Convolutional Neural Networks

Hao Dong

2019, Peking University

Convolutional Neural Networks

- Motivation
- Convolutional Algorithms
- Pooling Algorithms
- Hierarchical Representation Learning
- Convolutional Architectures
- Transposed Convolutional Algorithms
- Computer Vision Applications

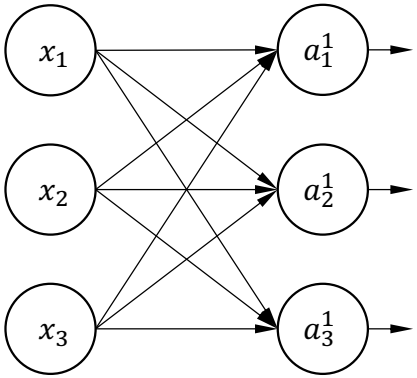
北京大學
PEKING UNIVERSITY

Motivation

Motivation

- Curse of Dimensionality

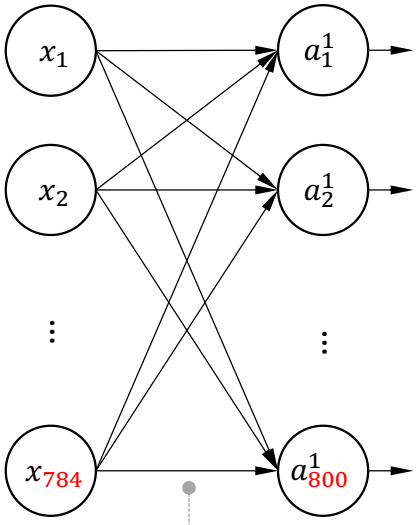
When we only have three inputs:



Fully connected layer

When we have a small image:

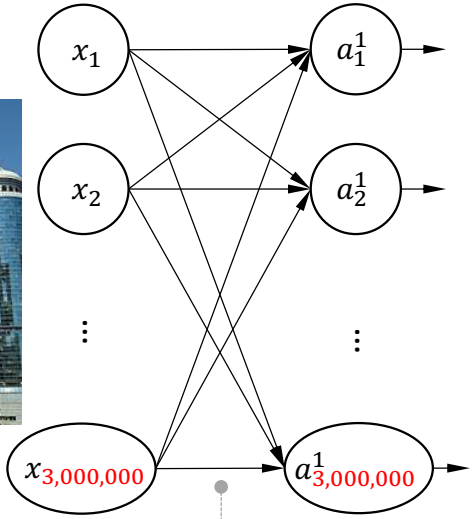
28×28×1



W is a 784×800 matrix

When we have a HD image:

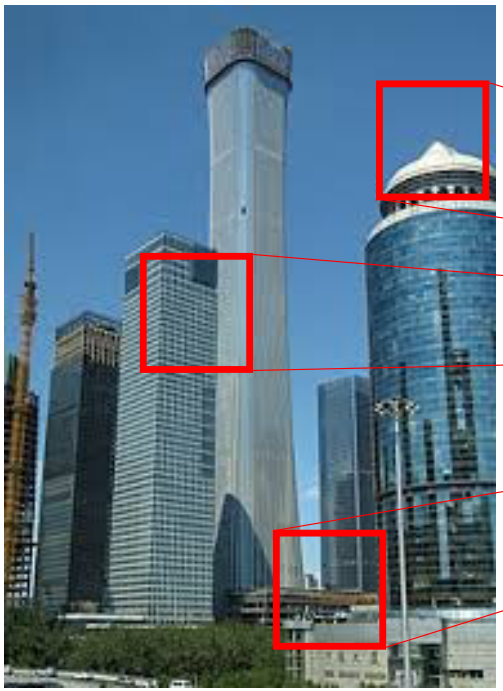
$1K \times 1K \times 3$



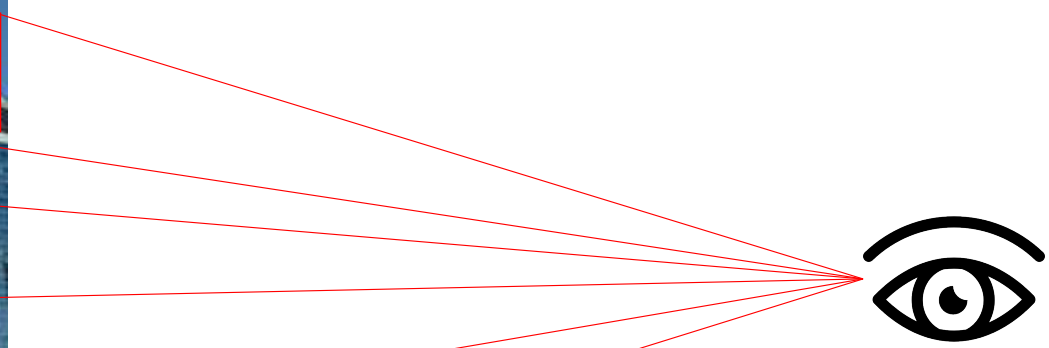
W cannot fit into the memory …

Motivation

- Eye != Fully Connected Layer



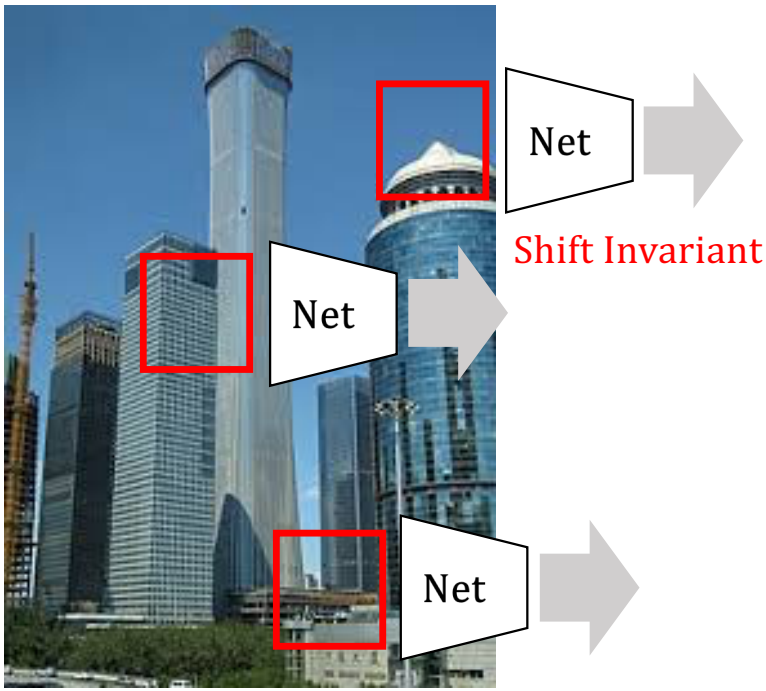
- Do human analysis each pixel using different ways? NO



Shift Invariant

Motivation

- Eye != Fully Connected Layer



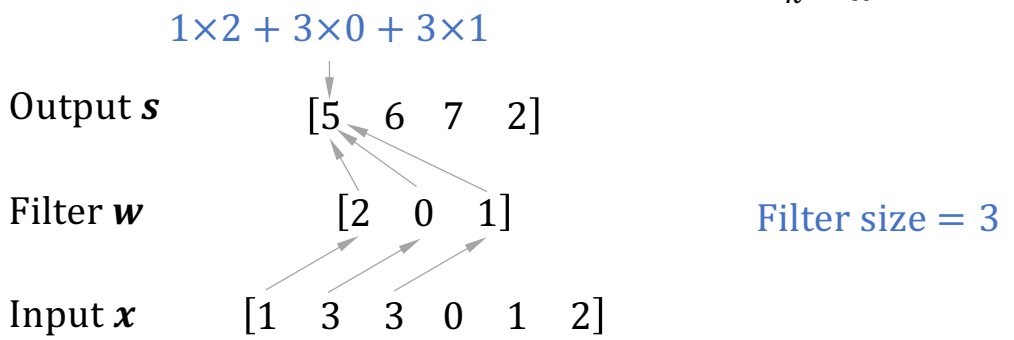
- Parameters shared over space
- Sparse connectivity
- Equivariant representation

Convolutional Algorithms

Convolutional Algorithms

- Convolution on 1D vector

1D discrete convolution: $s_t = (x * w)_t = \sum_{k=-\infty}^{\infty} x_k w_{t-k}$



Filter size = 3

- The more similar the filter (kernel) and local path are, the higher value on the output

Convolutional Algorithms

- Edge detection on greyscale 2D images

2D discrete convolution: $\boldsymbol{s}_{r,c} = \big((\boldsymbol{x} * \boldsymbol{W})\big)_{r,c} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \boldsymbol{x}_{r,c}\, \boldsymbol{w}_{r-i,c-j}$

h×w×1 image

$$\boldsymbol{W} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$

Detect vertical edges

h×w×1 image

Convolutional Algorithms

- Edge detection on RGB Images

R $\quad \boldsymbol{W}_{:,:1} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$

G $\quad \boldsymbol{W}_{:,:2} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$

B $\quad \boldsymbol{W}_{:,:3} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$

Each input channel needs one filter

$\boldsymbol{W}$ is a 3×3×3 tensor

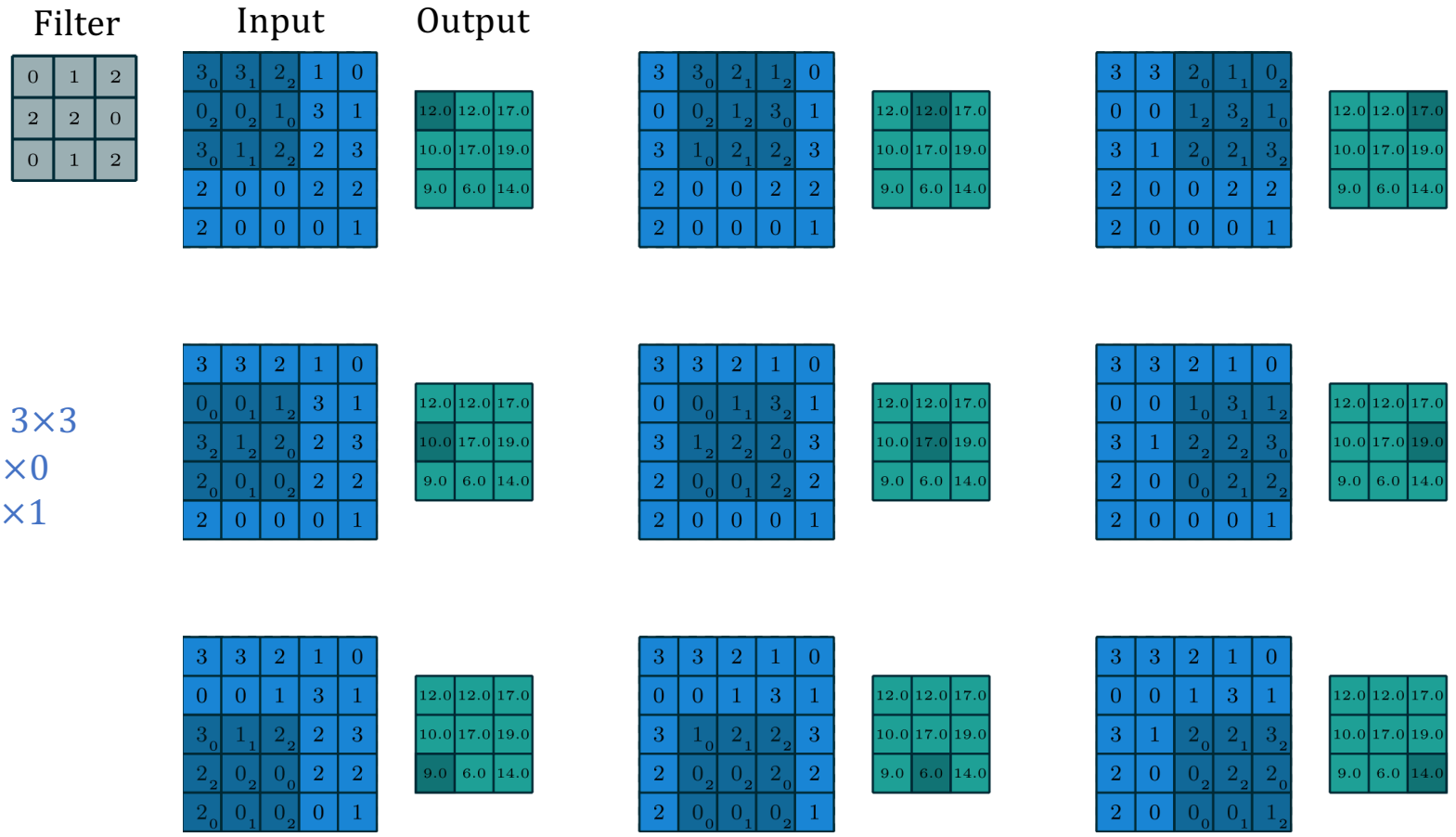
Height of filter size

Width of filter size

Depth of filter size == Number of input channels

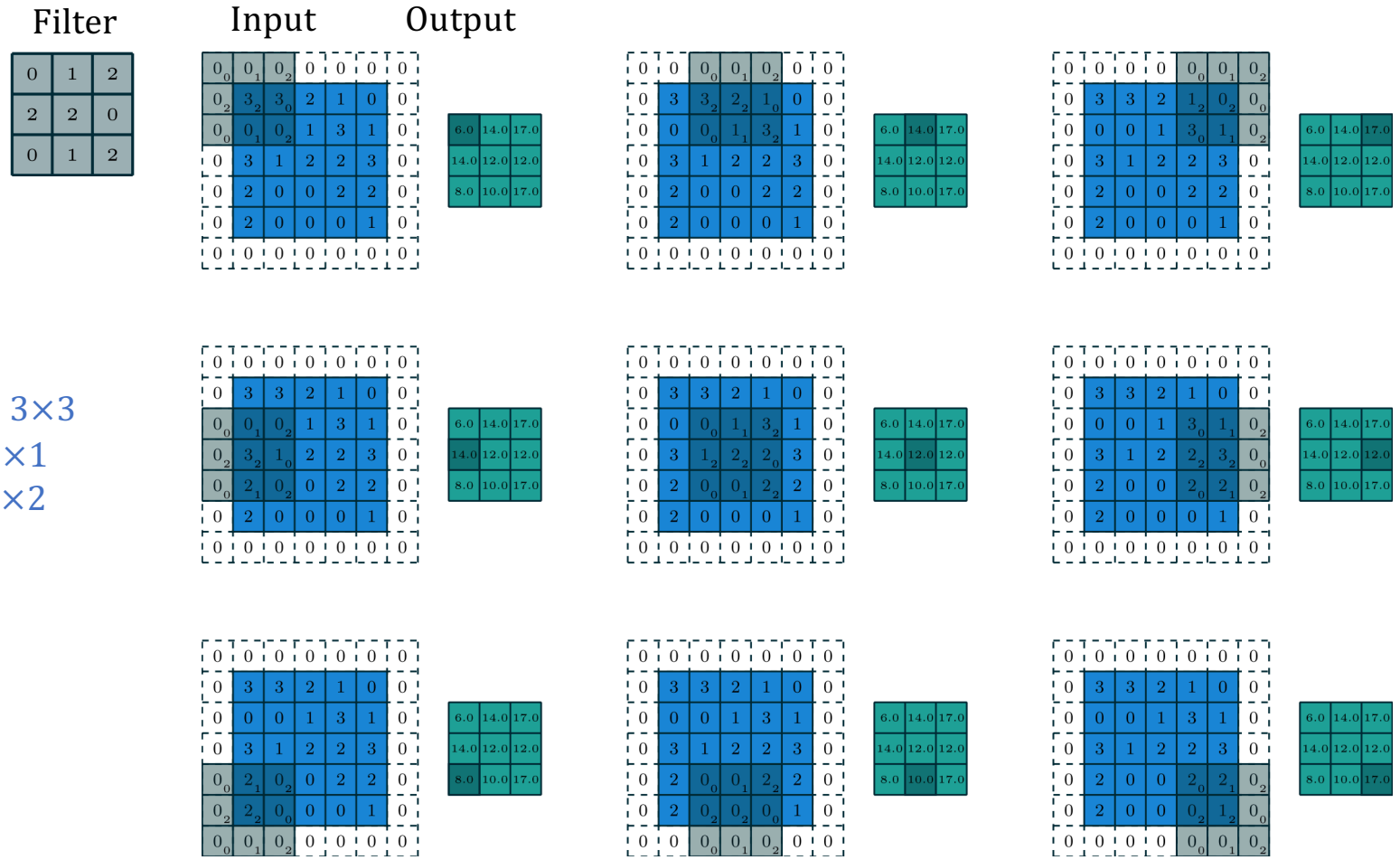
Convolutional Algorithms

- No padding, unit strides



Convolutional Algorithms

- Zero padding, non-unit strides

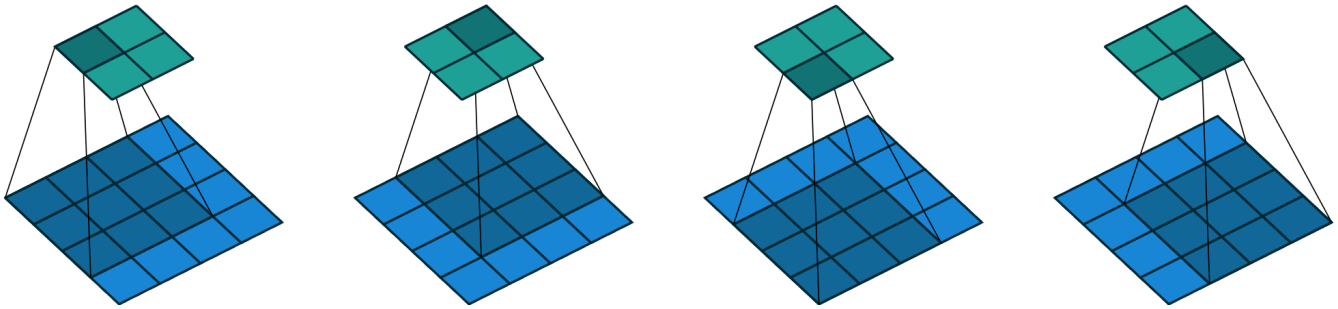


Convolutional Algorithms

- Side Views

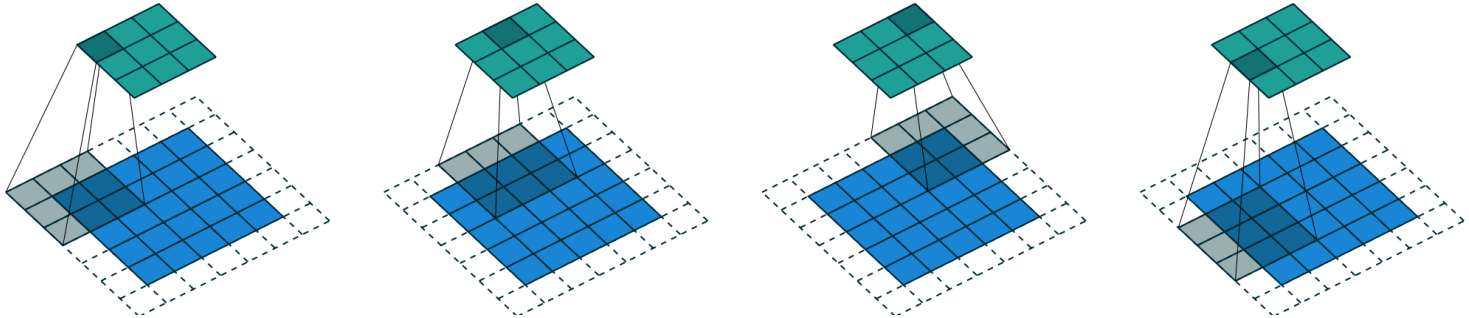
No padding and no strides

Filter size = 3×3
padding = 0×0
Strides = 1×1



Padding and strides

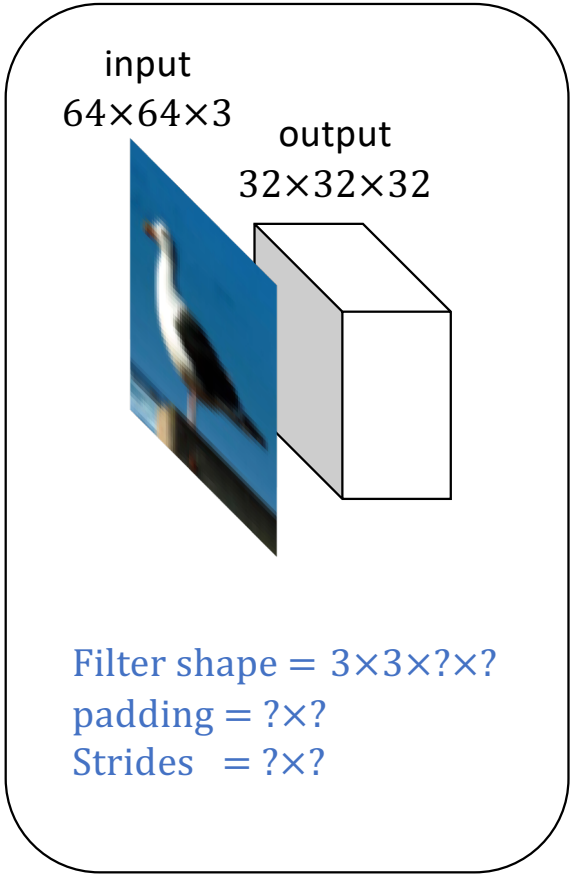
Filter size = 3×3
padding = 1×1
Strides = 2×2



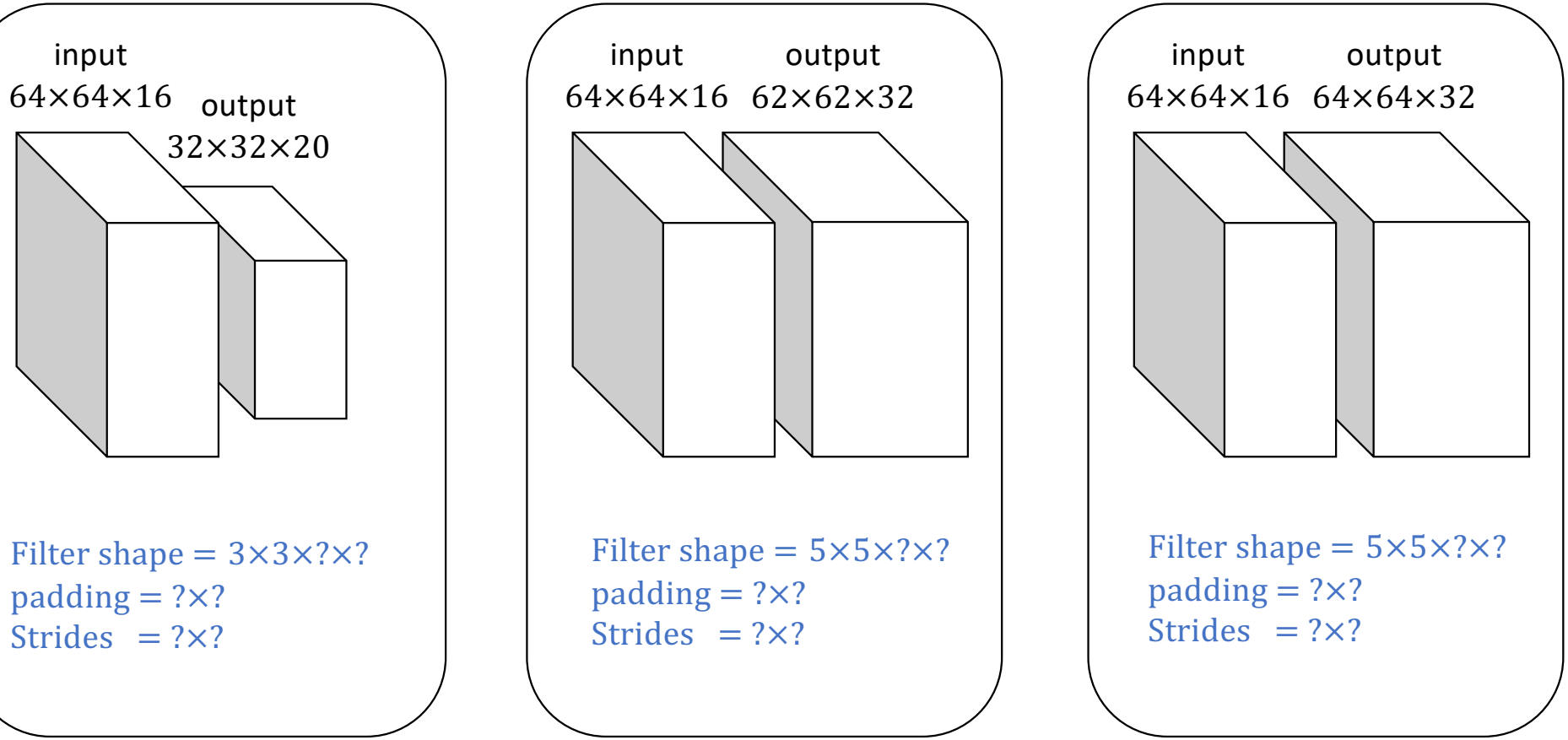
Convolutional Algorithms

- Filter (kernel) shapes
  - To convolute RGB images, a filter have 3 channels
  - The filter shape is
    (filter_height × filter_width × input_channels × n_filters)
- Number of filters (channels)
  - One filter can have one corresponding output feature map
  - The number of output channels == The number of filters
  - The output shape is (feature_height × feature_width × n_filters)

Rely on input size, padding, strides, and filter size



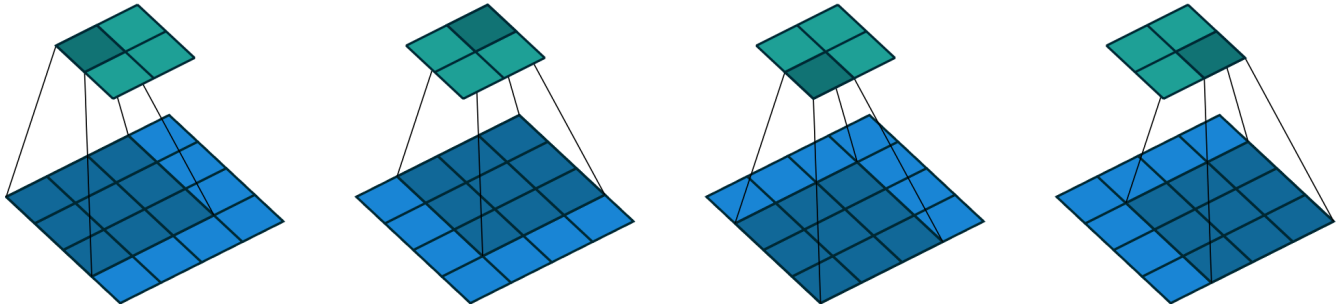
Convolutional Algorithms



Convolutional Algorithms

- Receptive field

Receptive field $= 3\times3$



Convolutional Algorithms

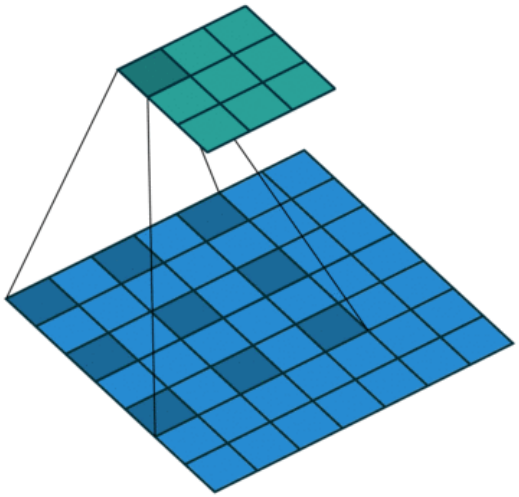
- Receptive field

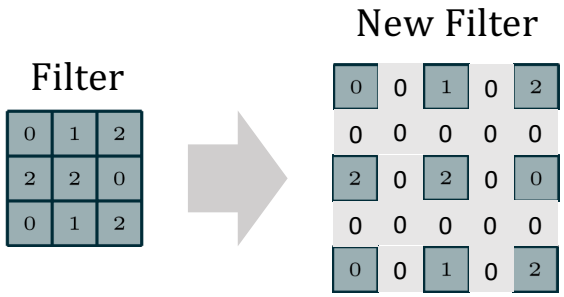
Layer #	Kernel Size	Stride	Dilation	Padding	Input Size	Output Size	Receptive Field
1	3	1	1	2	256	256	3
2	3	2	1	1	256	128	5
3	3	1	1	2	128	128	9
4	3	1	1	2	128	128	13
5	3	2	1	1	128	64	17
6	3	1	1	2	64	64	25
7	3	1	1	2	64	64	33
8	3	2	1	1	64	32	41

https://fomoro.com/research/article/receptive-field-calculator

Convolutional Algorithms

- Dilated convolution: Larger Receptive Filed





Dilation rate = 2×2

Receptive field : 3×3 → 5×5

Convolutional Algorithms

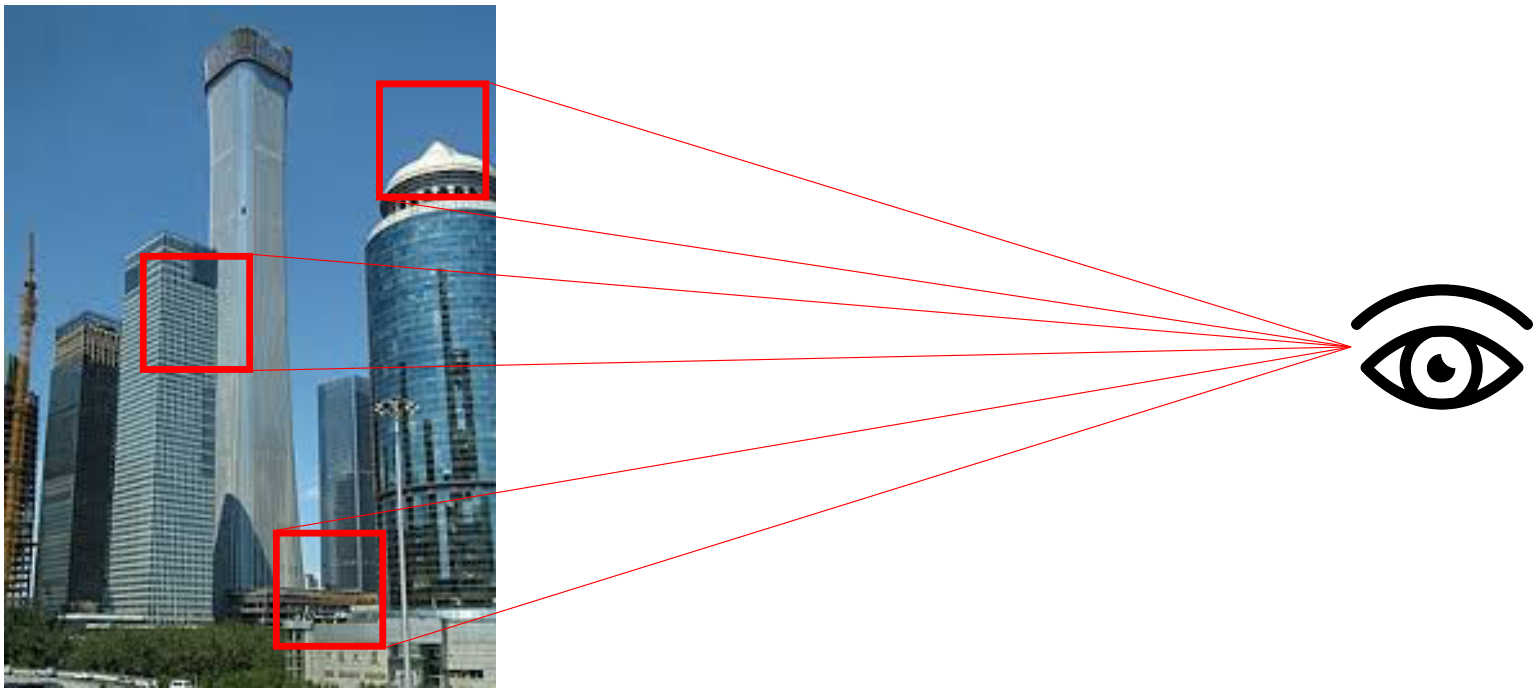
- Convolution on 3D volume
  - Filter (kernel) shapes
    - To convolute MRI images, a filter have 1 channels
    - The filter shape is (filter_depth × filter_height × filter_width × input_channels × n_filters)

  - Number of filters (channels)
    - One filter can have one corresponding output feature volume
    - The number of output channels == The number of filters
    - The output shape is (feature_depth × feature_height × feature_width × n_filters)

Pooling Algorithms

Pooling Algorithms

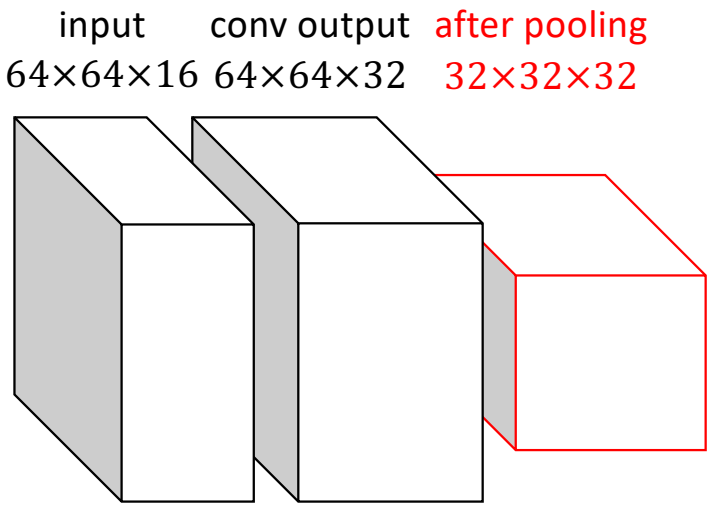
- Motivation: Shift Invariant



- Small shifts should not effect the result

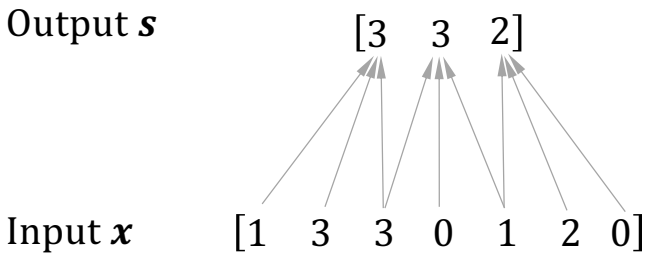
Pooling Algorithms

- Motivation: Features aggregation



Pooling Algorithms

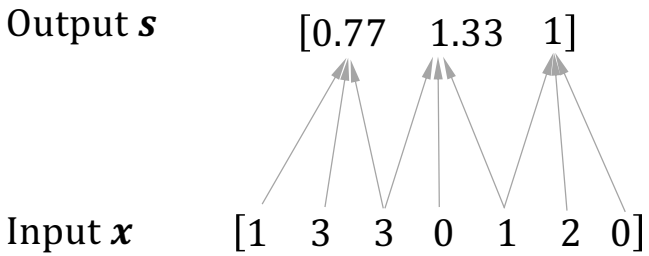
- Pooling on 1D vector
  - Max Pooling



Filter size = 3
Padding = 0
Stride = 2

  - Mean (Average) Pooling

Receptive field is increased

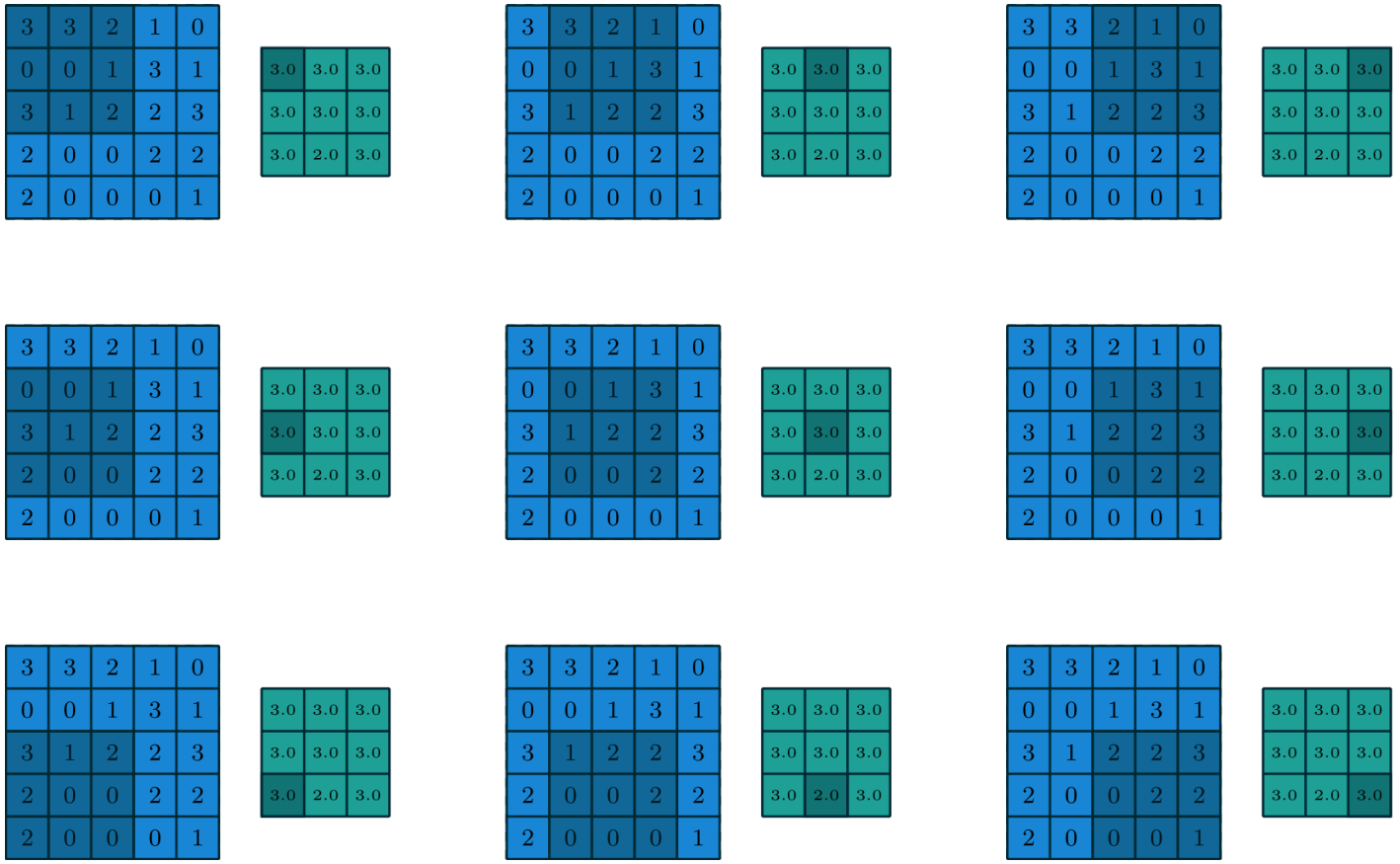


Filter size = 3
Padding = 0
Stride = 2

Pooling Algorithms

- Max Pooling on 2D matrix

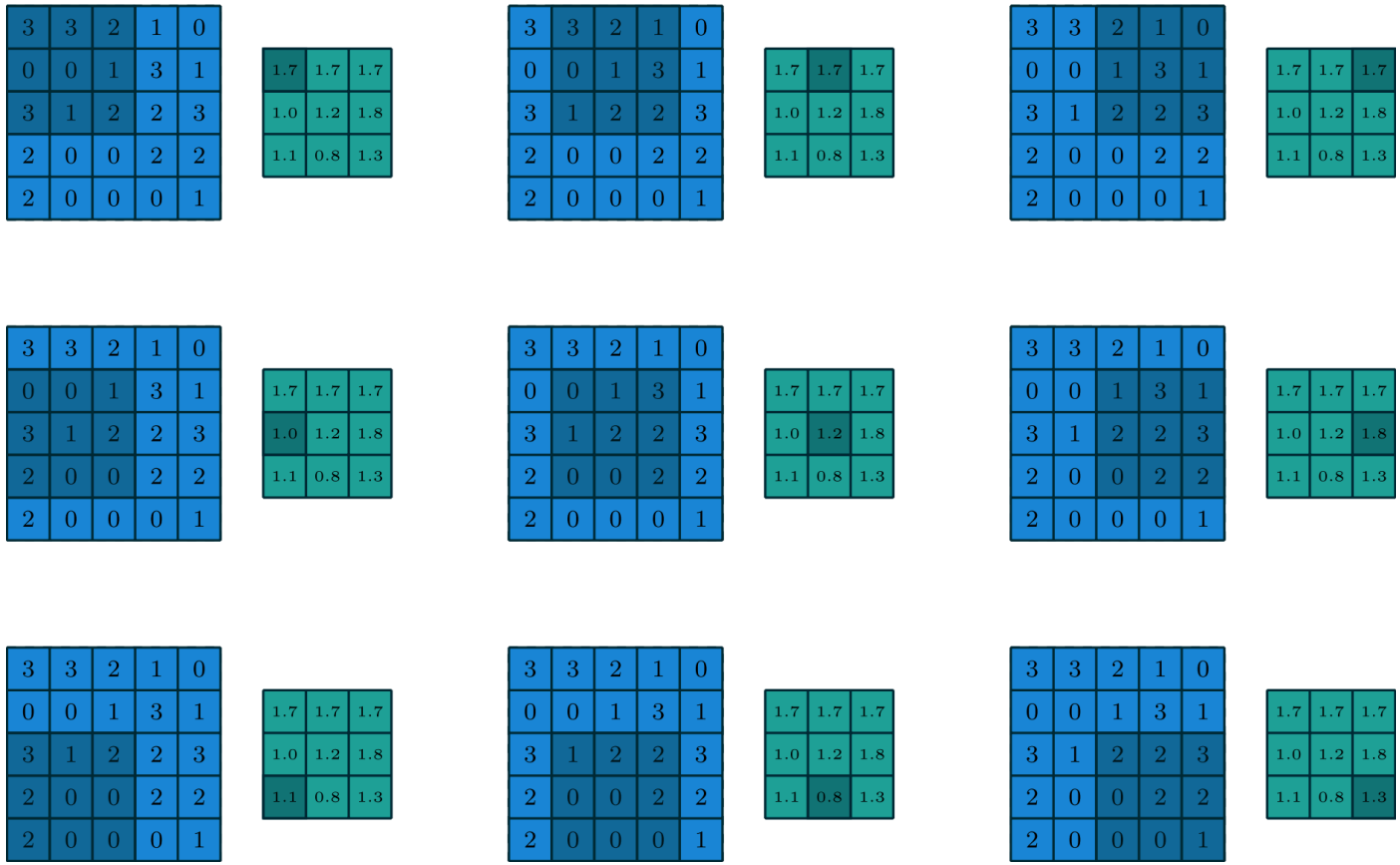
Filter size = 3×3
padding = 0×0
Strides = 1×1



Pooling Algorithms

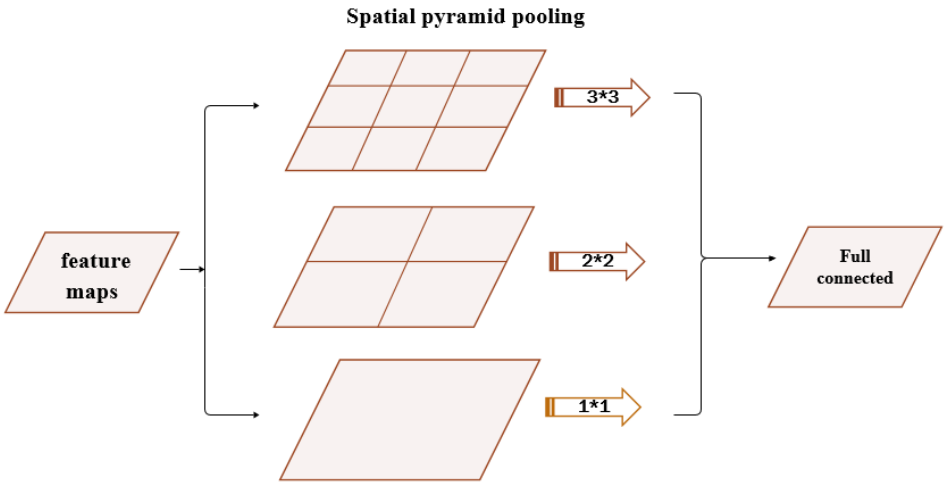
- Mean Pooling on 2D matrix

Filter size = 3×3
padding = 0×0
Strides = 1×1



Pooling Algorithms

- Spatial Pyramid Pooling



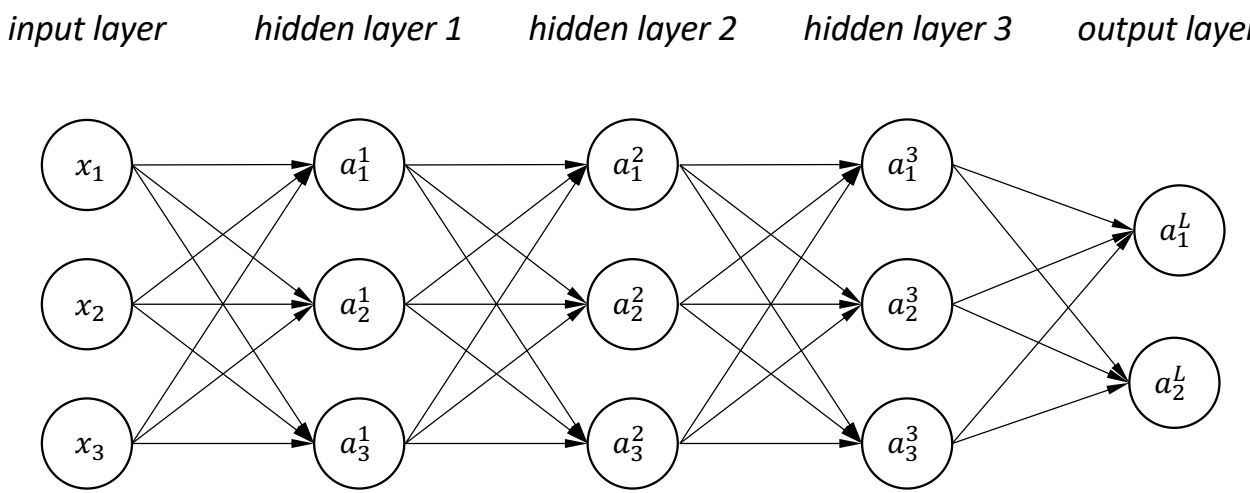
Pooling Algorithms

- Pooling on 3D volume
  - ….

Hierarchical Representation Learning

Hierarchical Representation Learning

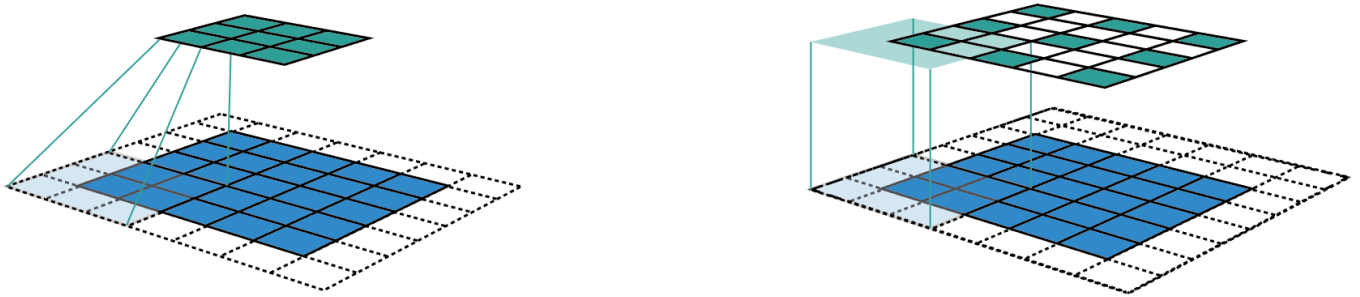
- Motivation



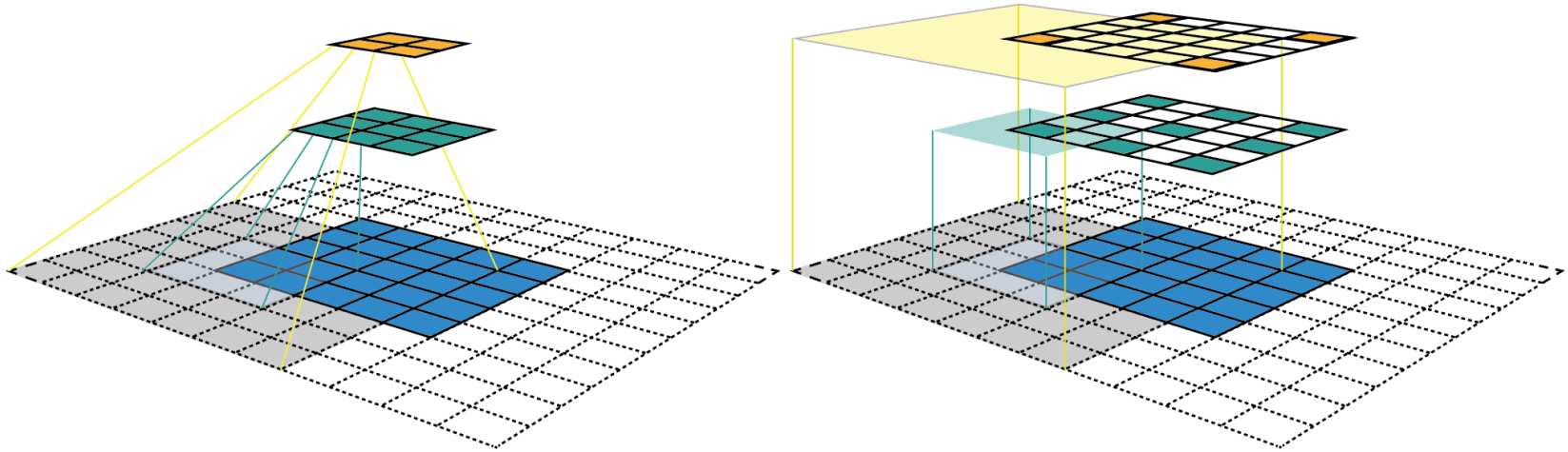
Hierarchical Representation Learning

- Large receptive field

1 layer



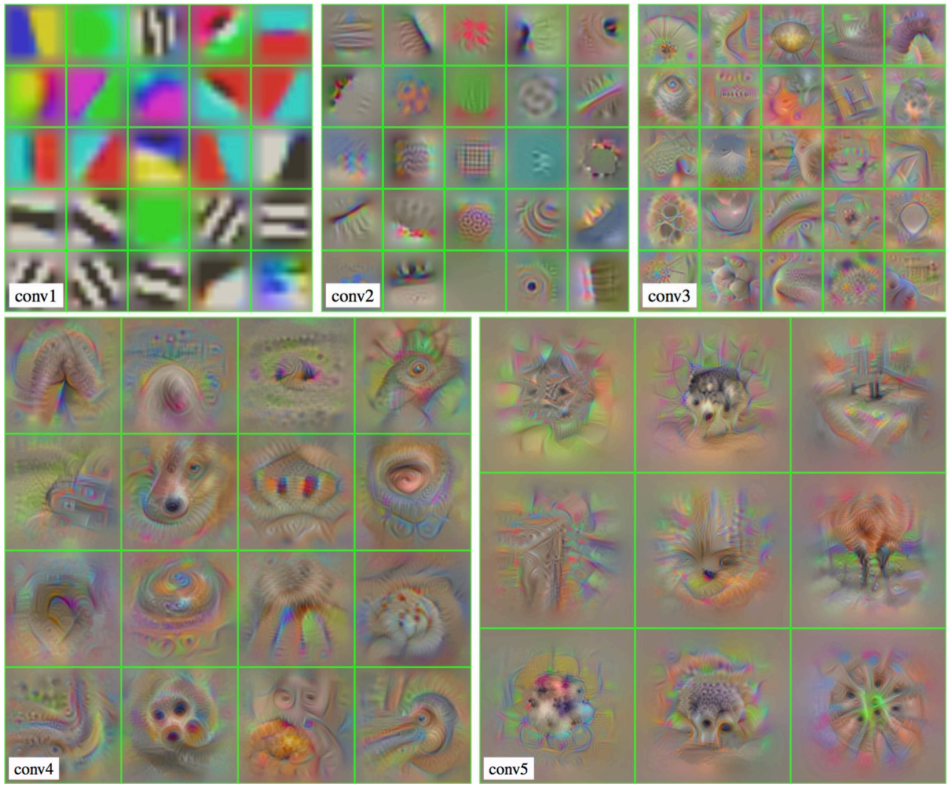
2 layers



Hierarchical Representation Learning

- Visualizing the activation outputs

Activation maximization of the first filters of each convolutional layer in a well-known CNN architecture called VGG. The notations of “conv1” through “conv5” distinct hidden layers, where a larger number represents a deeper hidden layer.



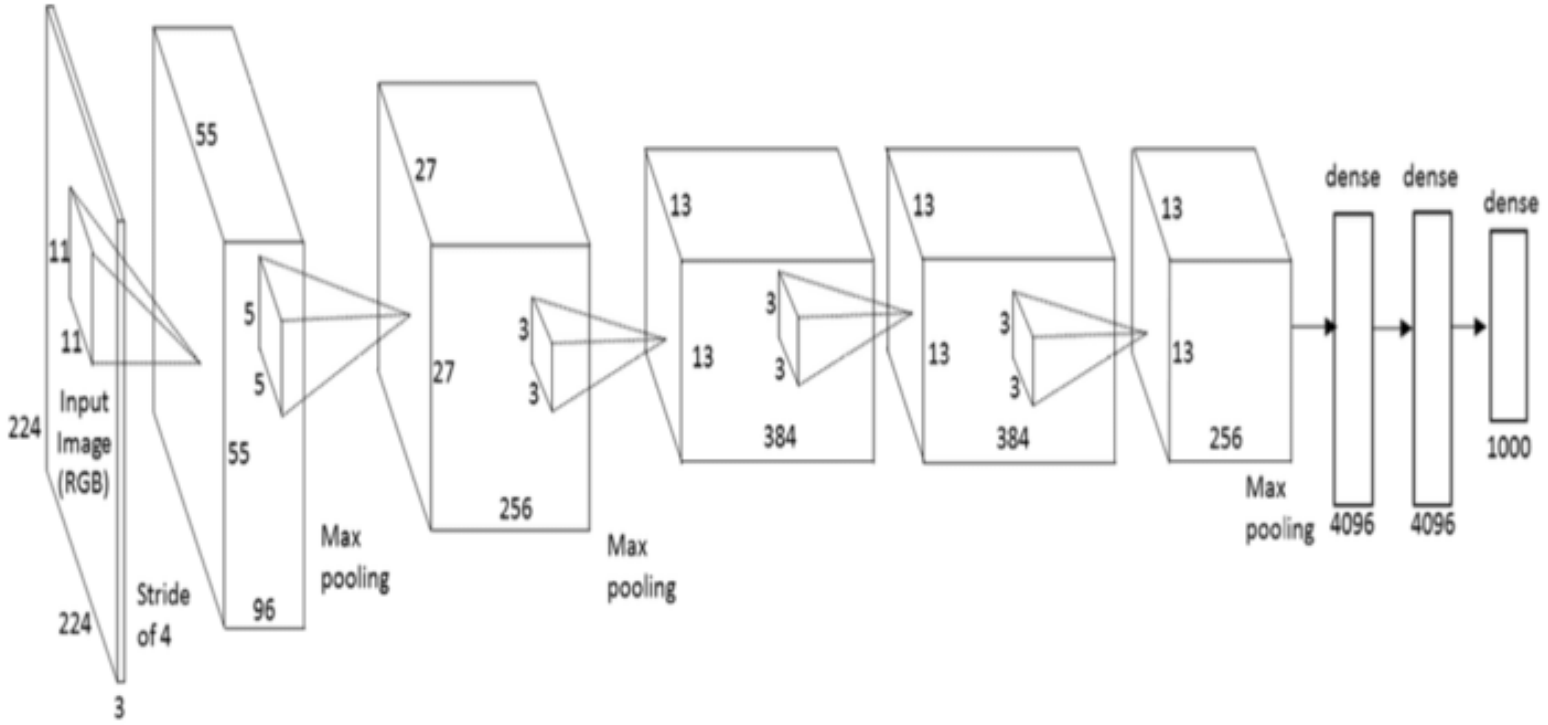
A. Mahendran and A. Vedaldi, “Visualizing deep convolutional neural networks using natural pre-images,” International Journal of Computer Vision (IJCV), vol. 120, no. 3, pp. 233–255, 2016.

北京大學
PEKING UNIVERSITY

Convolutional Architectures

Convolutional Encoding Architectures

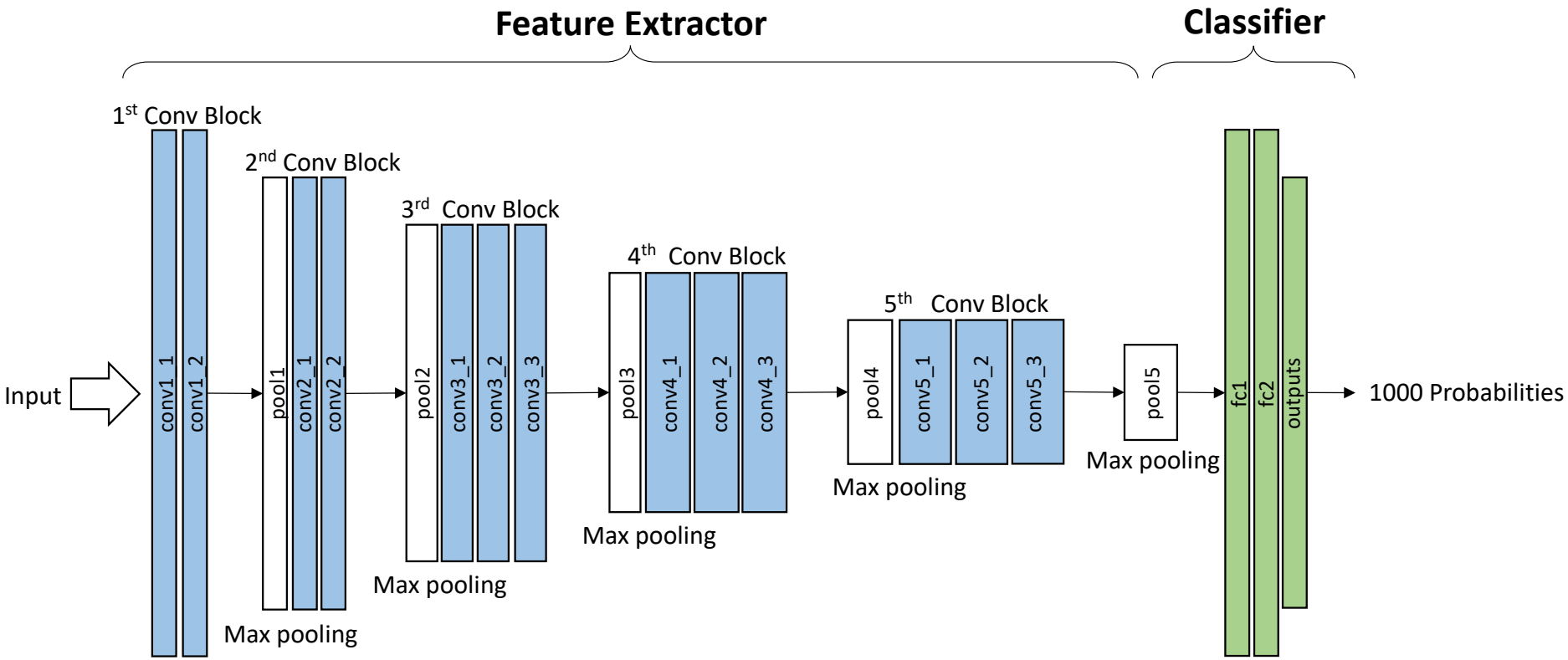
- AlexNet: make the history



In 2012, AlexNet achieves 10% performance gain on ImageNet compared to its previous methods

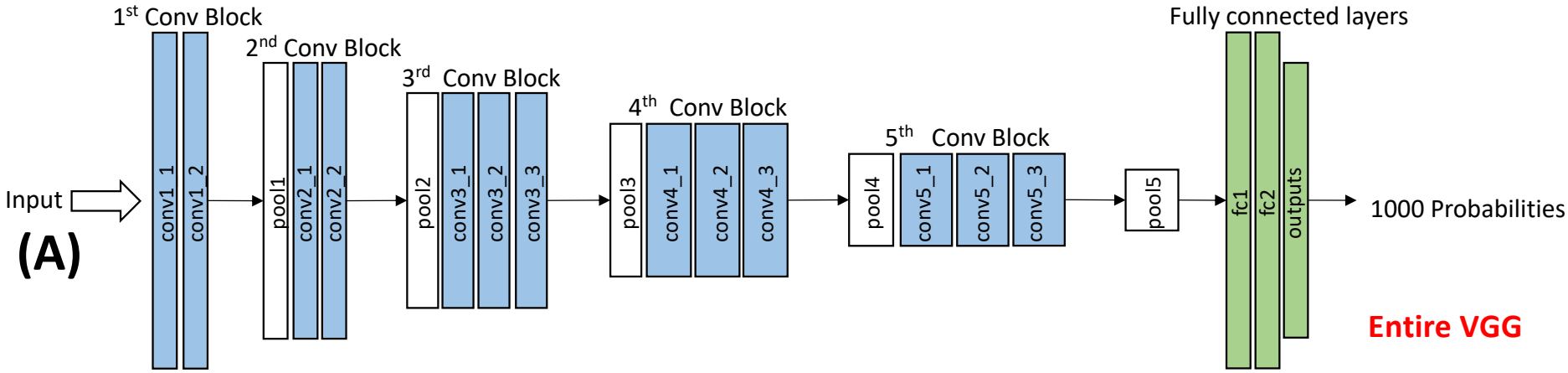
Convolutional Encoding Architectures

- VGG16



Convolutional Encoding Architectures

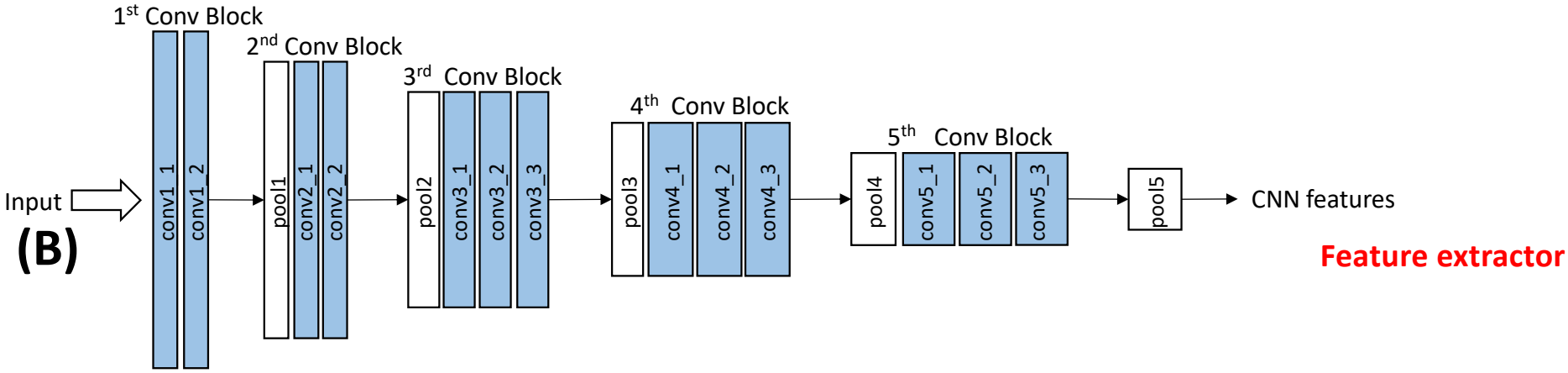
- VGG16




```
>>> # get the whole model, without pre-trained VGG parameters
>>> vgg = tl.models.vgg16()
>>> # get the whole model, restore pre-trained VGG parameters
>>> vgg = tl.models.vgg16(pretrained=True)
>>> # use for inferencing
>>> output = vgg(image, is_train=False)
>>> probs = tf.nn.softmax(output)[0].numpy()
```

Convolutional Encoding Architectures

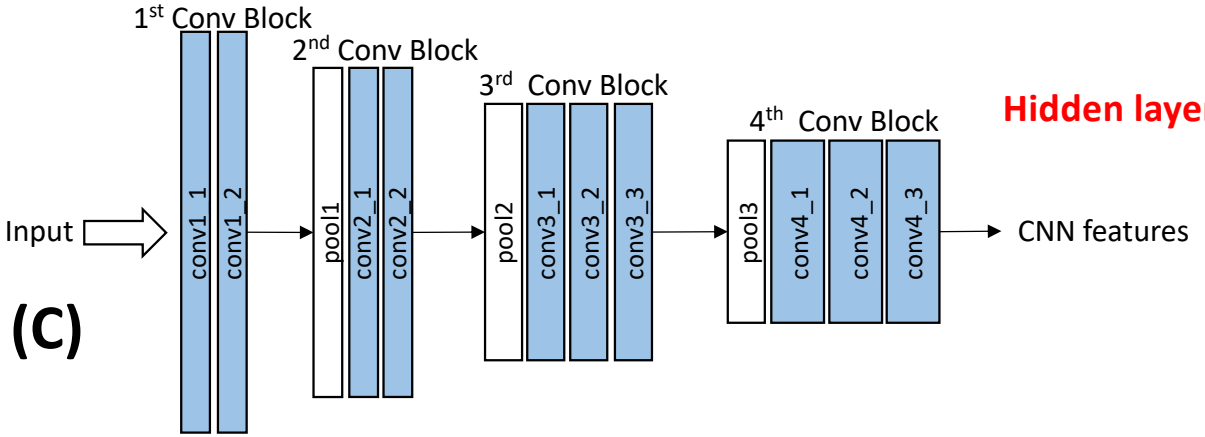
- VGG16




```
>>> vgg = tl.models.vgg16(pretrained=True, end_with='pool5', mode='static')
```

Convolutional Encoding Architectures

- VGG16



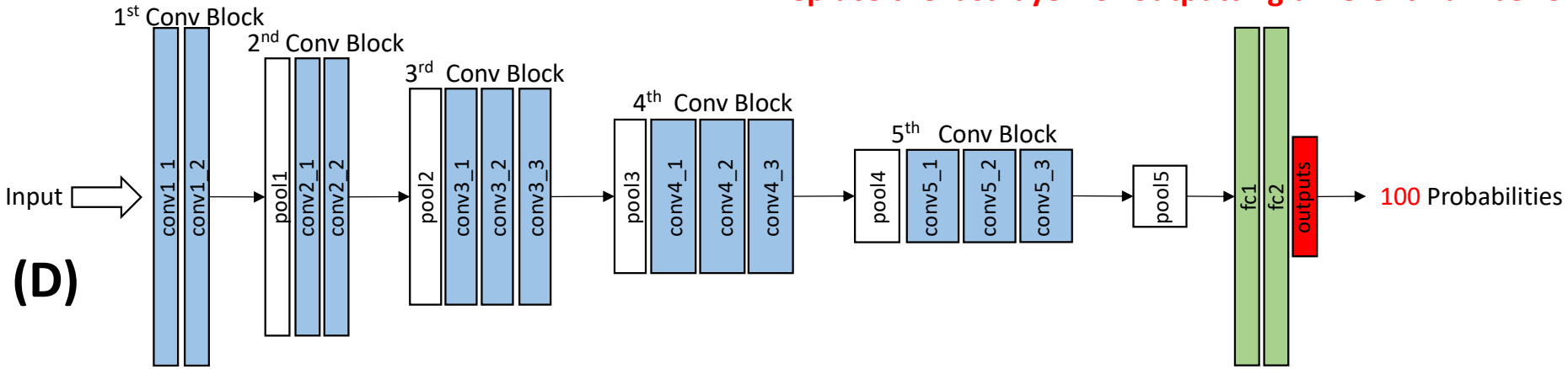
Hidden layer of feature extractor : lower level features

```
>>> vgg = tl.models.vgg16(pretrained=True, end_with='conv4_3', mode='static')
```

Convolutional Encoding Architectures

- VGG16

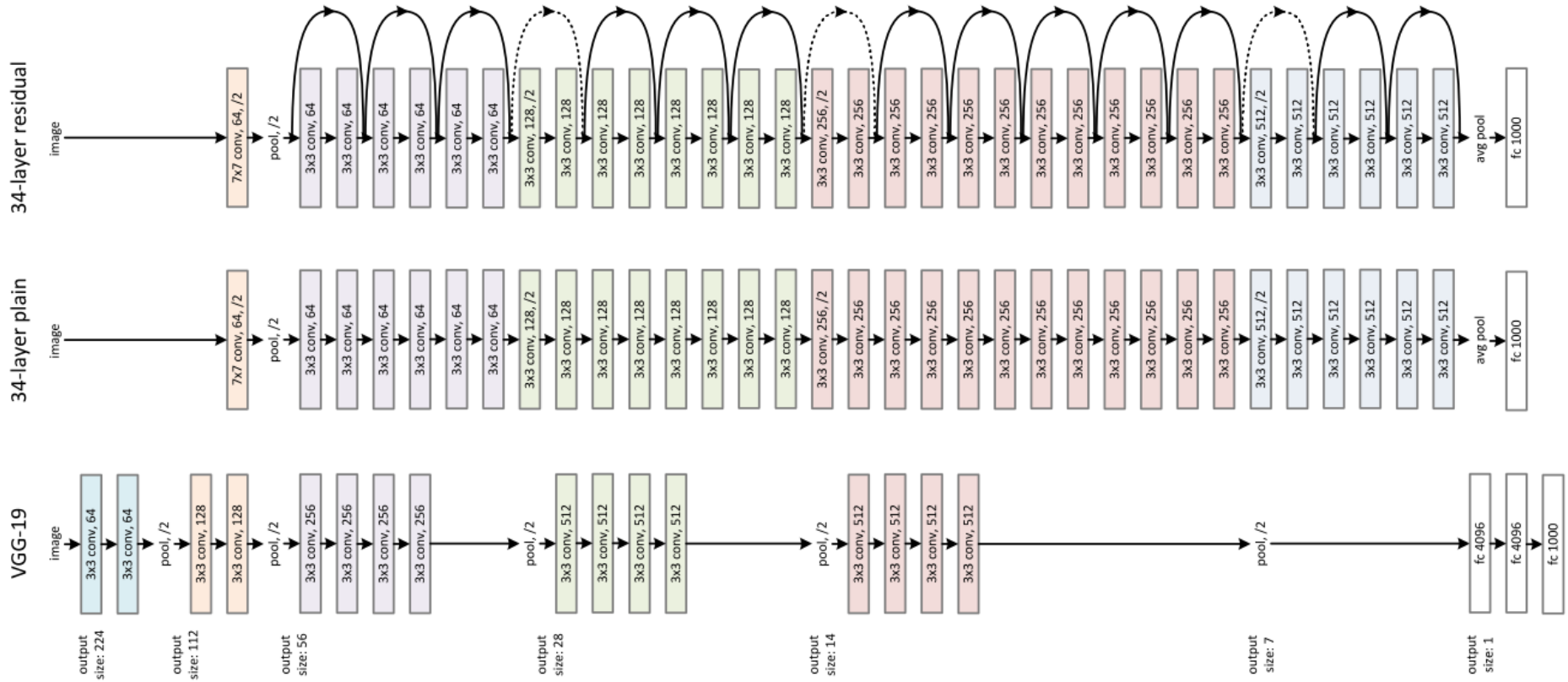
Replace the last layer for outputting different number of classes




```
>>> # get VGG without the last layer
>>> cnn = tl.models.vgg16(end_with='fc2_relu', mode='static').as_layer()
>>> # add one more layer and build a new model
>>> ni = Input([None, 224, 224, 3], name="inputs")
>>> nn = cnn(ni)
>>> nn = tl.layers.Dense(n_units=100, name='out')(nn)
>>> model = tl.models.Model(inputs=ni, outputs=nn)
>>> # train your own classifier (only update the last layer)
>>> train_weights = model.get_layer('out').trainable_weights
```

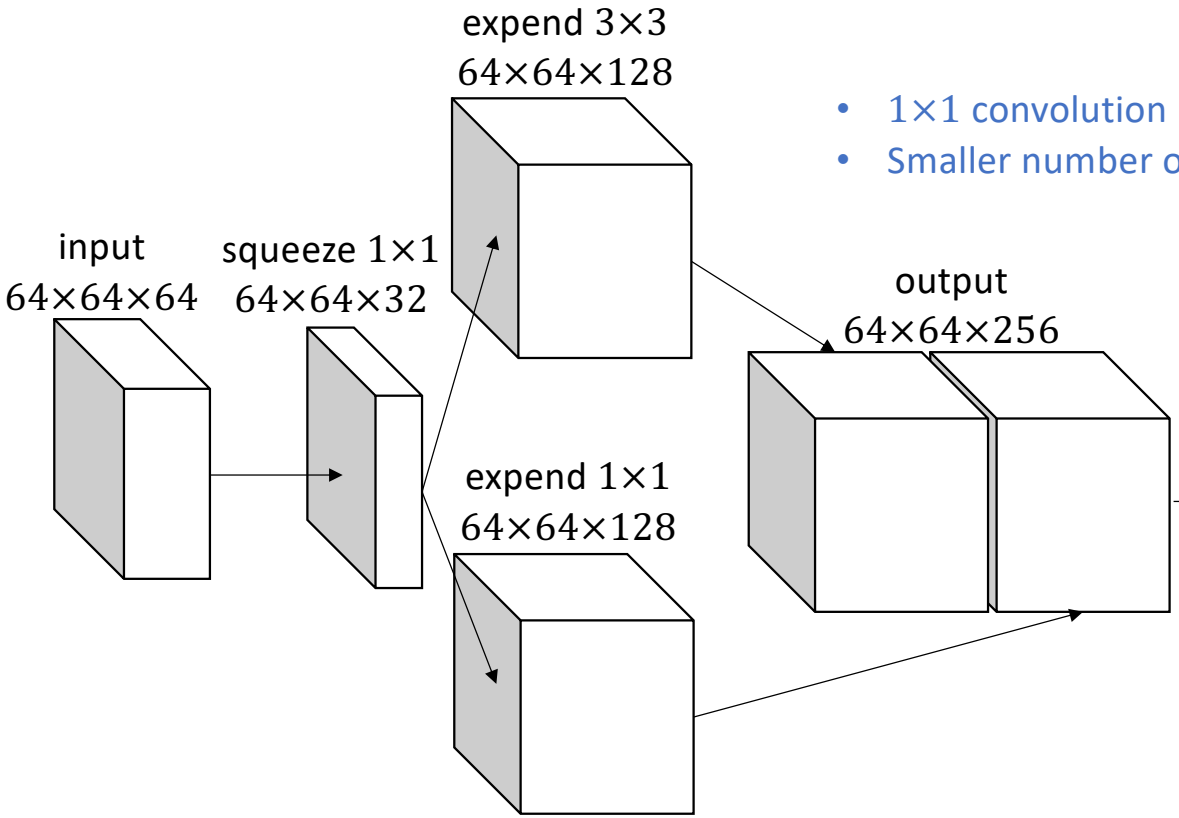
Convolutional Encoding Architectures

- ResNet



Convolutional Encoding Architectures

- SqueezeNet, MobileNet, ShuffleNet



An example of "fire" block

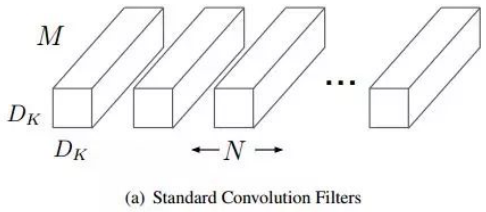
- 1×1 convolution (pointwise conv) is 9 times faster than 3×3 convolution
- Smaller number of channels == Less computation

```
import time
import numpy as np
import tensorflow as tf
import tensorlayer as tl
from tensorlayer.models.imagenet_classes import class_names

squeezenet = tl.models.SqueezeNetV1(pretrained=True)

img1 = tl.vis.read_image('data/tiger.jpeg')
img1 = tl.prepro.imresize(img1, (224, 224)) / 255
img1 = img1.astype(np.float32)[np.newaxis, ...]

start_time = time.time()
output = squeezenet(img1, is_train=False)
prob = tf.nn.softmax(output)[0].numpy()
print("  End time : %.5ss" % (time.time() - start_time))
preds = (np.argsort(prob)[::-1])[0:5]
for p in preds:
    print(class_names[p], prob[p])
```

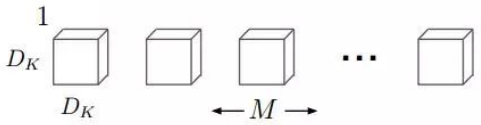
Very Fast even on CPU

Convolutional Encoding Architectures

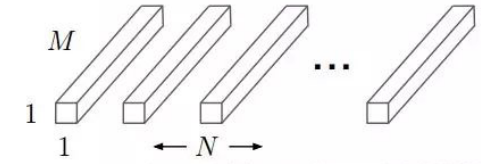
- SqueezeNet, MobileNet, ShuffleNet



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the con-

The number of multiplications of a **standard 2D CNN** layer without strides and padding:

- $filter\_size\_height \times filter\_size\_width \times height \times width \times in\_channels \times output\_channels$

The number of multiplications of a **depthwise 2D CNN** layer without strides and padding:

- $filter\_size\_height \times filter\_size\_width \times height \times width \times in\_channels$

The number of multiplications of a **pointwise 2D CNN** layer without strides and padding:

- $height \times width \times in\_channels \times output\_channels$

Convolutional Encoding Architectures

- SqueezeNet, MobileNet, ShuffleNet

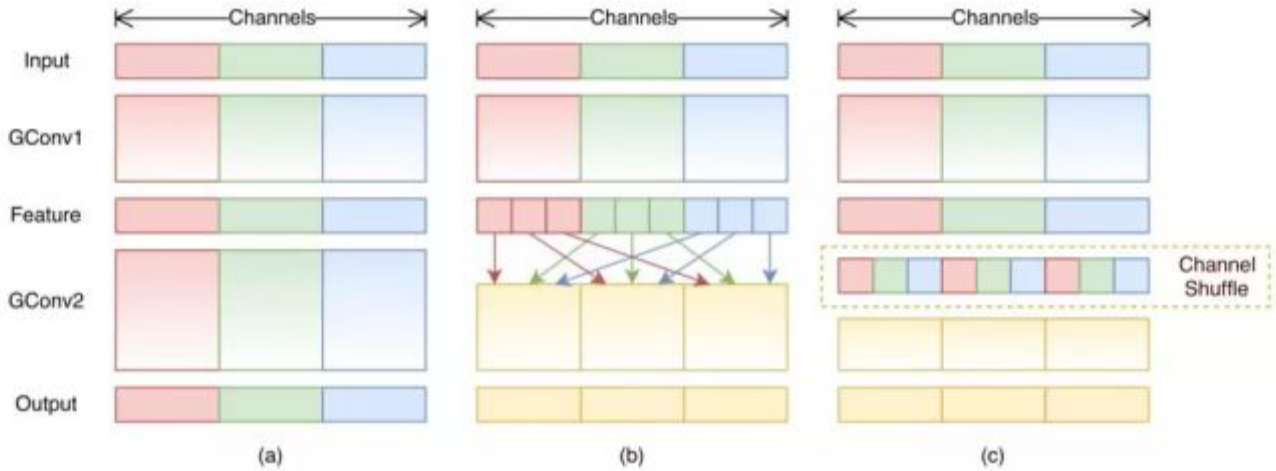


Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.

Group Convolution: Split the channels into several "ground" and perform standard convolution using different CNN layers.

Shuffle layer: Merge the information of different group by shuffling the channels.

Convolutional Encoding Architectures

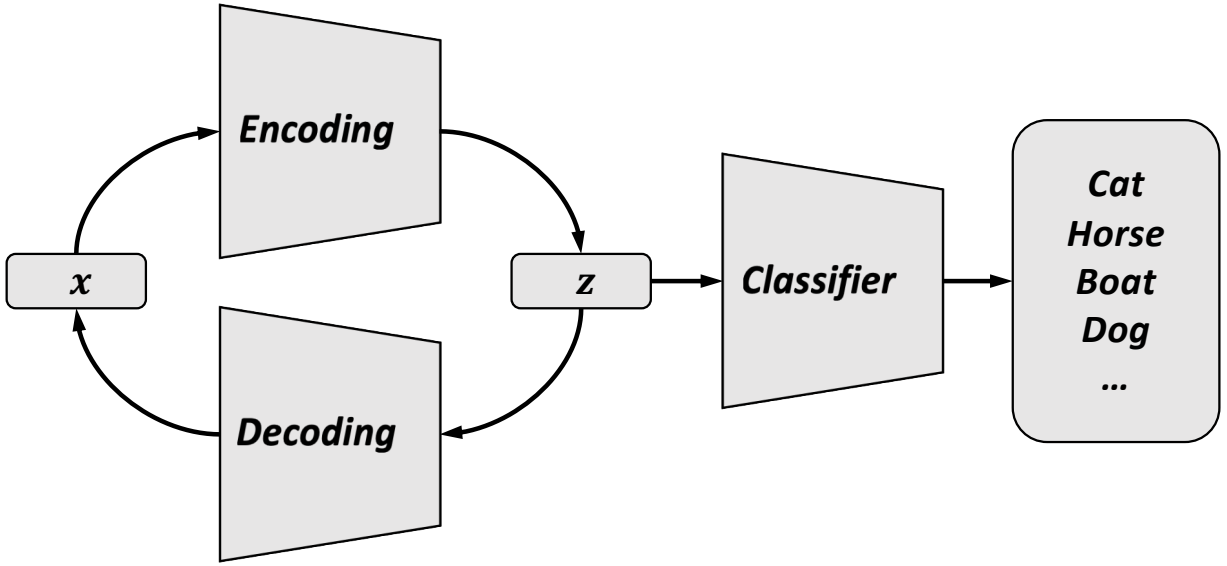
- SqueezeNet++
- MobileNetV2
- MobileNetV3
- ShuffleNetV2
- ...

Transposed Convolutional Algorithms

Transposed Convolutional Algorithms

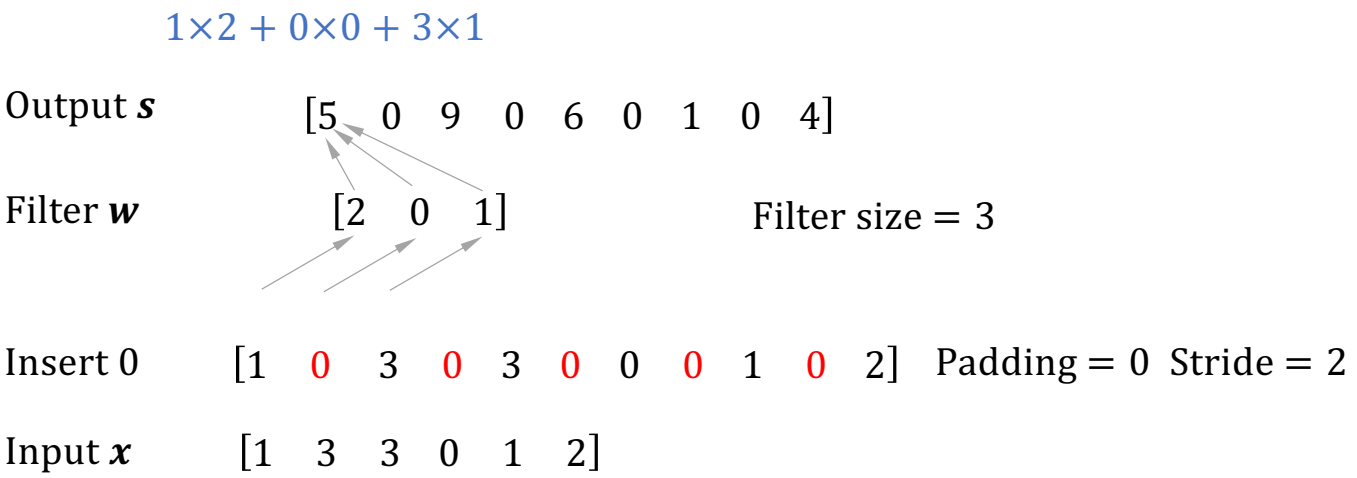
- Motivation

Convolution and pooling can only keep or reduce the size of the feature maps



Transposed Convolutional Algorithms

- Transposed convolution on 1D vector

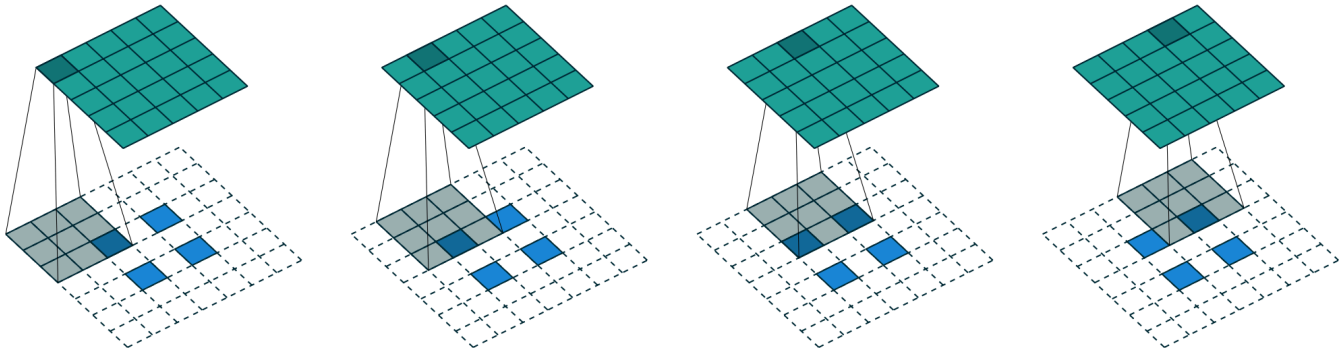


Transposed Convolutional Algorithms

- Inserting zeros between inputs

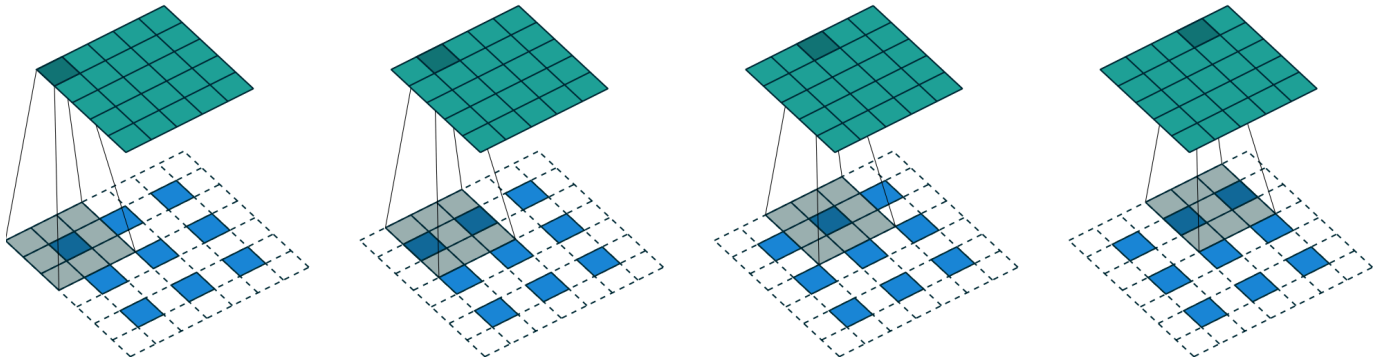
Padding , strides and transposed

Padding = 2×2
Strides = 2×2



Padding, strides and transposed

Padding = 1×1
Strides = 2×2



Transposed Convolutional Algorithms

- Resize convolution
- Subpixel convolution
- …