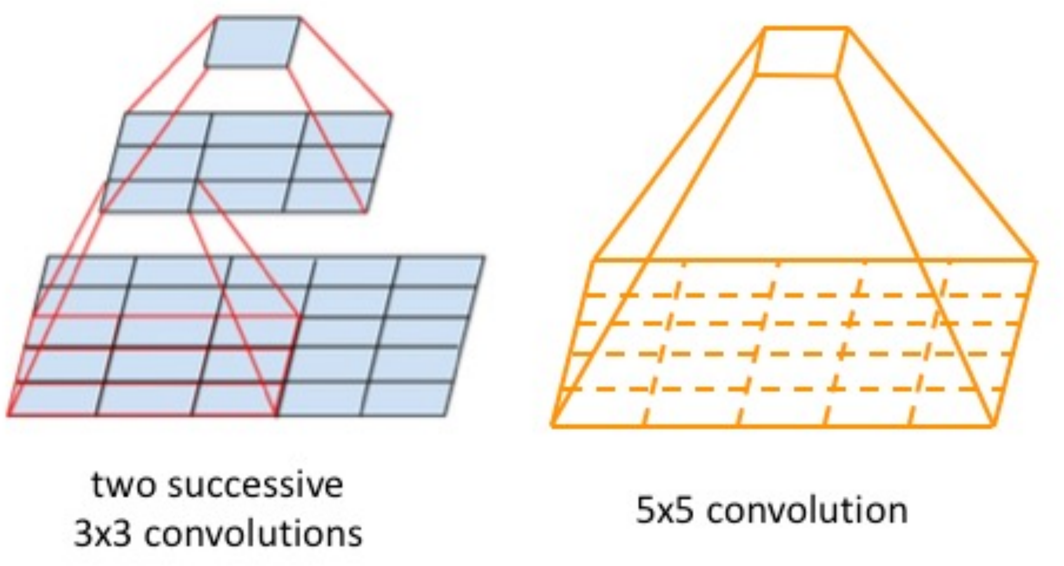


Receptive Field in Convolutional Neural Networks
https://medium.com/@rekalantar/receptive-fields-in-deep-convolutional-networks-43871d2ef2e9



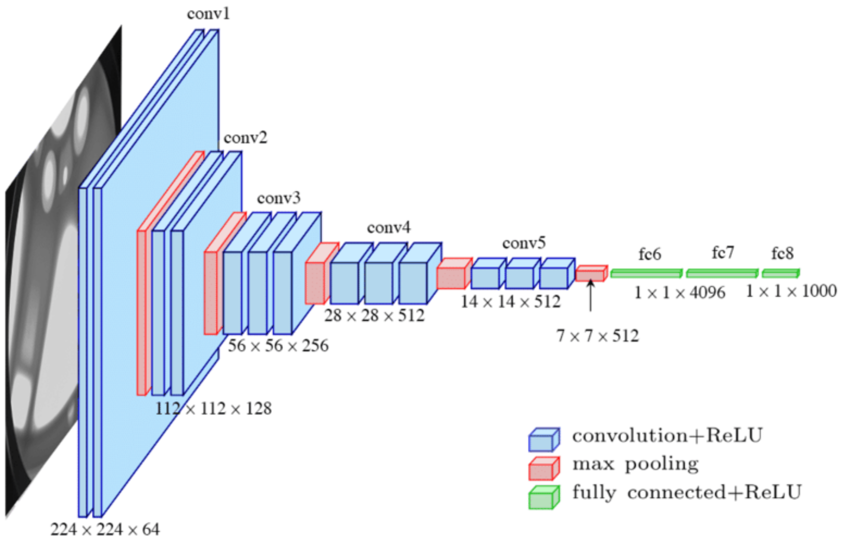
Chapter 9 – Convolutional Neural Networks
Deep Learning Book

Building a CNN



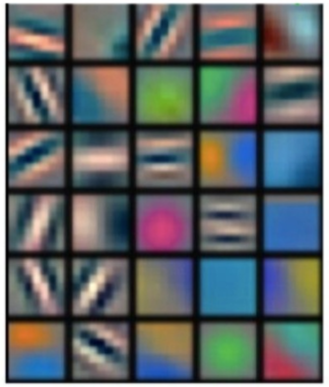
- The features that would be extracted will be highly local. This helps in capturing smaller, fine grained features in the image.
- Using kernels sequentially (i.e increasing number of layers) allows the network to learn a hierarchical feature representation

Building a CNN

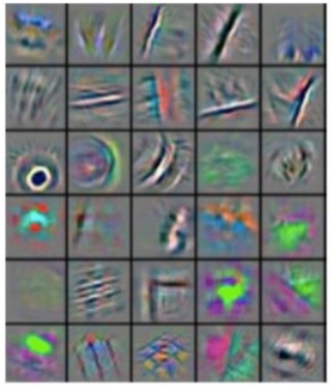


4 stacked 3x3 convs get the same receptive field as a 9x9 conv, but use fewer parameters

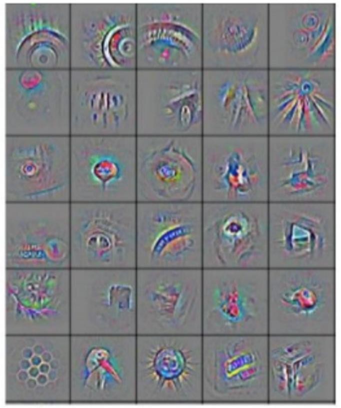
low-level features



mid-level features



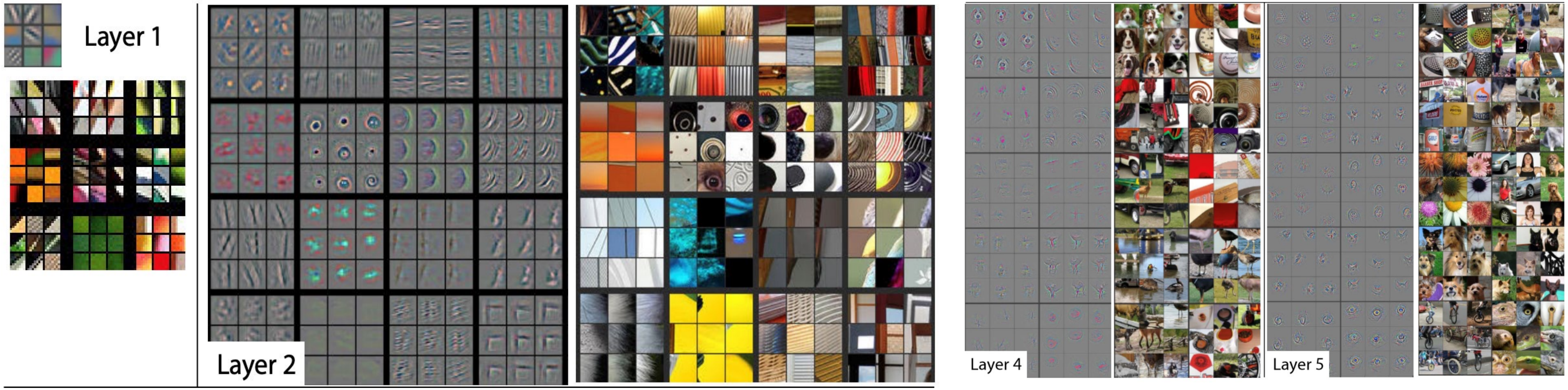
high-level features



Hierarchical Feature Learning

Building a CNN

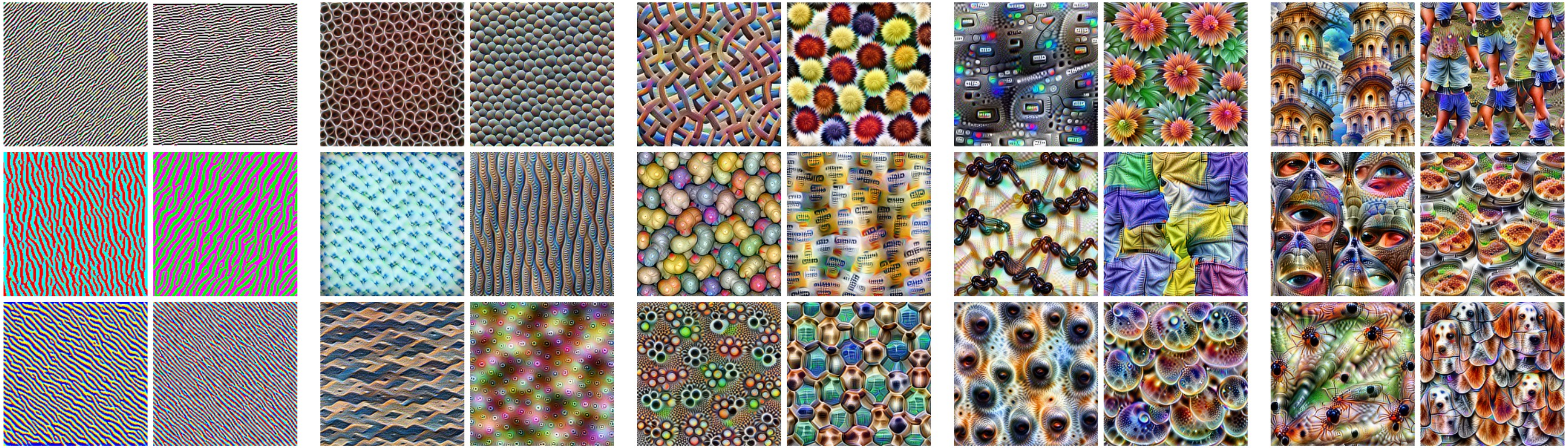
Hierarchical Feature Learning



Visualizing and Understanding Convolutional Networks
arXiv:1311.2901v3

Building a CNN

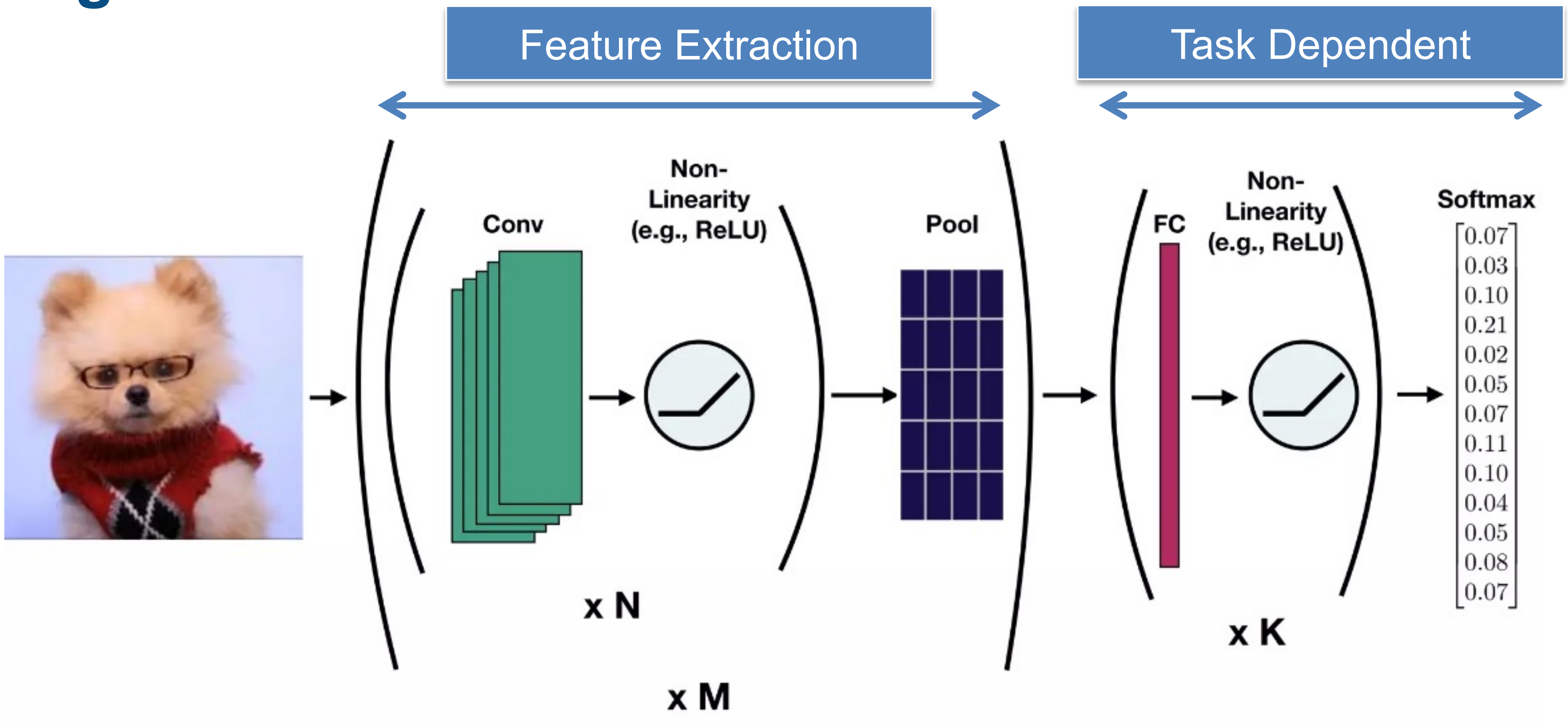
Hierarchical Feature Learning



Edges (layer conv2d0) **Textures** (layer mixed3a) **Patterns** (layer mixed4a) **Parts** (layers mixed4b & mixed4c) **Objects** (layers mixed4d & mixed4e)

Feature Visualization
https://distill.pub/2017/feature-visualization/

Building a CNN

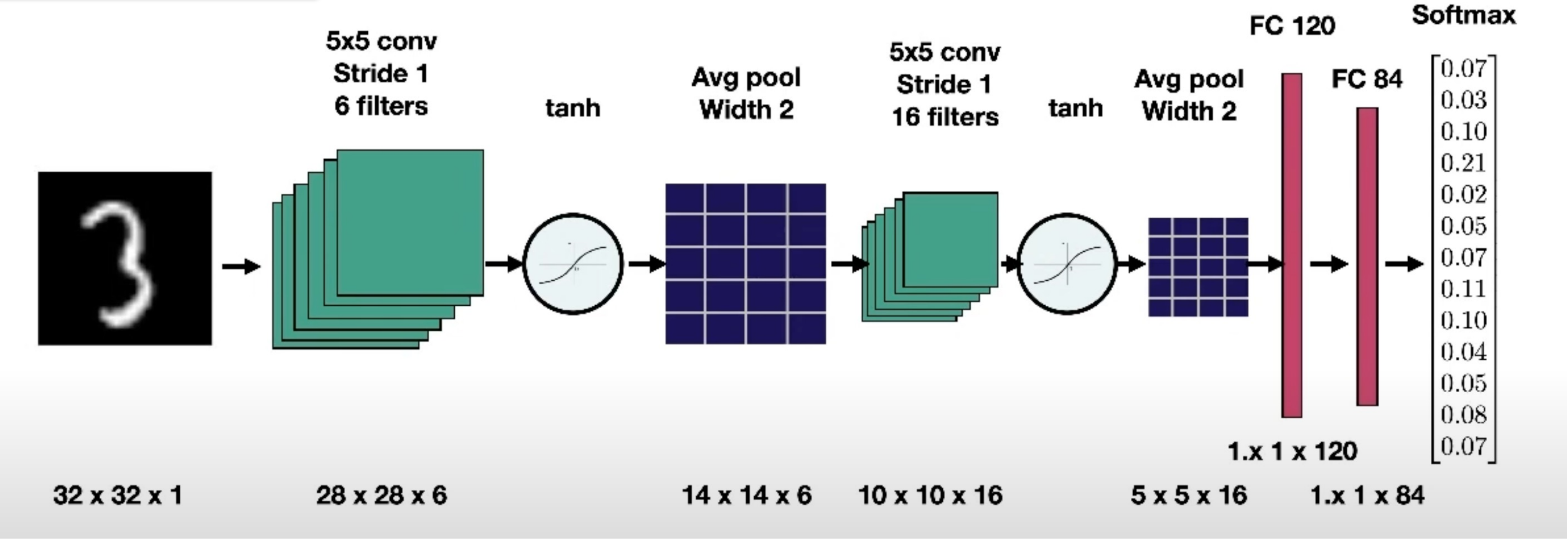


1. **Convolution**: Apply filters to generate feature maps.
2. **Non-linearity**: Often ReLU.
3. **Pooling**: Downsampling operation on each feature map.

Lecture 2A: Convolutional Neural Networks (Full Stack Deep Learning - Spring 2021)

Building a CNN

LeNet Architecture



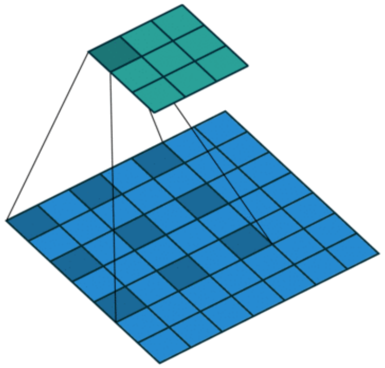
Lecture 2A: Convolutional Neural Networks
(Full Stack Deep Learning - Spring 2021)

Building a CNN

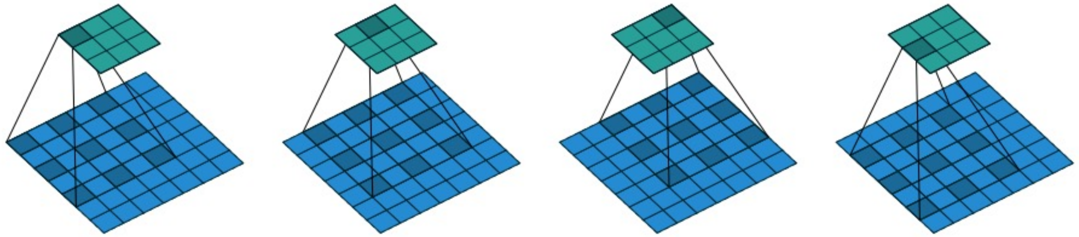
Multi-Scale context aggregation by dilated convolutions
arXiv:1511.07122v3

Wavenet: A generative model for raw audio
arXiv:1609.03499v2

Dilated Convolutions



- Dilated convolutions can “see” a greater portion of the image by skipping pixels
- The (3, 3) 1-dilated convolution illustrated here has a (5, 5) receptive field
- Stacking dilated convolutions up quickly gets to large receptive fields



```
import torch.nn as nn
m = nn.Conv2d(16, 33, 3, stride=1,padding=1,dilation=2)
m
✓ 0.0s
Conv2d(16, 33, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), dilation=(2, 2))
```

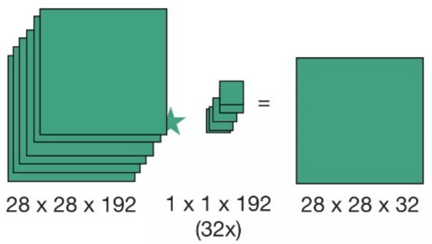
- Input: $(N, C_{in}, H_{in}, W_{in})$ or (C_{in}, H_{in}, W_{in})
- Output: $(N, C_{out}, H_{out}, W_{out})$ or $(C_{out}, H_{out}, W_{out})$, where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

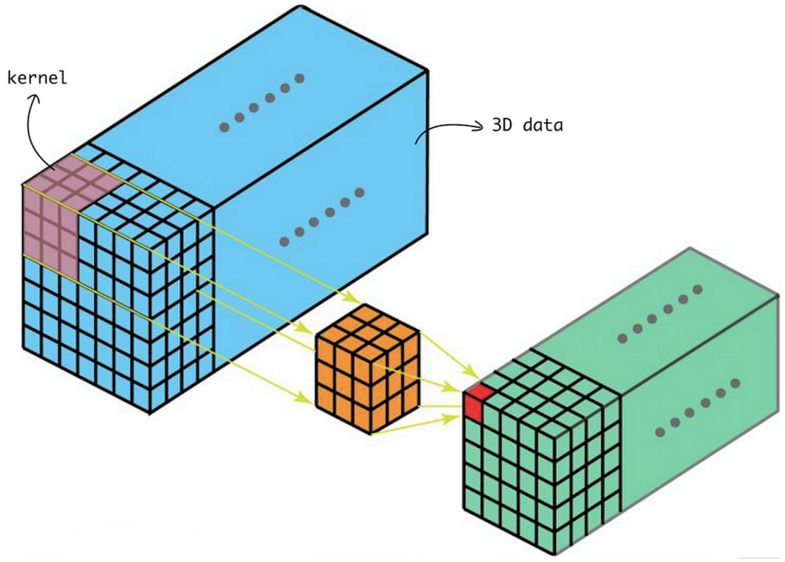
$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

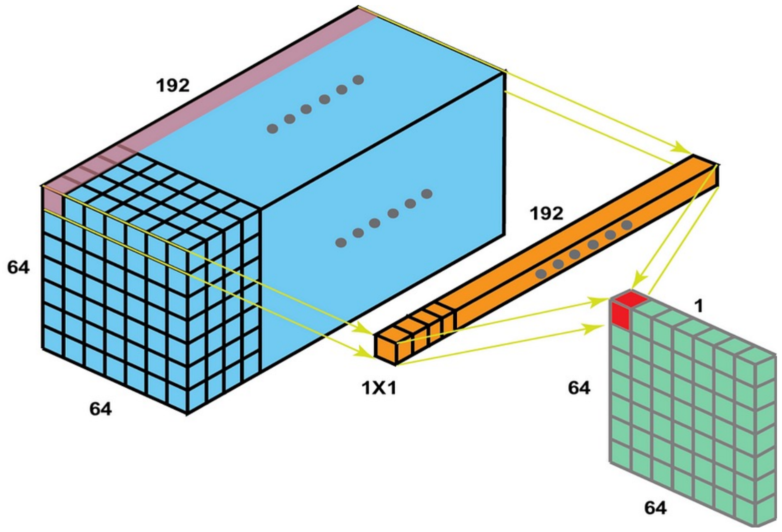
Building a CNN

1x1 Convolution



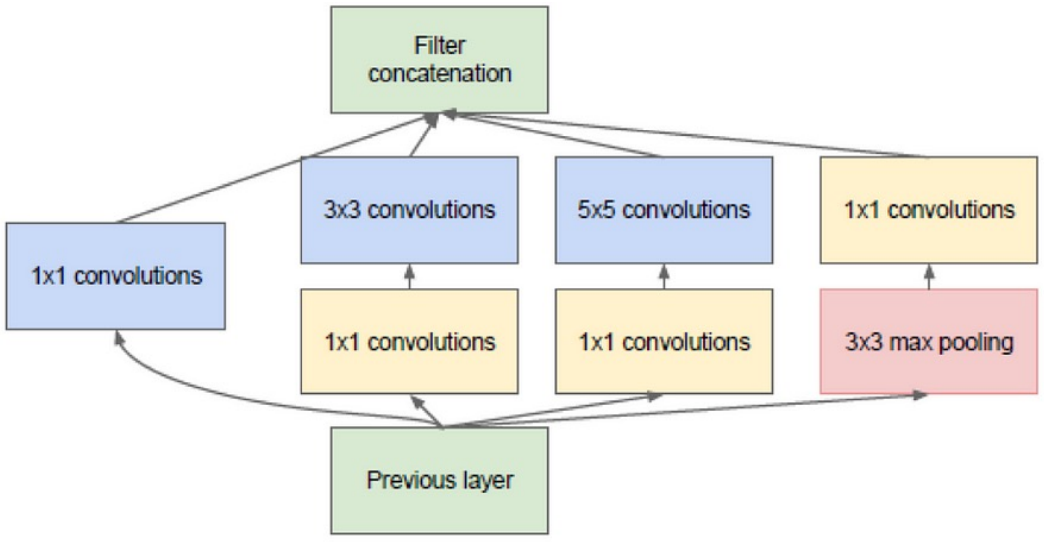
- A way to reduce the "depth" dimension of convolutional outputs
- Corresponds to applying an MLP to every pixel in the convolutional output
- Crucial to popular convnet architectures like Inception (GoogleNet)



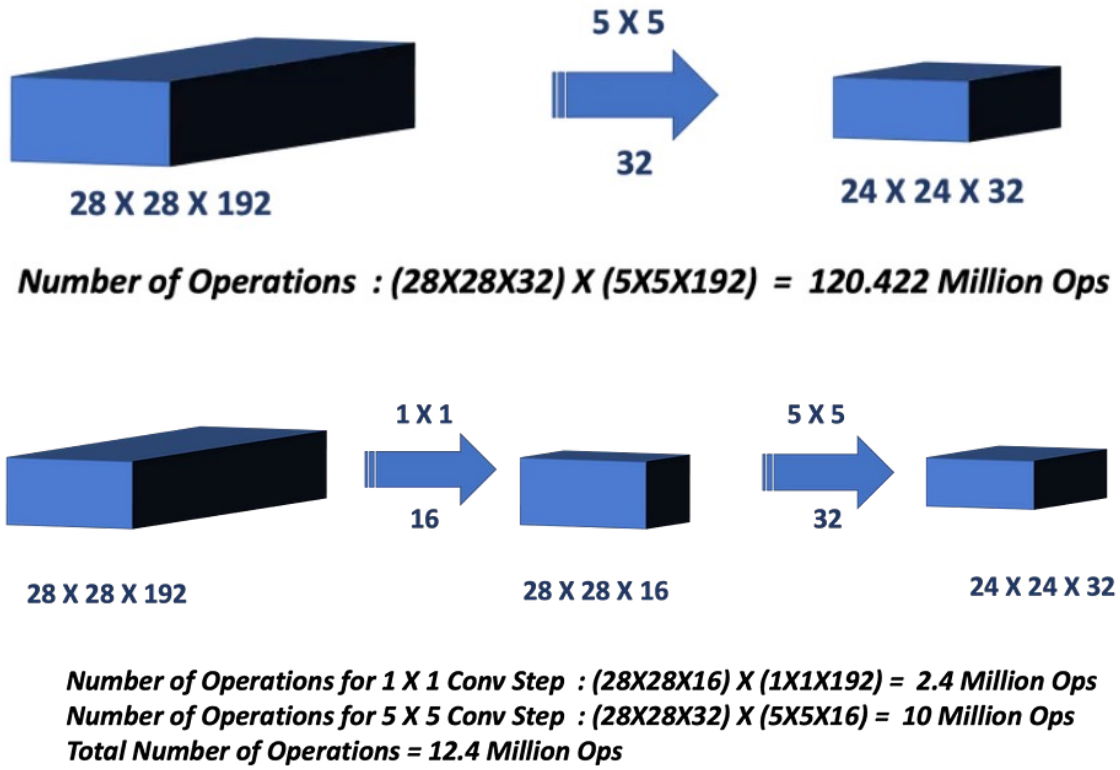


Building a CNN

1x1 Convolution



(b) Inception module with dimensionality reduction

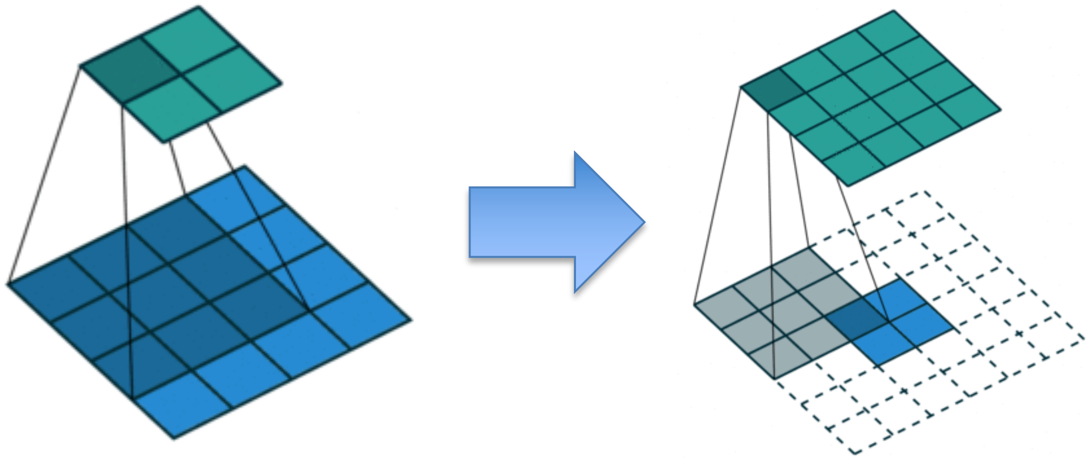


Going deeper with convolutions
arXiv:1409.4842v1

Network In Network
arXiv:1312.4400v3

Building a CNN

Transposed Convolution



- A transposed convolutional layer aims to **reconstruct** the spatial dimensions of the convolutional layer and reverses the downsampling techniques applied to it.

- In contrast to the regular convolution that reduces input elements via the kernel, the transposed convolution broadcasts input elements via the kernel, thereby producing an output that is larger than the input

Note about Building CNNs – Pytorch and TensorFlow

PyTorch

CONV2D

CLASS torch.nn.Conv2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*, *bias=True*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 2D convolution over an input signal composed of several input planes.

In the simplest case, the output value of the layer with input size (N, C_{in}, H, W) and output $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$ can be precisely described as:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) \star \text{input}(N_i, k)$$

- NCHW (Number of Samples, Channels, Height, Width) - channels precede height and width dimension

- **NHWC** (Number of Samples, Height, Width, Channels) - height and width dimensions comes first.

TensorFlow

```
tf.nn.conv2d(
    input,
    filters,
    strides,
    padding,
    data_format='NHWC',
    dilations=None,
    name=None
)
```

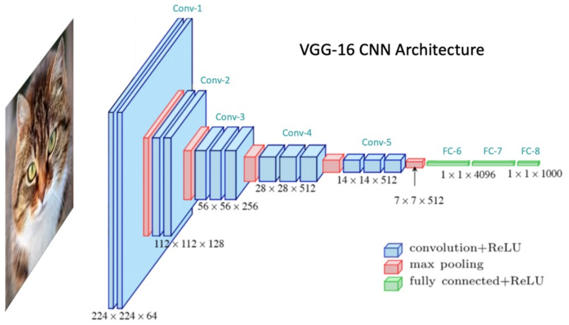
The `input` tensor may have rank `4` or higher, where shape dimensions `[:-3]` are considered batch dimensions (`batch_shape`).

Given an input tensor of shape `batch_shape + [in_height, in_width, in_channels]` and a filter / kernel tensor of shape `[filter_height, filter_width, in_channels, out_channels]`, this op performs the following:

1. Flattens the filter to a 2-D matrix with shape `[filter_height * filter_width * in_channels, output_channels]`.
2. Extracts image patches from the input tensor to form a *virtual* tensor of shape `[batch, out_height, out_width, filter_height * filter_width * in_channels]`.
3. For each patch, right-multiplies the filter matrix and the image patch vector.

In detail, with the default NHWC format,

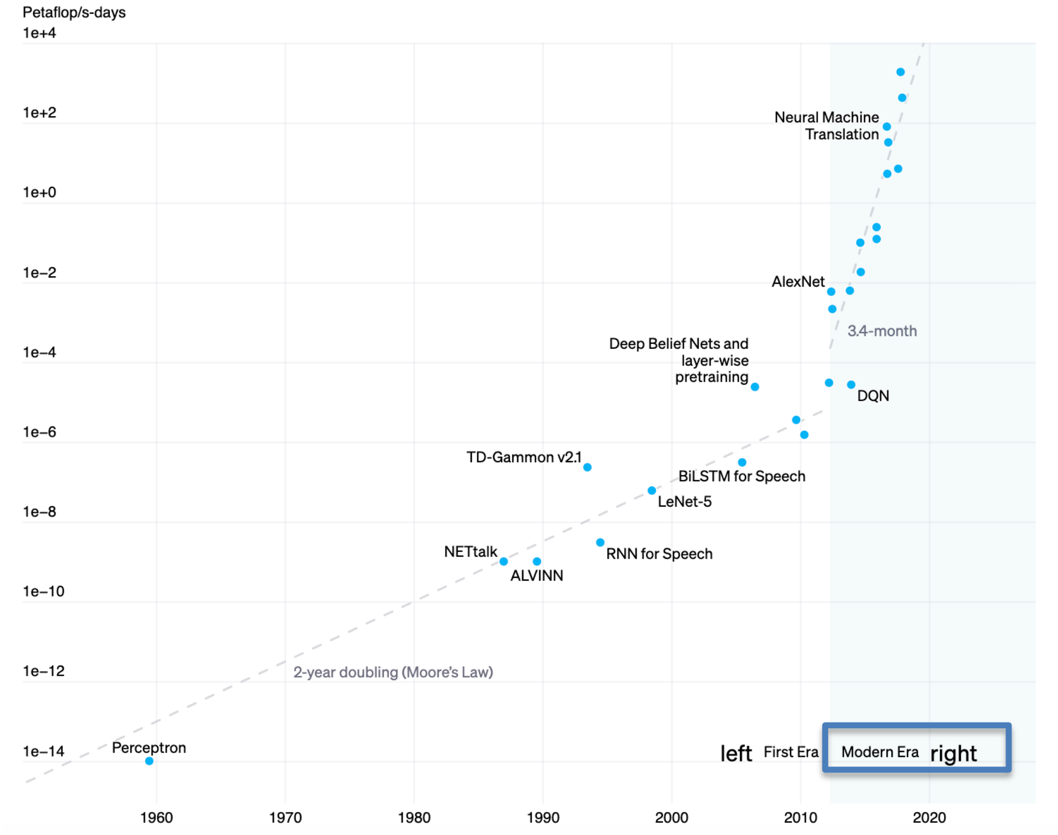
Key concepts & Inductive Biases



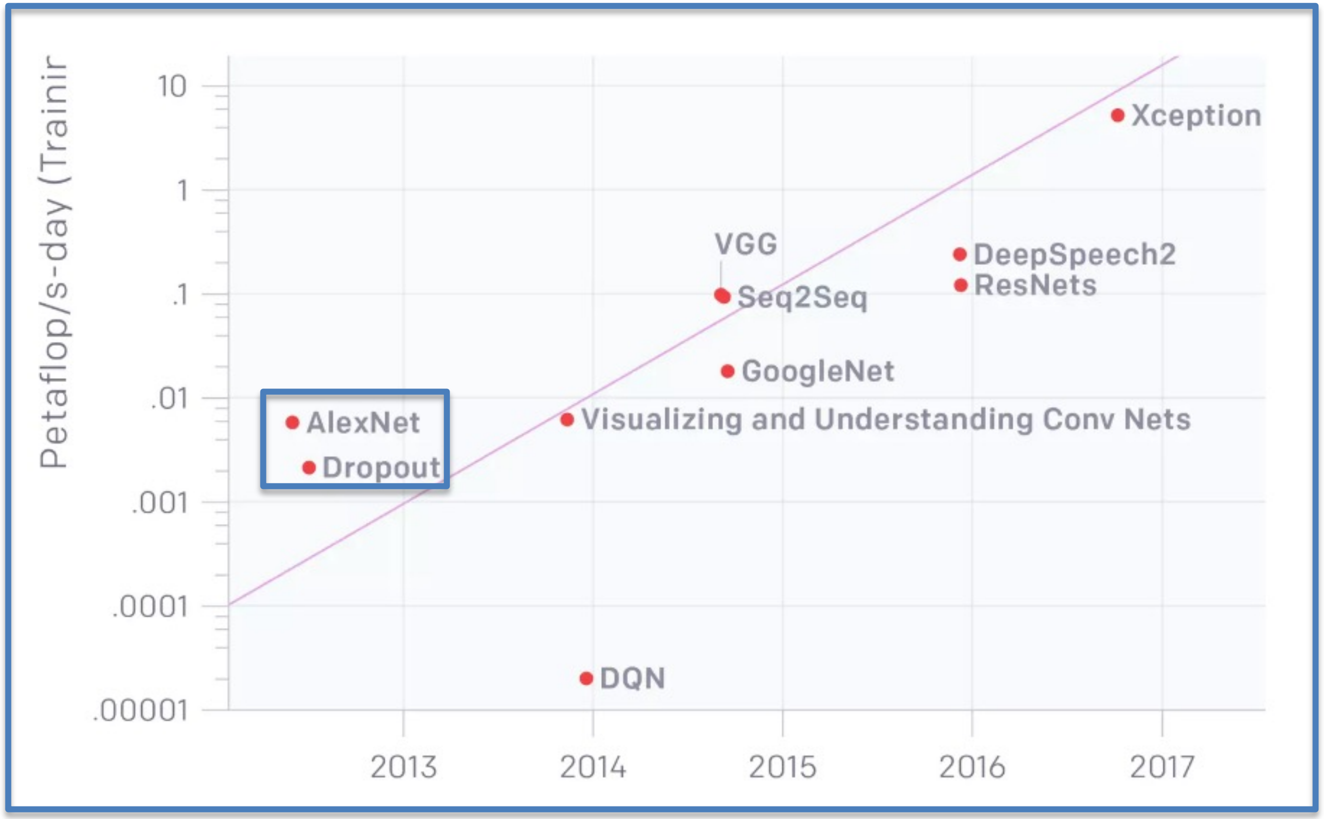
- Sparse connectivity
- Locality
- Inputs of variable size
- Parameter Sharing & Translation Equivariance
- Translation invariance – Pooling Layers
- Hierarchical Feature Learning

Popular Convolutional Neural Network Architectures – Quick Review

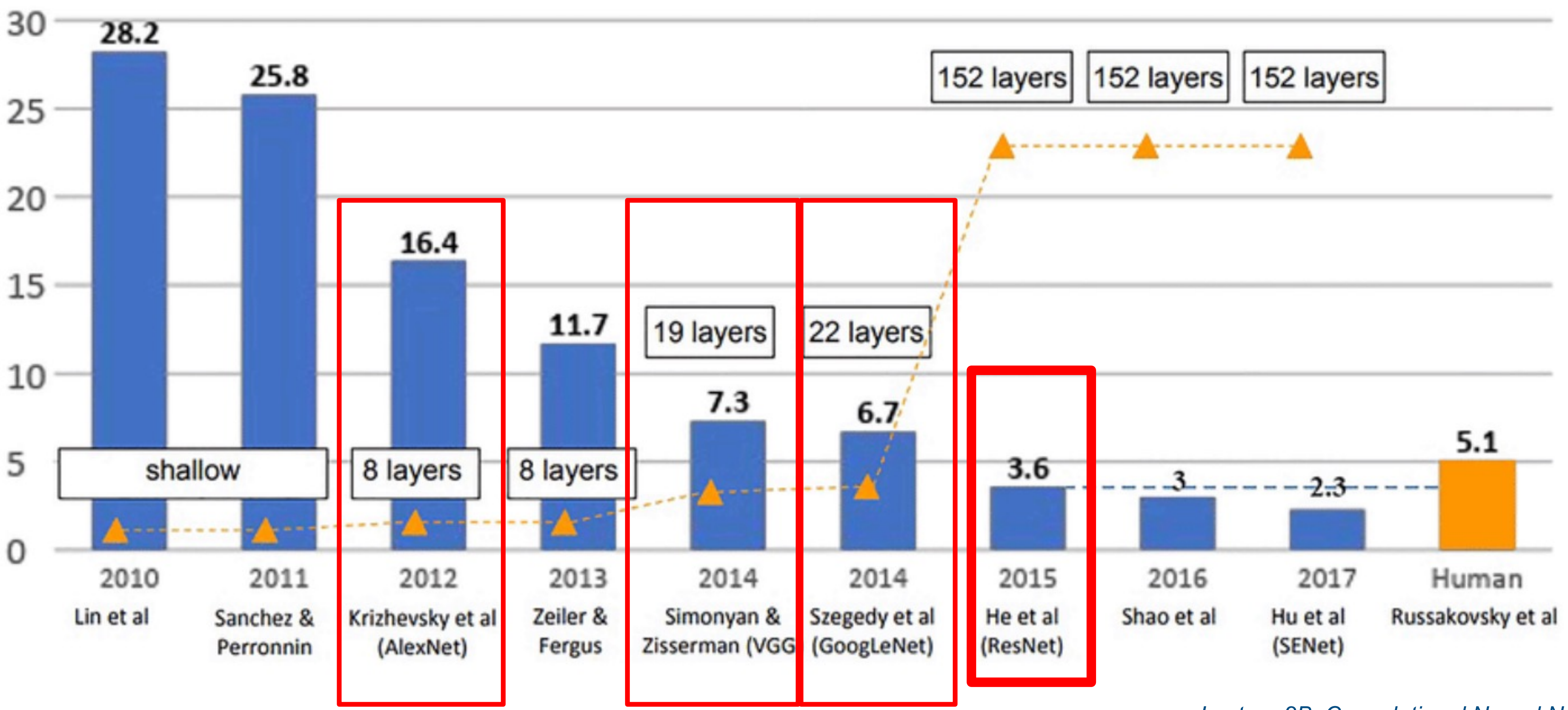
Popular CNN-based architectures



OpenAI – AI and Compute Blog



Popular CNN-based architectures



Lecture 2B: Convolutional Neural Networks (Full Stack Deep Learning - Spring 2021)

Popular CNN-based architectures - ResNets

ResNet

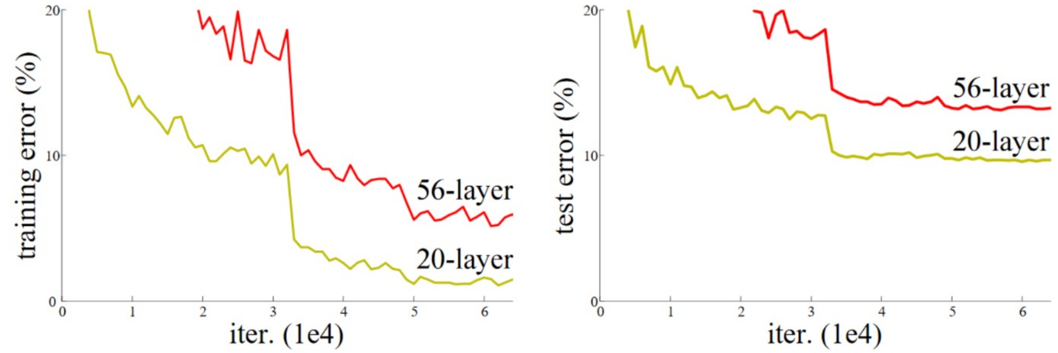
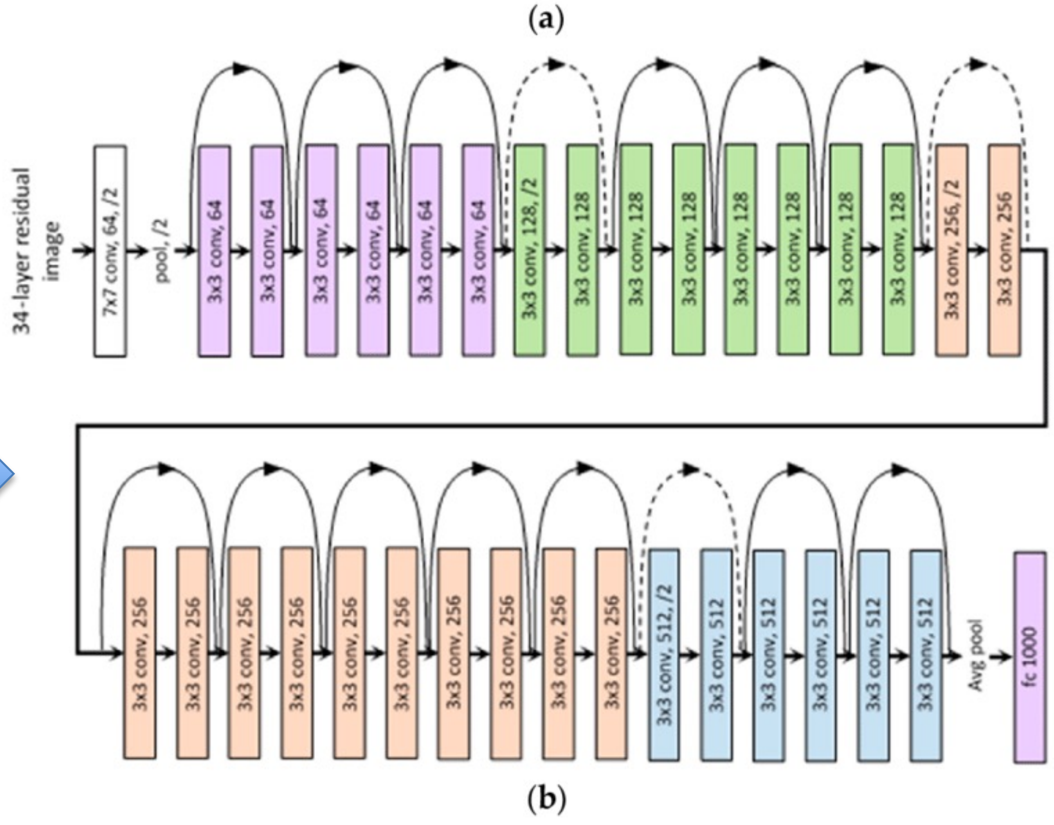


Figure 14.1: Training of networks of different depth (courtesy of Kaiming He et al.)

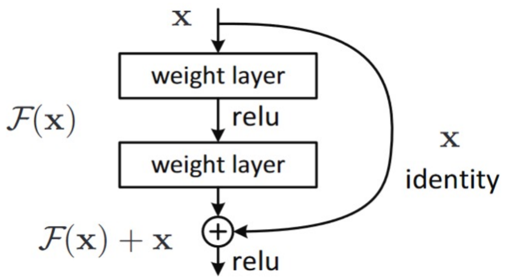
[“Deep Residual Learning for Image Recognition”]()

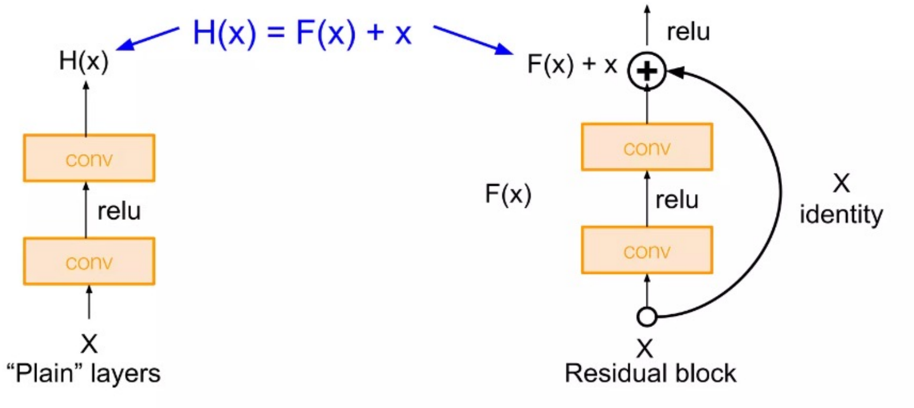


ResNet-34 Layered architecture

Popular CNN-based architectures

Residual Blocks – Skip Connections





Residual Blocks – Bottleneck Layer

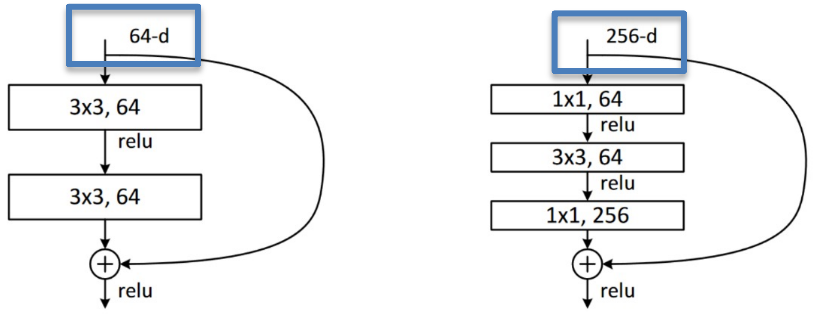
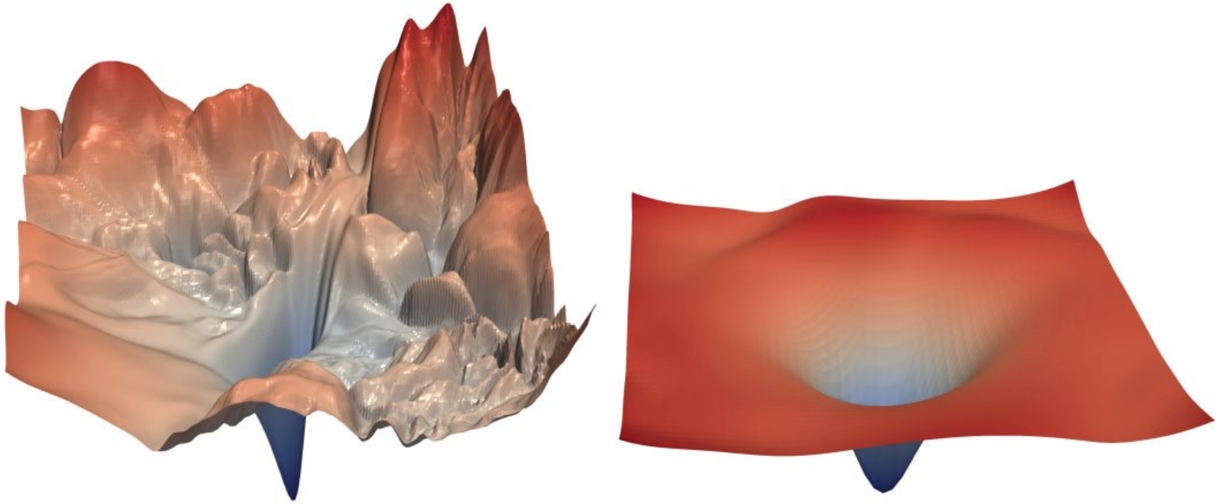


Figure 14.4: Comparison of regular and bottleneck ResNet blocks (courtesy of Kaiming He et al.)

Popular CNN-based architectures



(a) without skip connections

(b) with skip connections

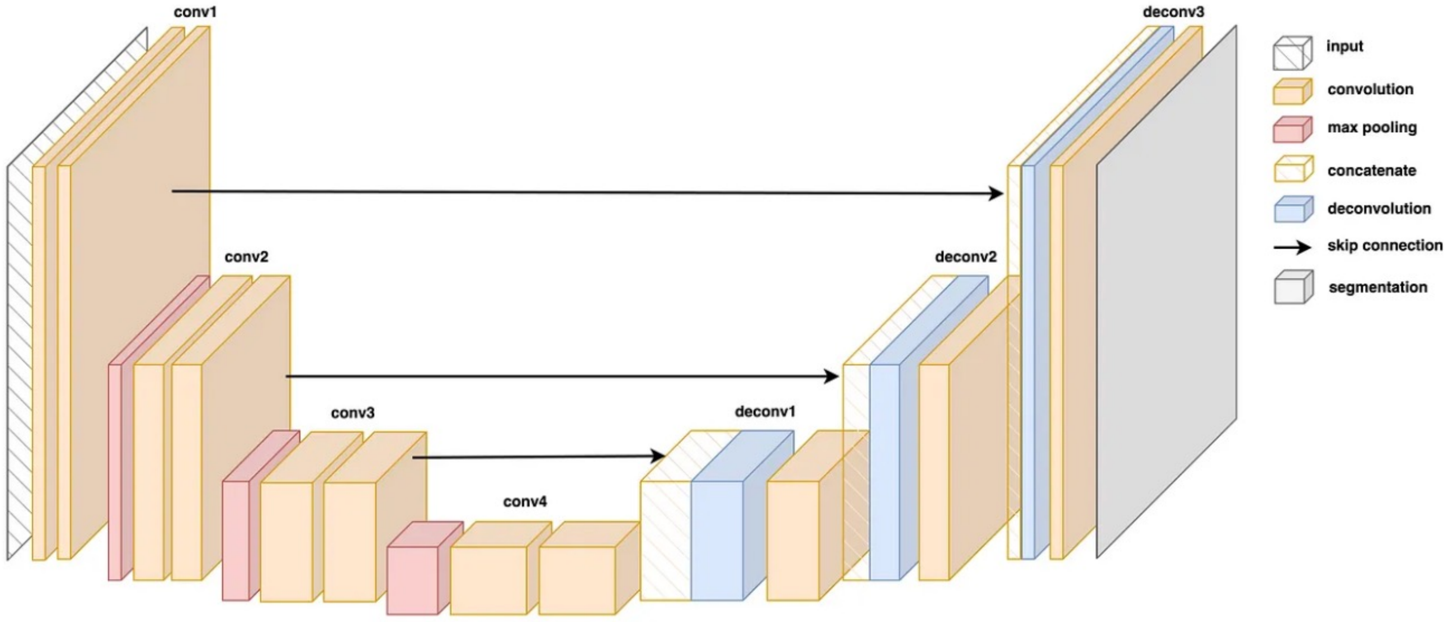
The loss surfaces of ResNet-56 with and without skip connections

Using skip connections helps smooth the loss function, which makes training easier as it avoids falling into a very sharp area.

Visualizing the Loss Landscape of Neural Nets
https://arxiv.org/abs/1712.09913

Popular CNN-based architectures – U-Nets

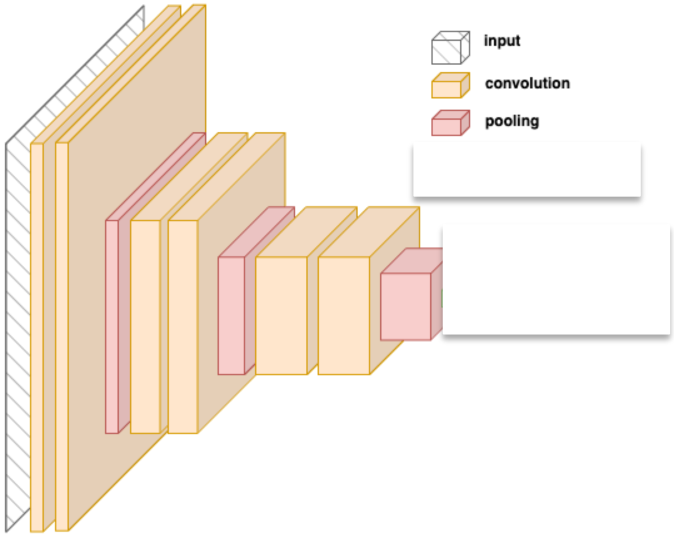
U-Net



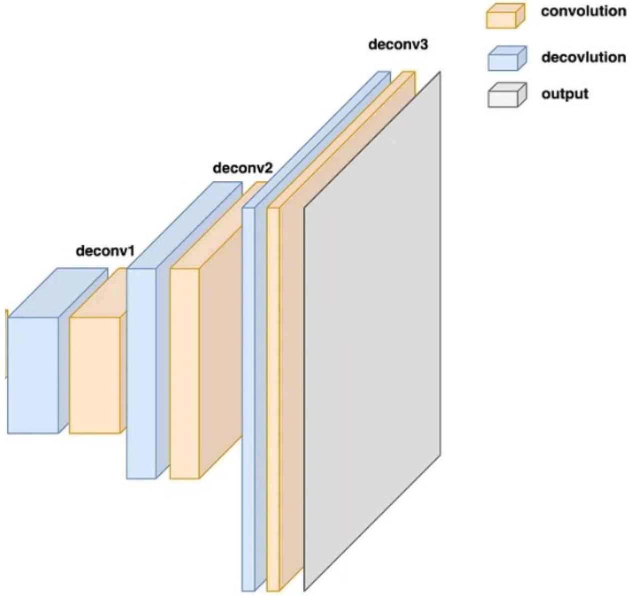
U-Net: Convolutional Networks for Biomedical Image Segmentation
arxiv.org/abs/1505.04597

Popular CNN-based architectures – U-Nets

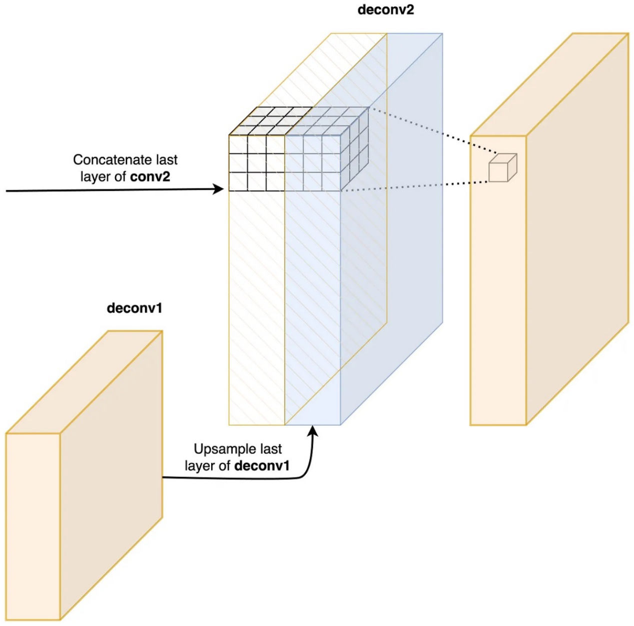
U-Net



Encoder



Decoder

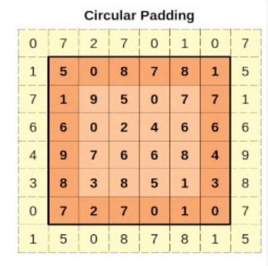


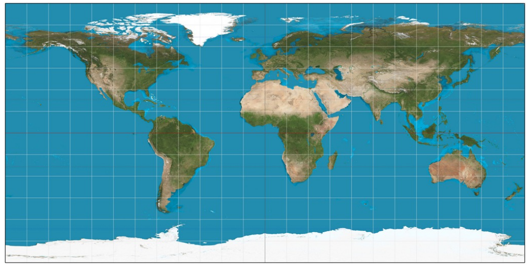
Skip Connections

https://towardsdatascience.com/u-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a

Popular CNN-based architectures – U-Nets

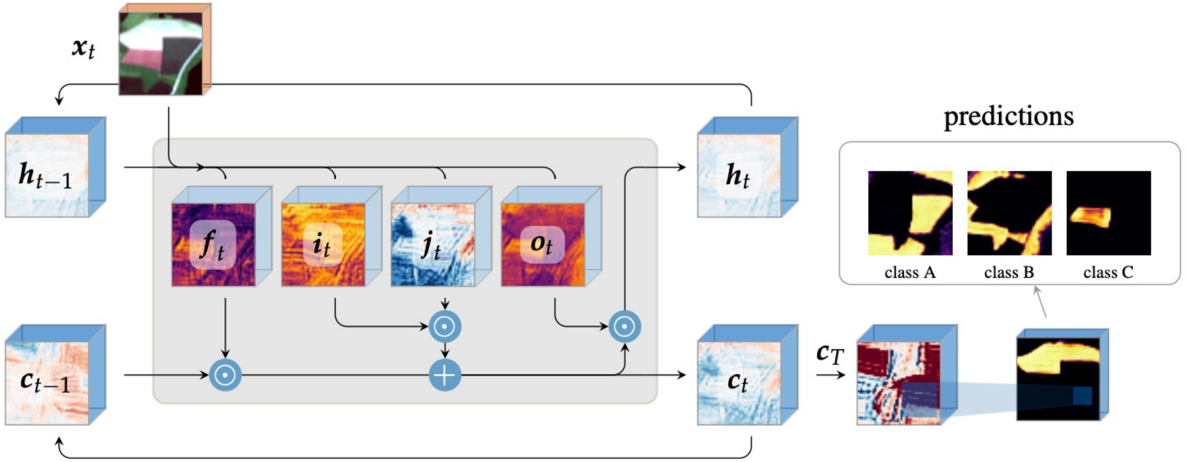
Periodic Convolutions





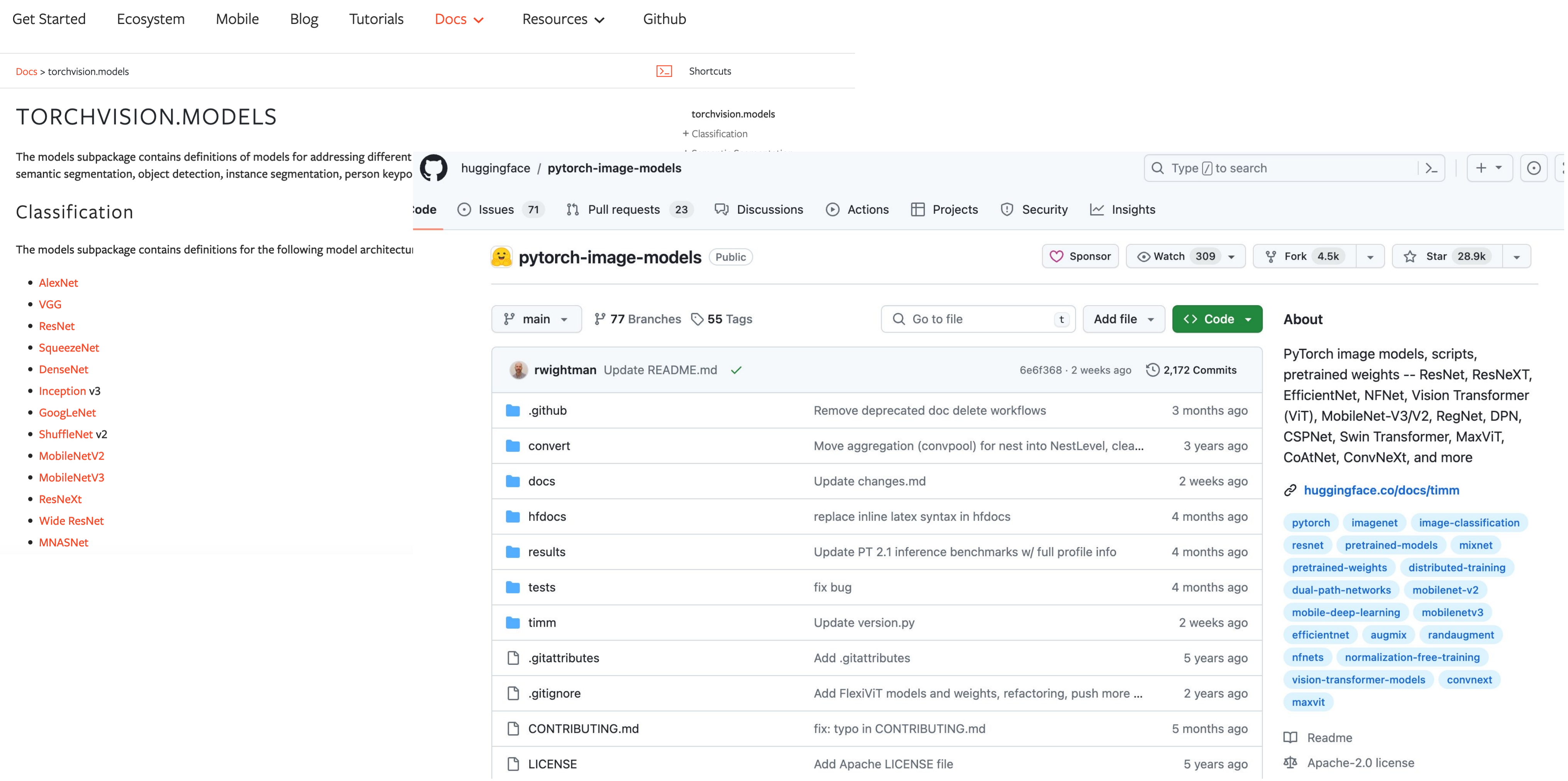
https://github.com/pangeo-data/WeatherBench/blob/master/src/train_nn.py#L102

Spatio-Temporal Data - ConvLSTMS



Convolutional LSTMs for Cloud-Robust Segmentation of Remote Sensing Imagery
Marc Rußwurm

Popular CNN-based architectures



References

- Deep Learning book CNN Chapter - https://www.deeplearningbook.org/contents/convnets.html
- Understanding Deep Learning – CNN chapter - https://udlbook.github.io/udlbook/
- CNN feature visualization - https://distill.pub/2017/feature-visualization/
- CNN feature visualization - https://arxiv.org/pdf/1311.2901.pdf
- *Intuitively Understanding Convolutions for Deep Learning*
- *A guide to convolution arithmetic for deep Learning - Dumoulin V., Visin. F, 2018, arXiv:1603.07285*
- *CNN from different viewpoints - Matt Kleinsmith*
- *Lecture 2A and Lecture 2B Convolutional Neural Networks (Full Stack Deep Learning - Spring 2021)*
- *Invariance and equivariance -* https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/equivariance.pdf
- *Invariance and equivariance -* 05 Imperial's Deep learning course: Equivariance and Invariance - Bernhard Kainz
- Inductive bias - locality