3011979 Intro to Deep Learning for Medical Imaging

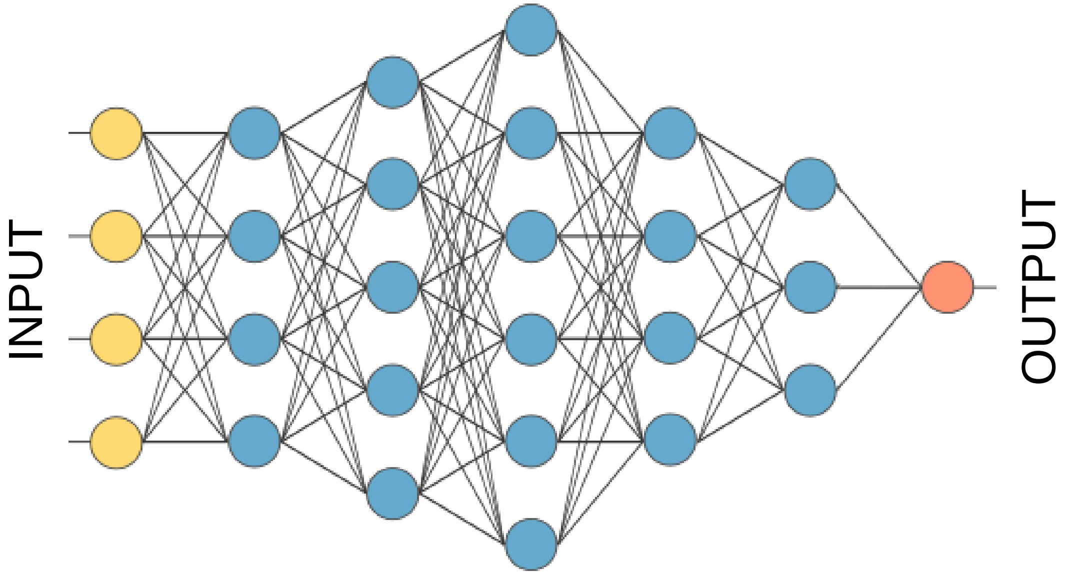
L11: Neural network architectures (for imaging data)

Apr 16th, 2021

Sira Sriswasdi, Ph.D.
Research Affairs, Faculty of Medicine
Chulalongkorn University

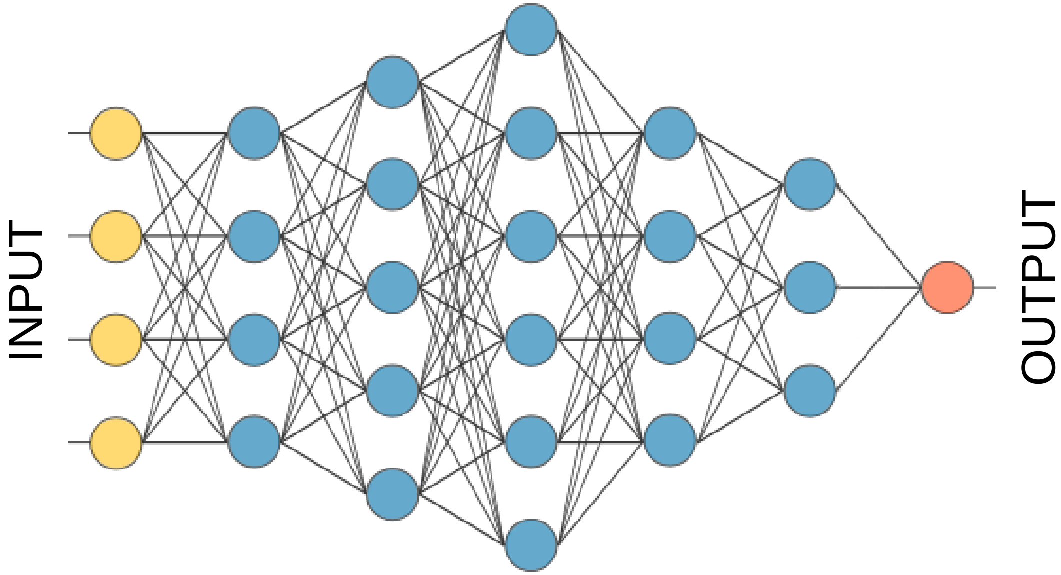
Limitations of multilayer perceptron (fully connected / dense network)

Fully connected network – powers



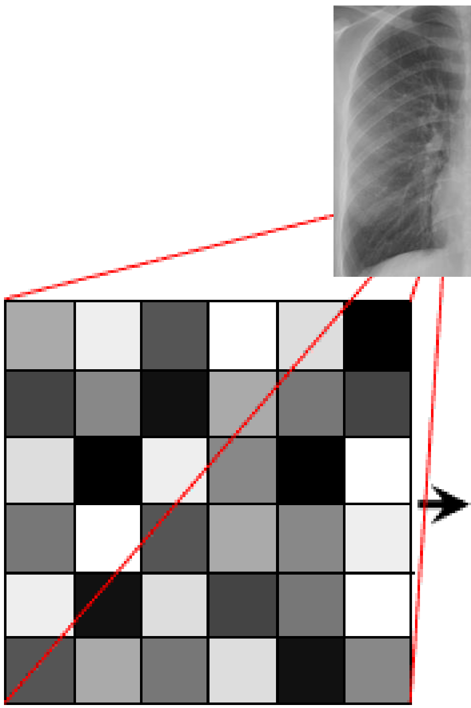
- Generate powerful predictor as a complex function of input features
- Can approximate any real-value function with simple, non-polynomial activation function

Fully connected network – limitations



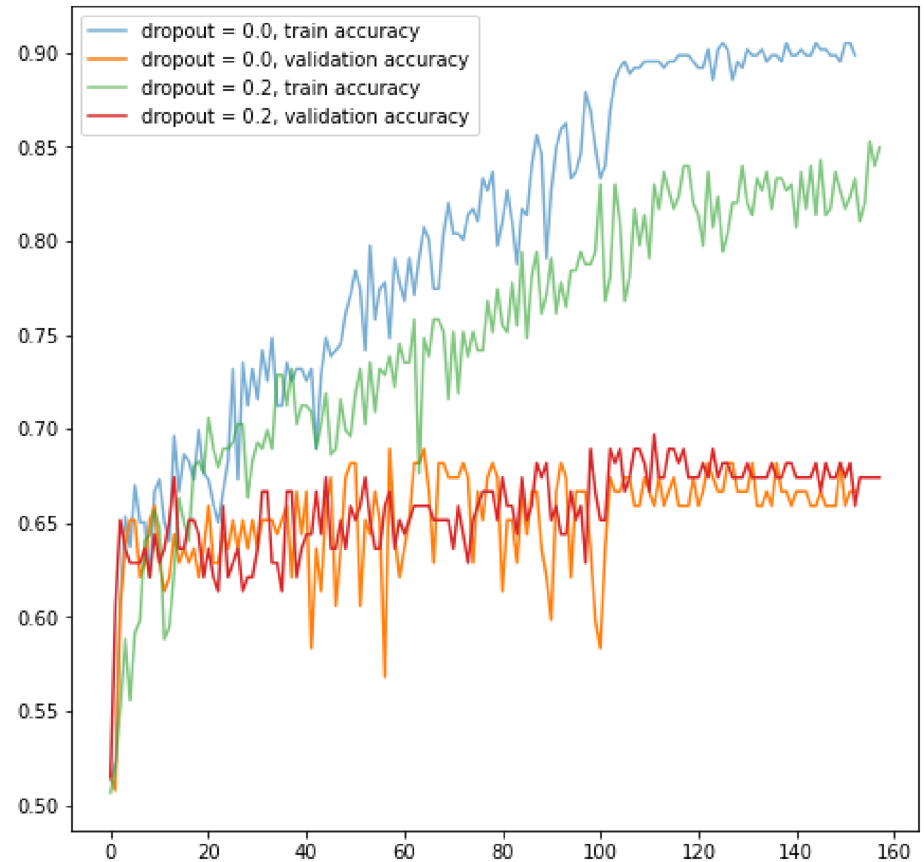
- What if features have contextual relationships?
  - Adjacent pixels in an image
  - Adjacent words in a sentence
- What if input size is variable?
  - Number of words in a medical report

Fully connected network's image perception



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Image from naushardsblog.wordpress.com



- Fully connected network cannot learn anything from pixel values of chest x-ray images

Image object

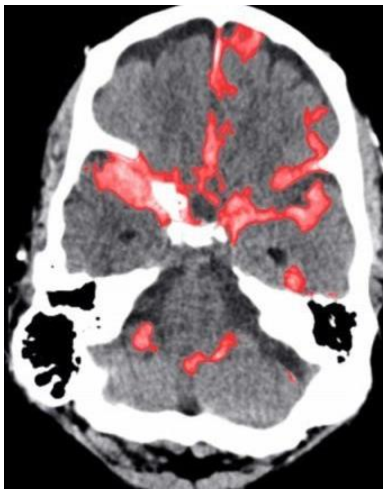
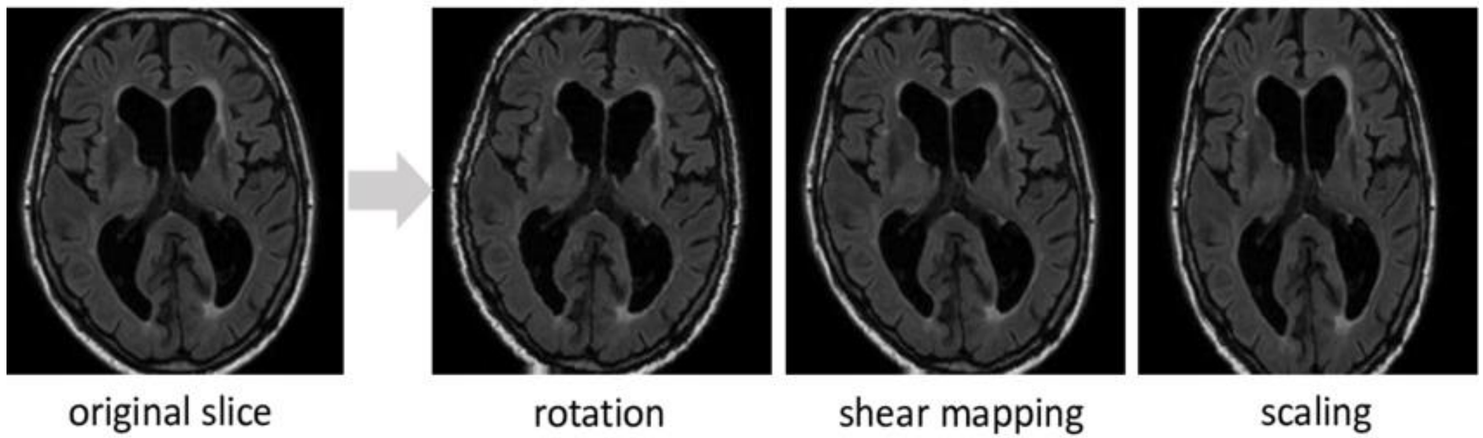


Image from tectales.com



Li et al. Neuroimage 183 (2018)

- An image object is a collection of related pixels
- Remain the same when rotated or translated
    - Fully connect network will perceive these variations as totally different inputs

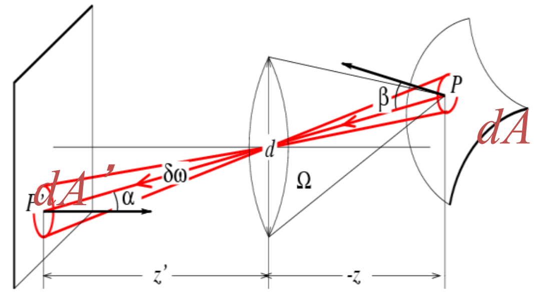
Computer vision

(Credit: Minh Hoai Nguyen, MLRS2019)

Sciences of image

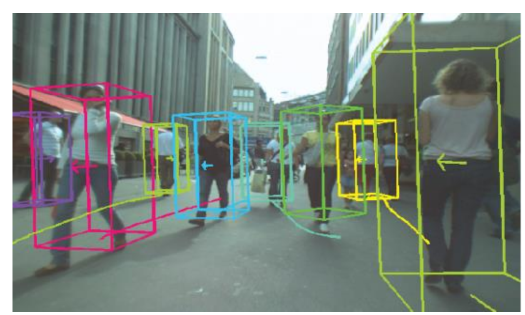
- Image processing
  - Image to image

- Imaging
  - Physics to image



- Graphics
  - Symbols to image

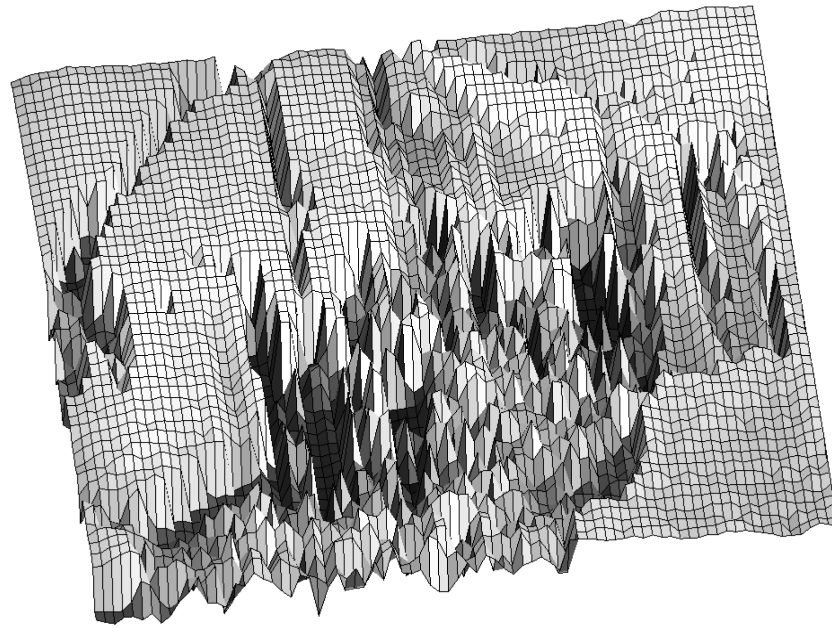
- Computer vision
  - Image to symbols



Challenges of CV

Image through human eyes

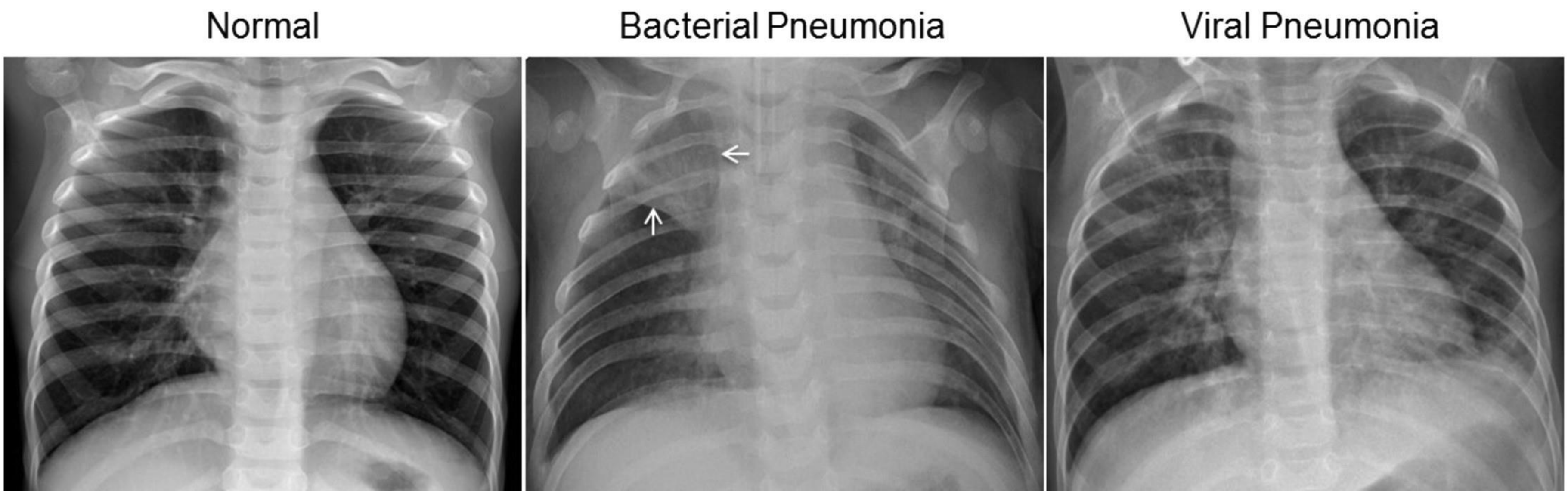
Image in computer



- We see more than just colors and numbers
- Try describing the concept “ugly” to a blind man
  - Or to a computer

Context matters a lot

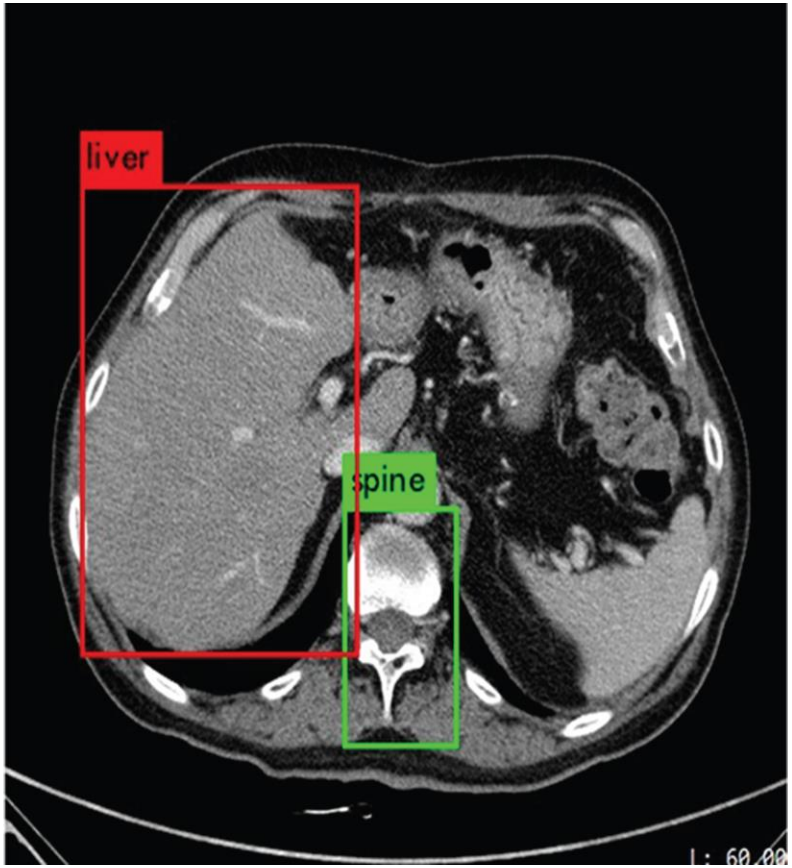
Main tasks in CV – image classification



Kermany et al. Cell (2018)

- Input: An image
- Output: Single-label or multi-label classification

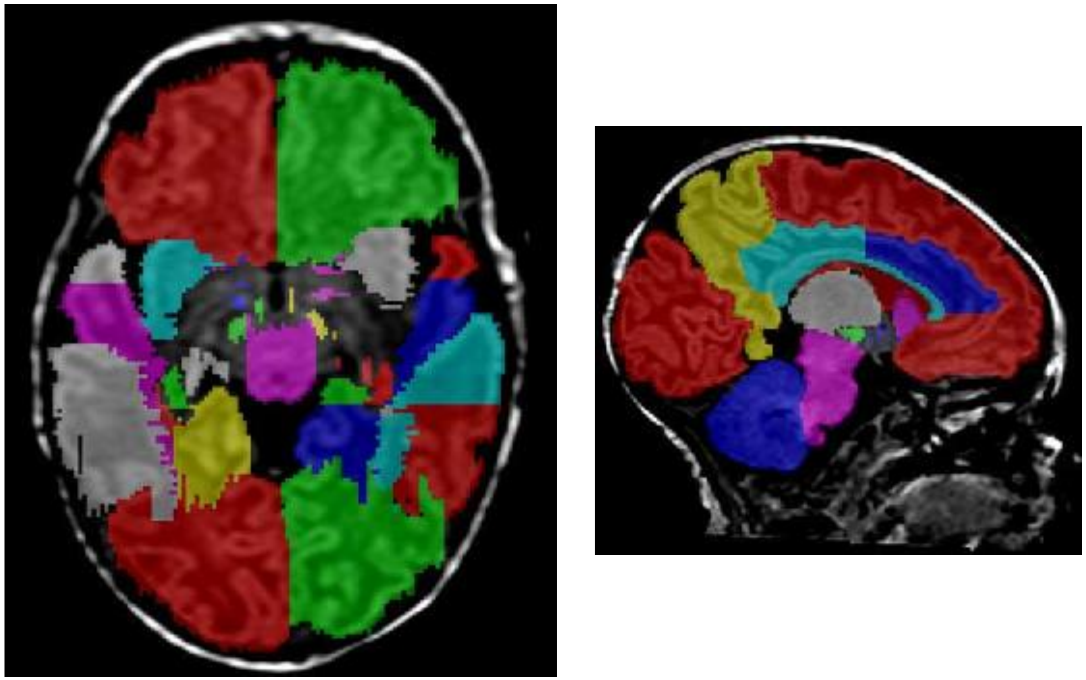
Main tasks in CV – object detection



Pang et al. PLoS ONE (2019)

- Input: An image
- Output: Bounding box + Object label

Main tasks in CV – semantic segmentation



Makropoulos et al. IEEE Trans Med Imaging (2014)

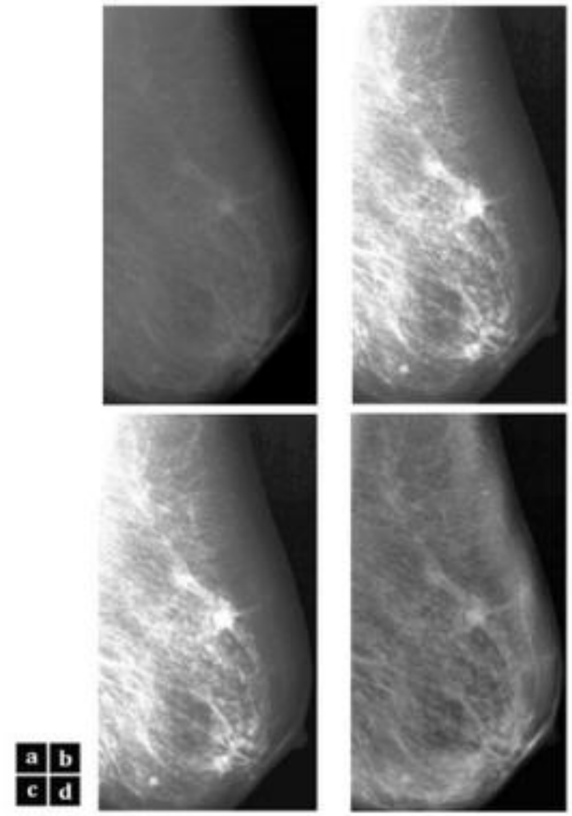
- Input: An image
- Output: Pixel-level classification (and more)

Image processing

Contrast adjustment

a. Original Image
c. Linear Streatching
b. Histogram Equilization
d. Adaptive Histogram Equilization

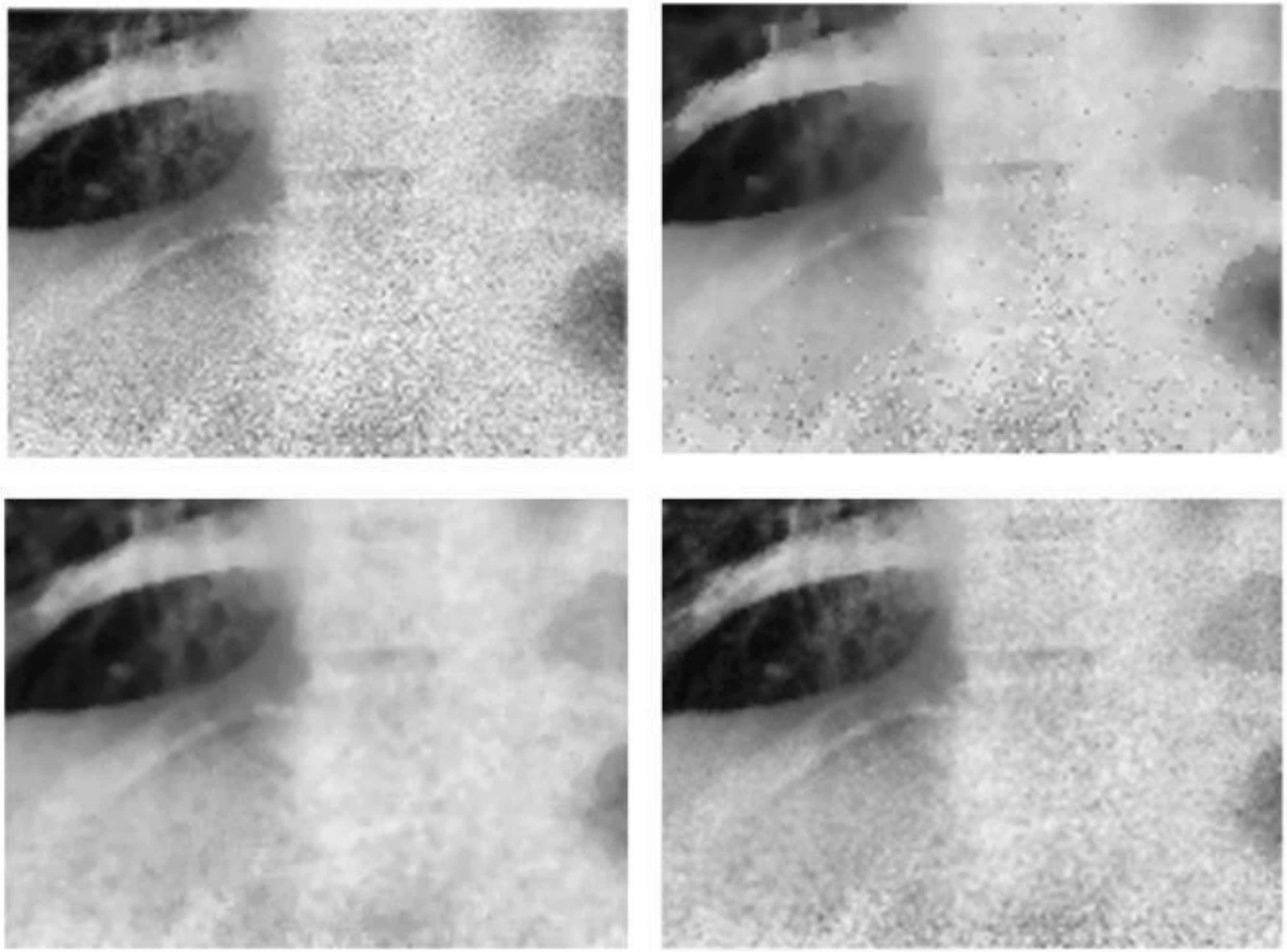
a. Original Image
c. Linear Streatching
b. Histogram Equilization
d. Adaptive Histogram Equilization



a. Original Image
c. Linear Streatching
b. Histogram Equilization
d. Adaptive Histogram Equilization

Ahmed et al. TechnoMoot conference (2011)

Noise filtering



a b
c d

a. Croped CLAHE image
b. Sigma Filter
c. Median Filter
d. Wiener Filter

Ahmed et al. TechnoMoot conference (2011)

Mean filtering operation

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0								

- Setting center pixel = average of 3x3 area

Mean filtering operation

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10							

- Setting center pixel = average of 3x3 area

Mean filtering operation

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20						

- Setting center pixel = average of 3x3 area

Mean filtering operation

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

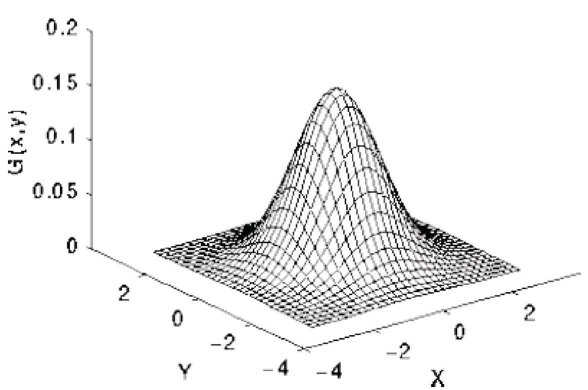
- Output is a smoothed-out version of the input image

Filtering as mathematical operation

- 3x3 mean filtering
- 5x5 Gaussian filtering
- Integrate contextual info!

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

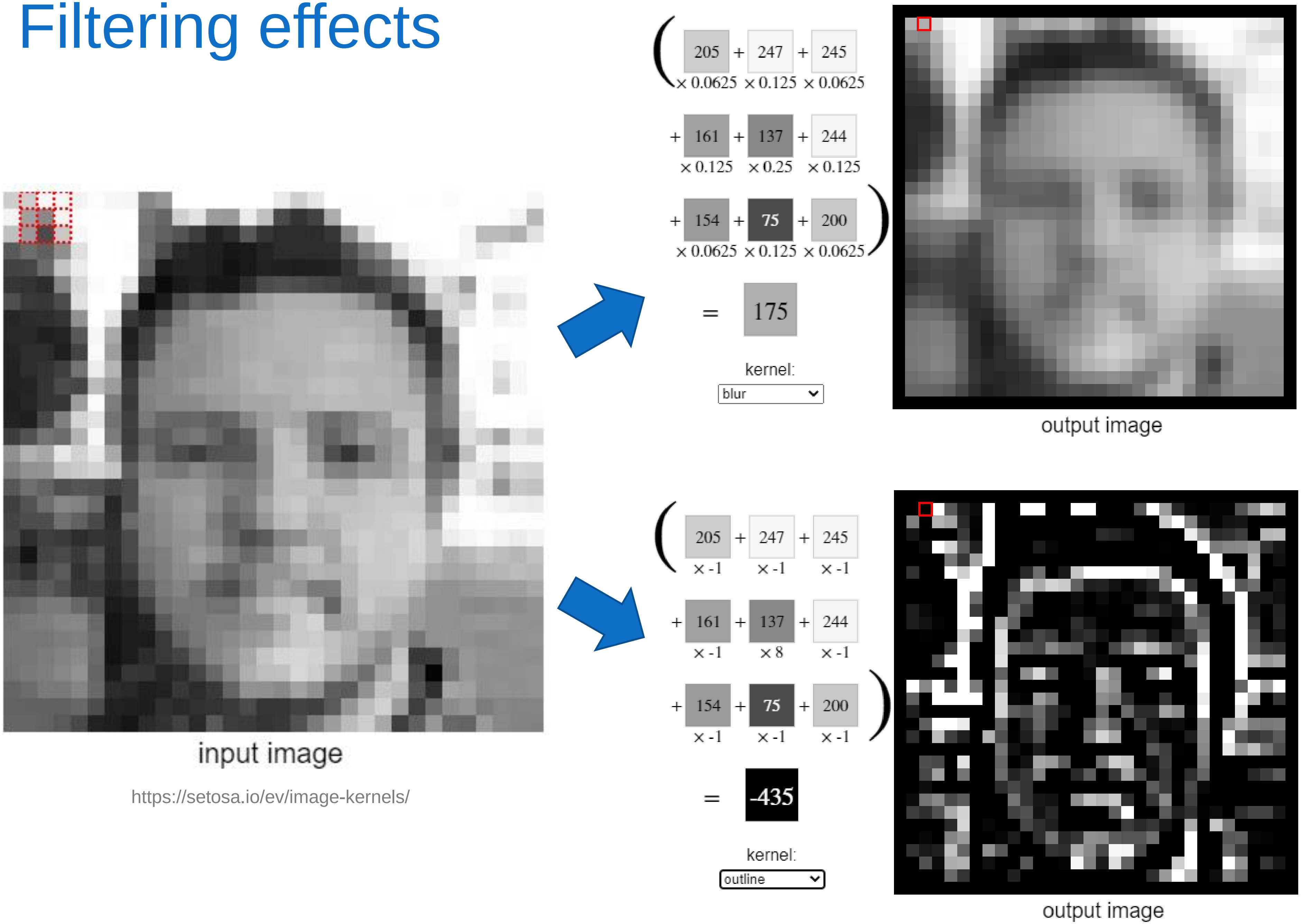


$\sigma = 1$

$\frac{1}{115}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

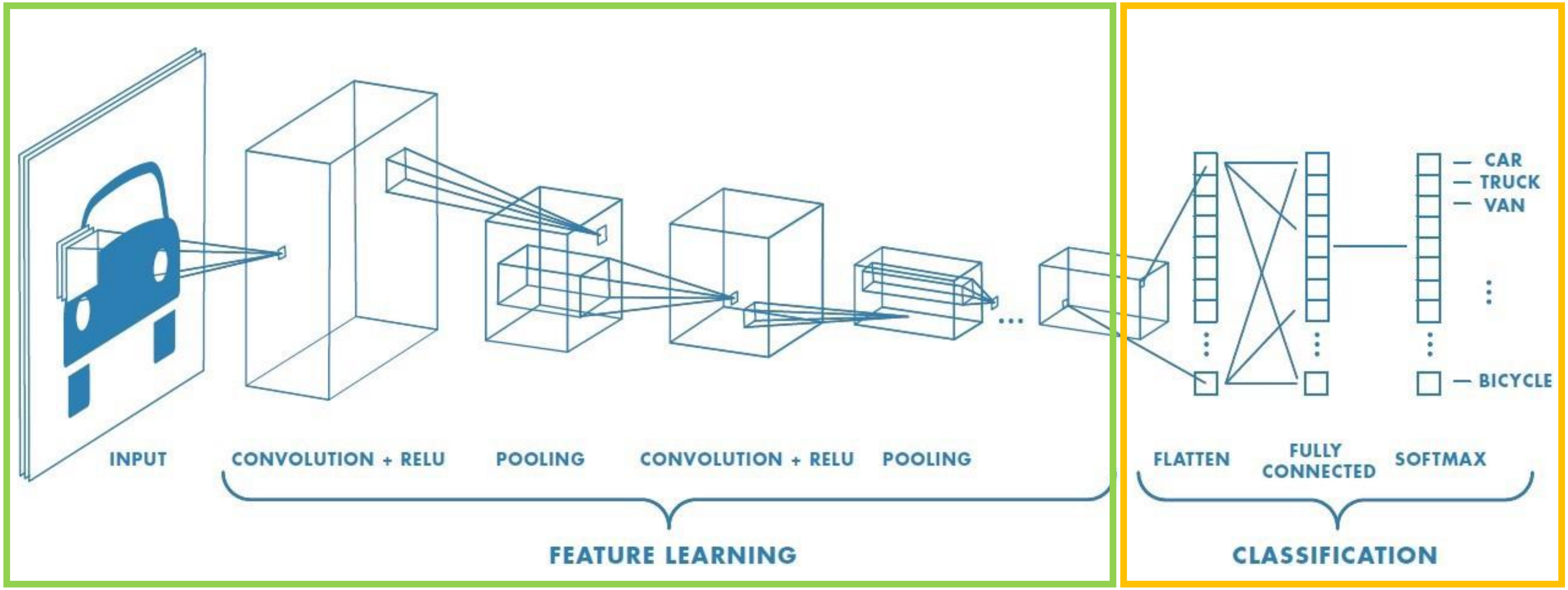
Filtering effects



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

Convolutional neural network

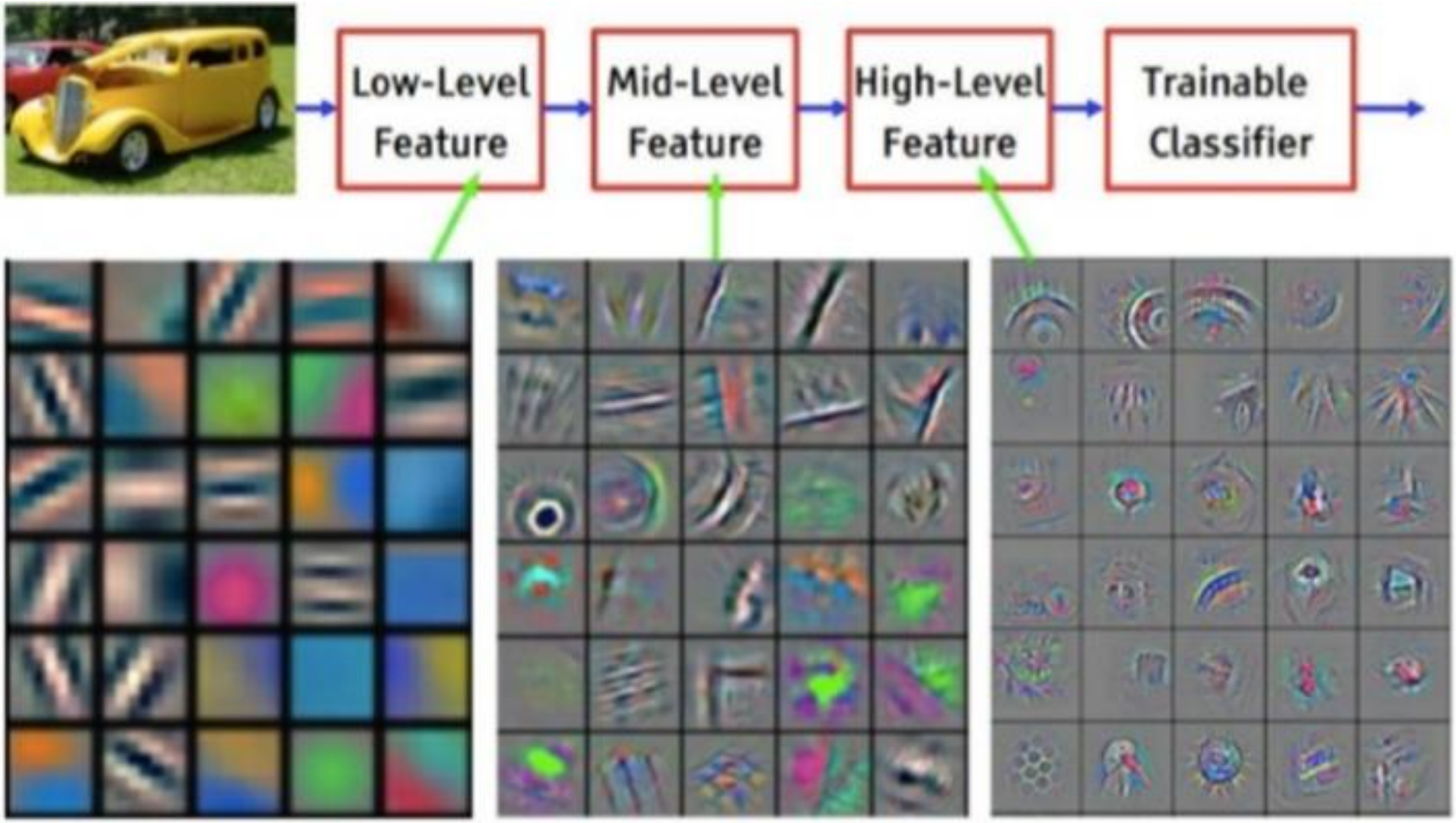
Convolutional neural network



Source: towardsdatascience.com by Saha, S.

- Data-driven, automatic search for optimal filters
  - Similar objective as radiomics, but no hand-crafting
- CNN layers extract features + fully-connect layers performing prediction

Capability of deep CNN



Source: Zeiler and Fergus (2013)

- Early layers produce simple geometric features
- Deeper layers produce complex shapes and patterns

Backpropagation through CNN

- Input: 1D signal $[x_1, x_2, \dots, x_d]$
- Convolution: 1x2 filter $[w_1, w_2]$
- Output of CNN:
  - $y_1 = x_1 w_1 + x_2 w_2$
  - $y_2 = x_2 w_1 + x_3 w_2$
  - …
  - $y_{d-1} = x_{d-1} w_1 + x_d w_2$
- Output of network:
  - $\hat{y} = \text{sigmoid}(y_1 u_1 + y_2 u_2 + \cdots + y_{d-1} u_{d-1})$
- Loss: $L(\hat{y}, y)$

- We can use chain rule to compute $\frac{\delta L}{\delta w_1}$ and $\frac{\delta L}{\delta w_2}$

Convolutional layer in Tensorflow

tf.keras.layers.Conv2D

See Stable See Nightly

TensorFlow 1 version View source on GitHub

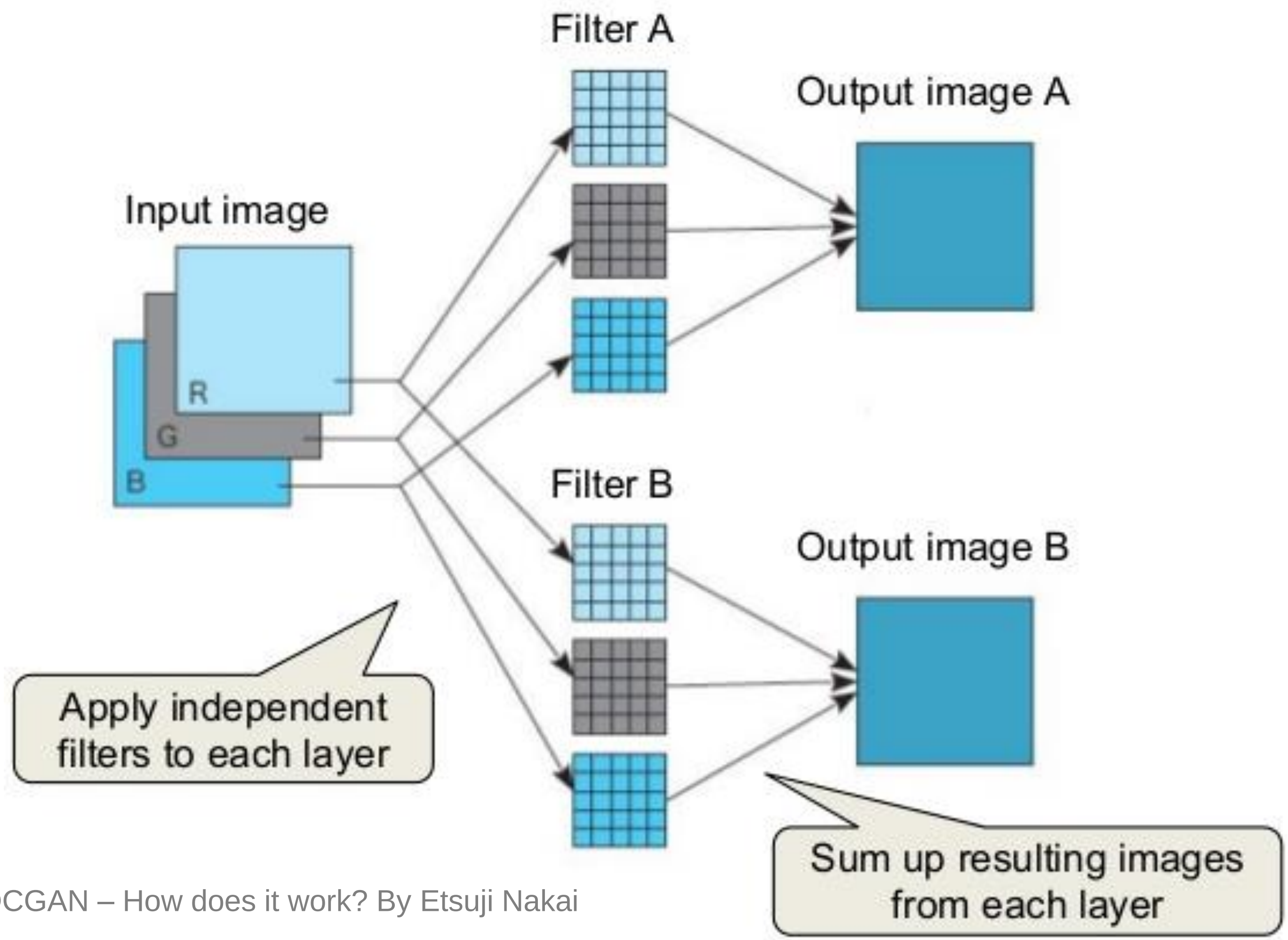
2D convolution layer (e.g. spatial convolution over images).

Inherits From: `Layer`, `Module`

View aliases

```
tf.keras.layers.Conv2D(
    filters, kernel_size, strides=(1, 1), padding='valid',
    data_format=None, dilation_rate=(1, 1), groups=1, activation=None,
    use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None,
    bias_regularizer=None, activity_regularizer=None, kernel_constraint=None,
    bias_constraint=None, **kwargs
)
```

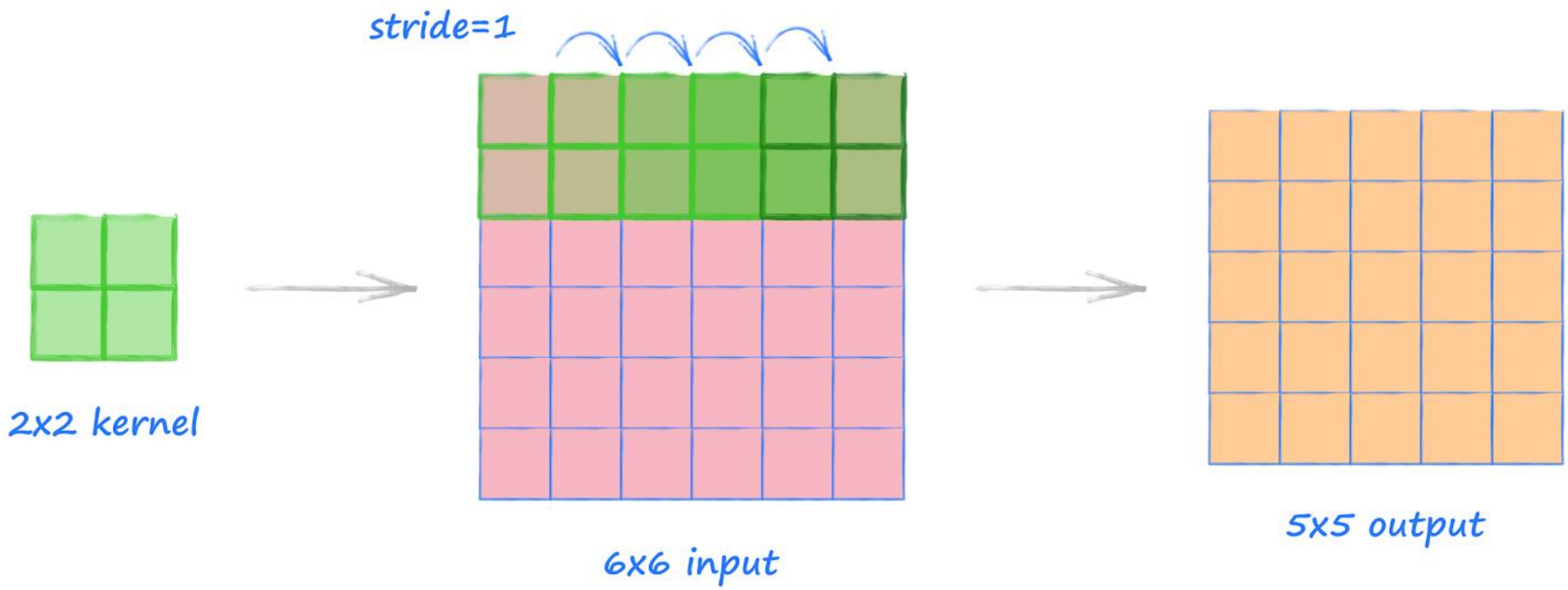
Number of filters



Source: DCGAN – How does it work? By Etsuji Nakai

- More filters generates more geometric features

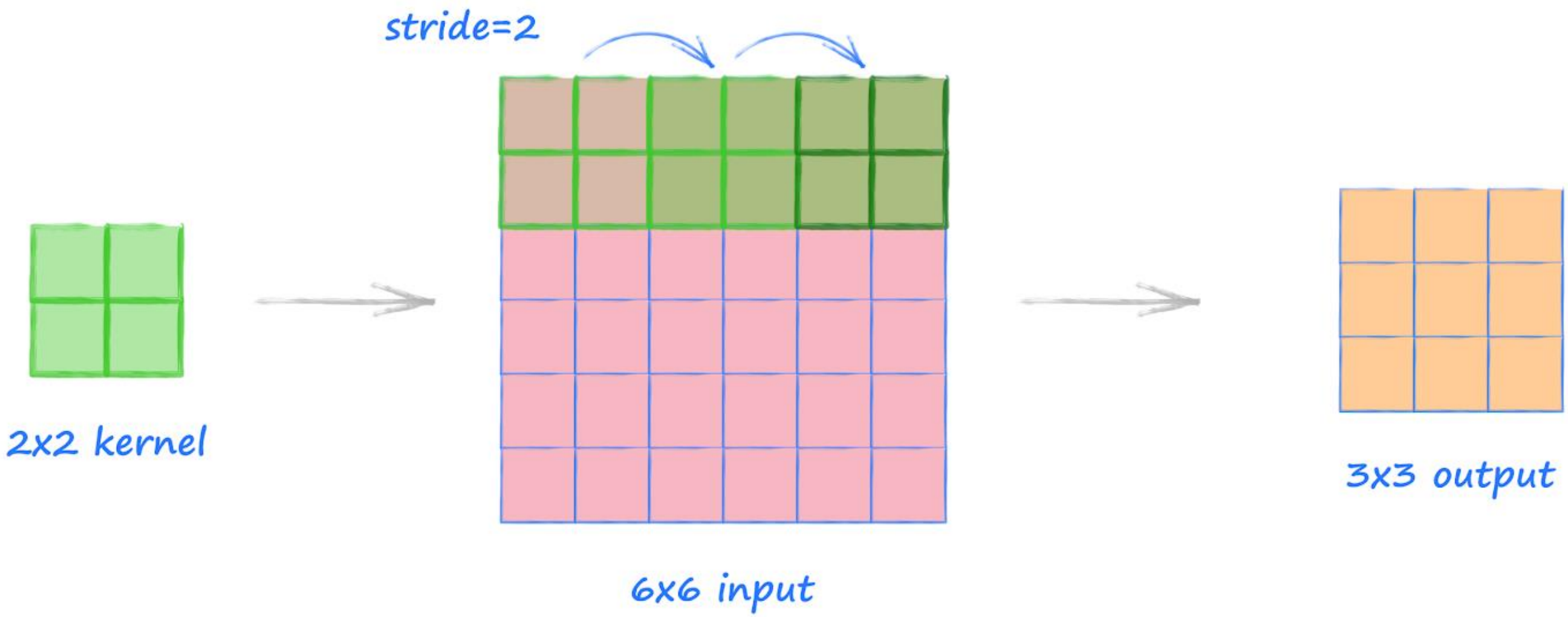
Kernel size and stride



Source: makeyourownneuralnetwork.blogspot.com

- The size of each filter or kernel is *k* x *k*
  - Larger filter incorporates broader contextual data
- Stride = frequency at which to apply a filter

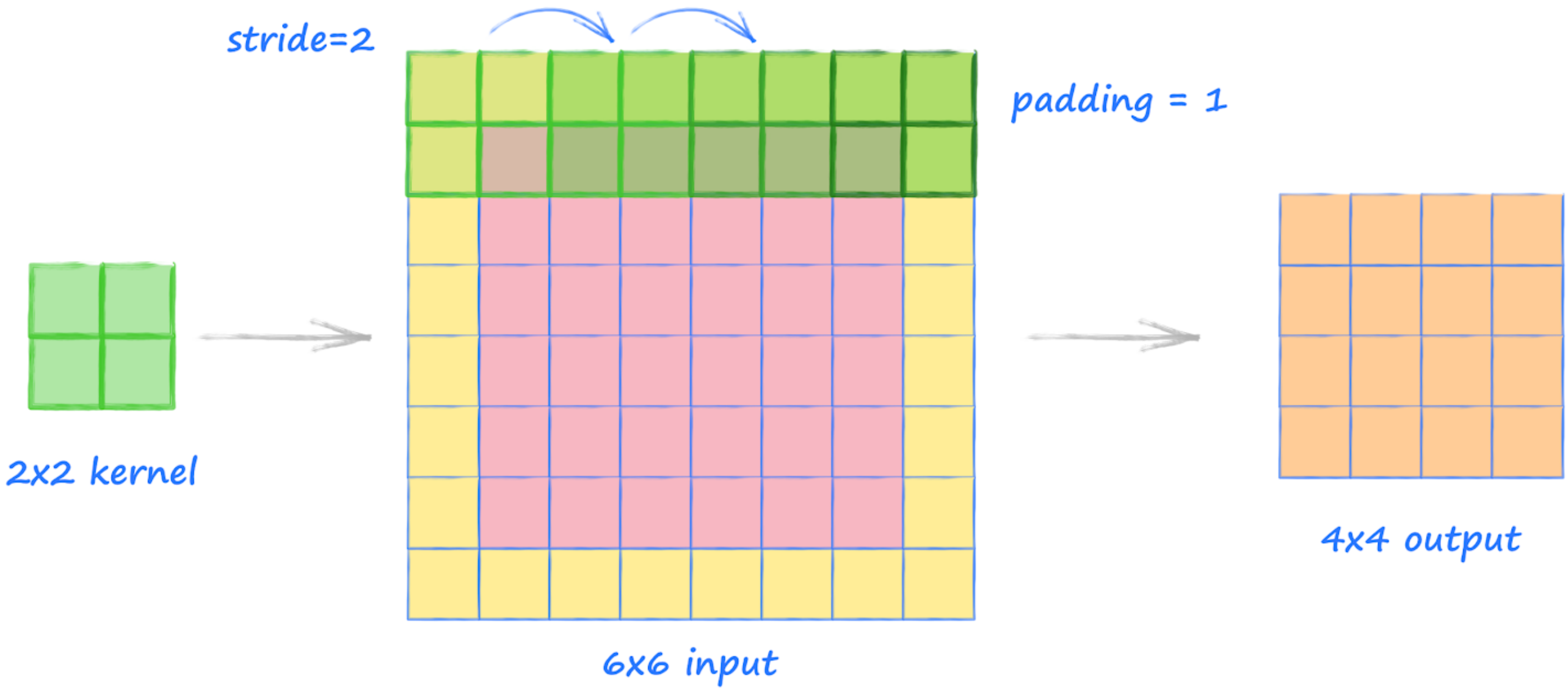
Stride



Source: makeyourownneuralnetwork.blogspot.com

- Larger stride further reduces the output size in exchange for some loss in resolution

Padding



Source: makeyourownneuralnetwork.blogspot.com

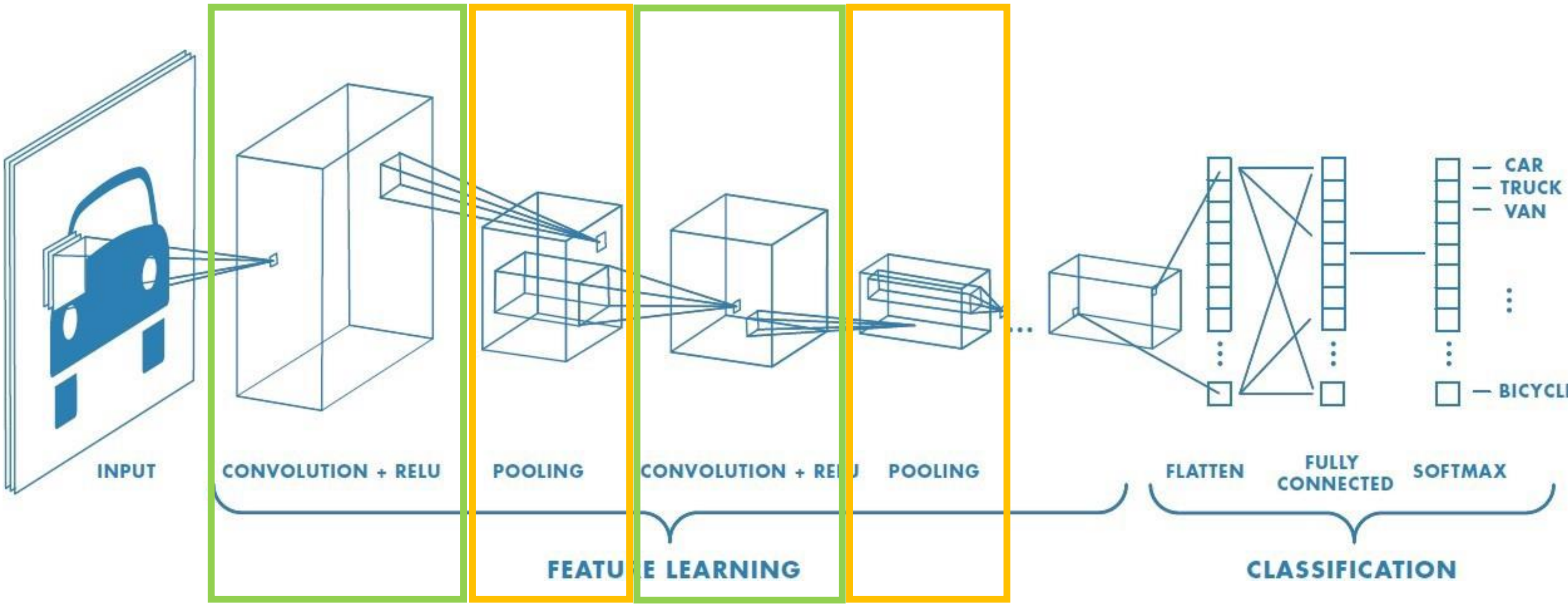
- Padding = filling in zeros to the exterior of image
  - padding = “valid” → no padding
  - padding = “same” → output size = input size

Number of parameters in CNN

- Setting A
  - tf.keras.layers.Conv2D(filters = 4, kernel_size = (9, 9), strides = (1, 2), padding = ‘valid’)
  - Input image size = 128 x 128

- Setting B
  - tf.keras.layers.Conv2D(filters = 32, kernel_size = (3, 3), strides = (1, 2), padding = ‘valid’)
  - tf.keras.layers.Conv2D(filters = 8, kernel_size = (3, 3), strides = (1, 2), padding = ‘valid’)
  - Input image size = 64 x 64

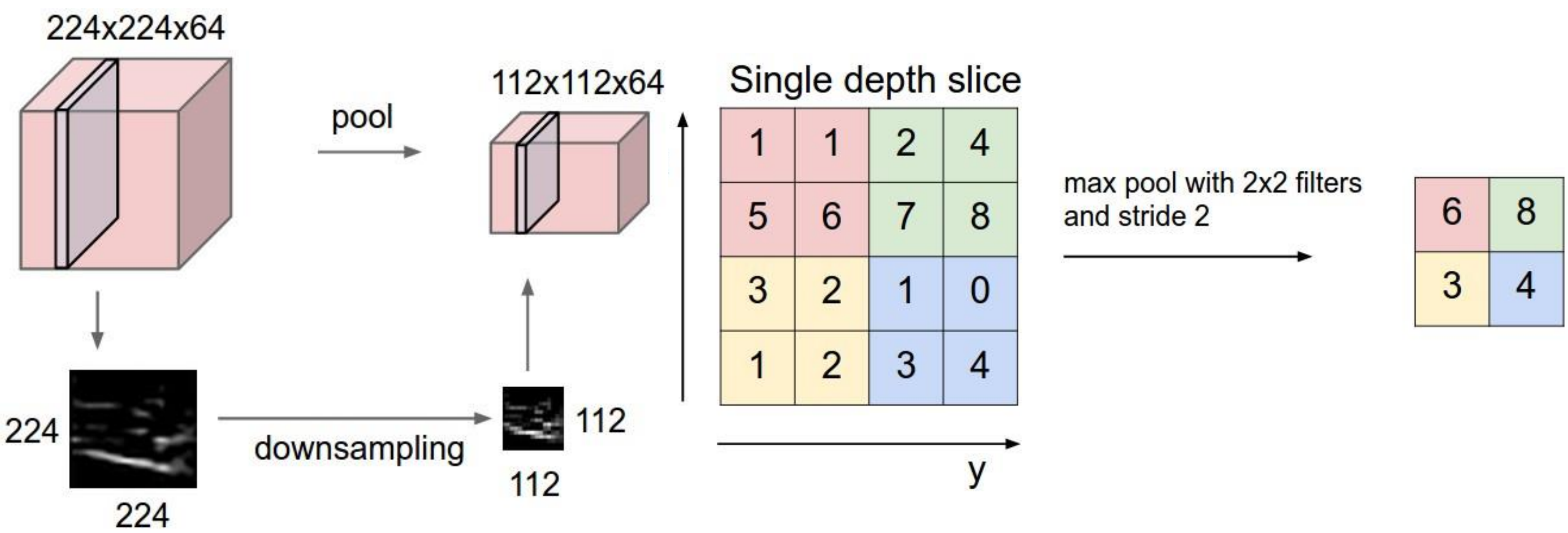
CNN block architecture



Source: towardsdatascience.com by Saha, S.

- Convolutional layer = compute feature
- Activation layer = apply non-linear transformation
- Pooling layer = reduce dimension

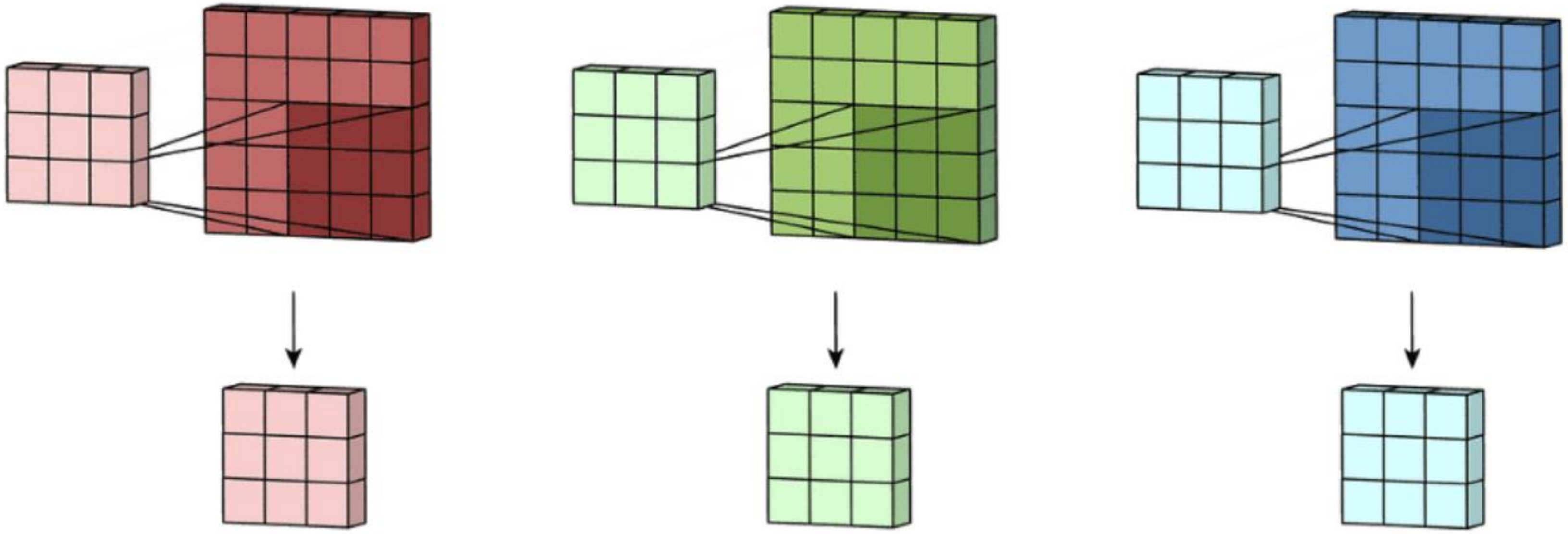
Pooling



Source: cs231n.github.io

- Like stride, pooling layer also reduce output size
  - Max pooling
  - Average pooling

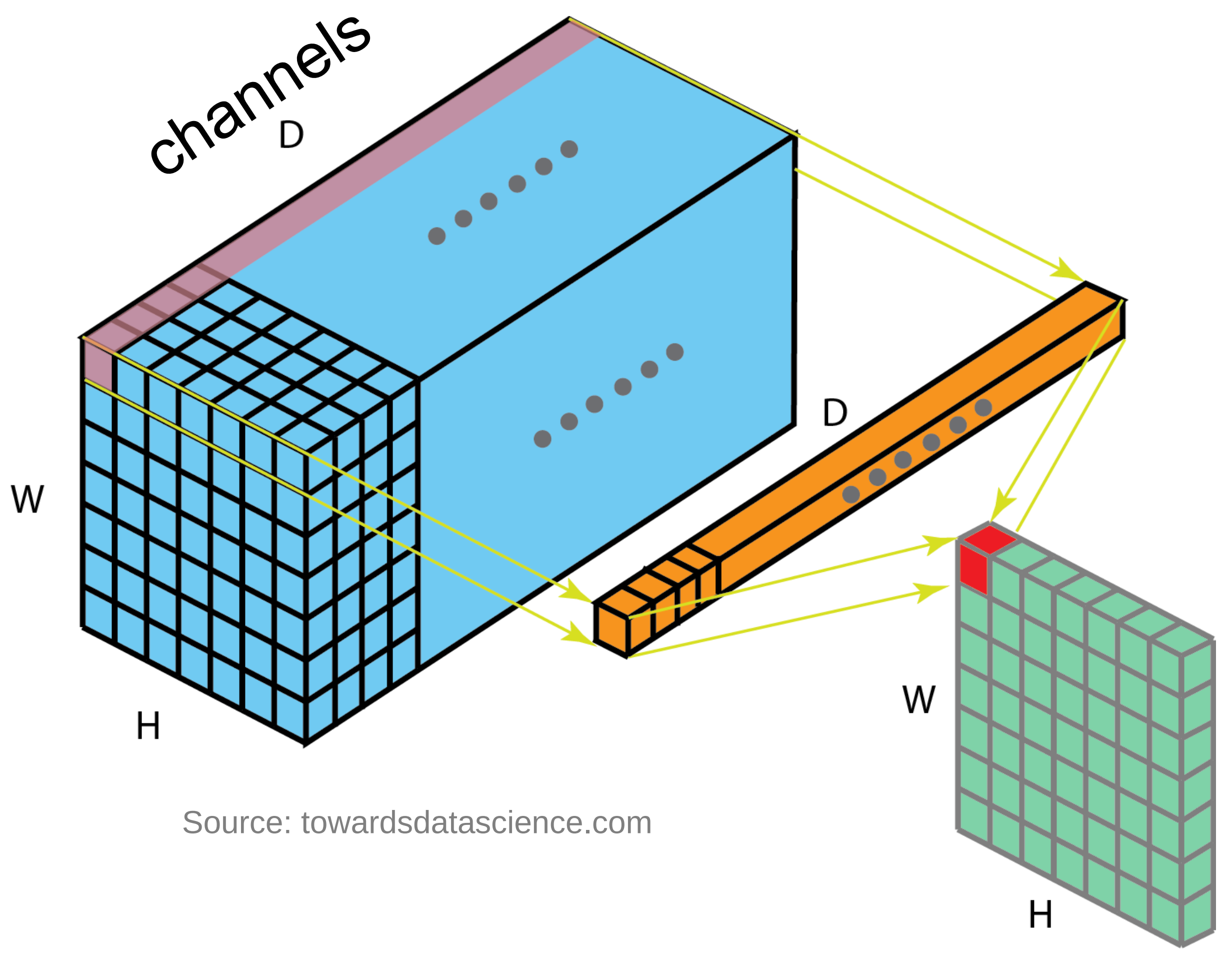
Channel



Source: towardsdatascience.com

- Image can have multiple channels (like RGB)
- Each filter is applied to each channel separately
- Output of each filter is also its own channel

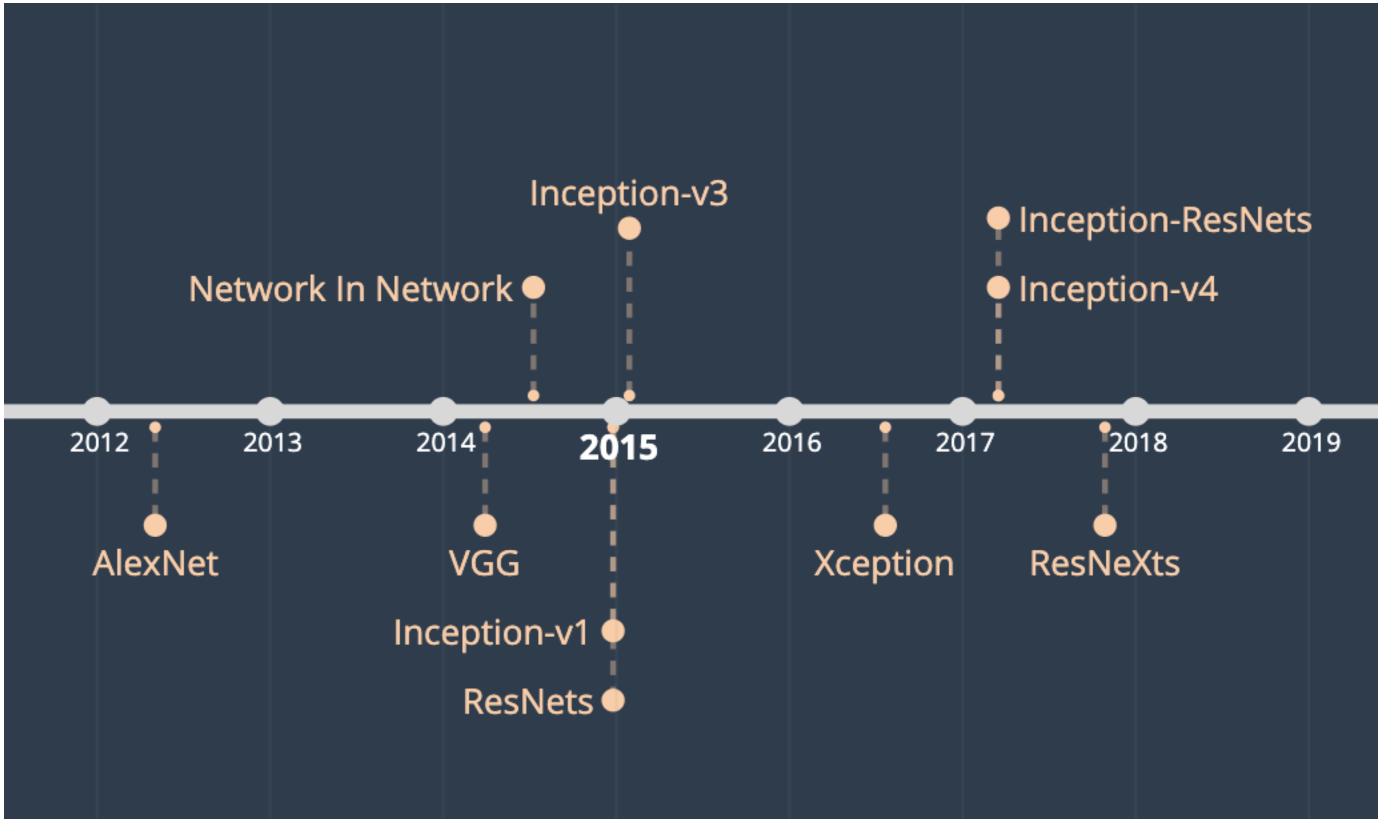
Merging channels



- Combine values across channels for each pixel
  - Like a D-to-1 fully connected layer on channels
  - Or a 1x1 convolution on image

Key CNN architectures

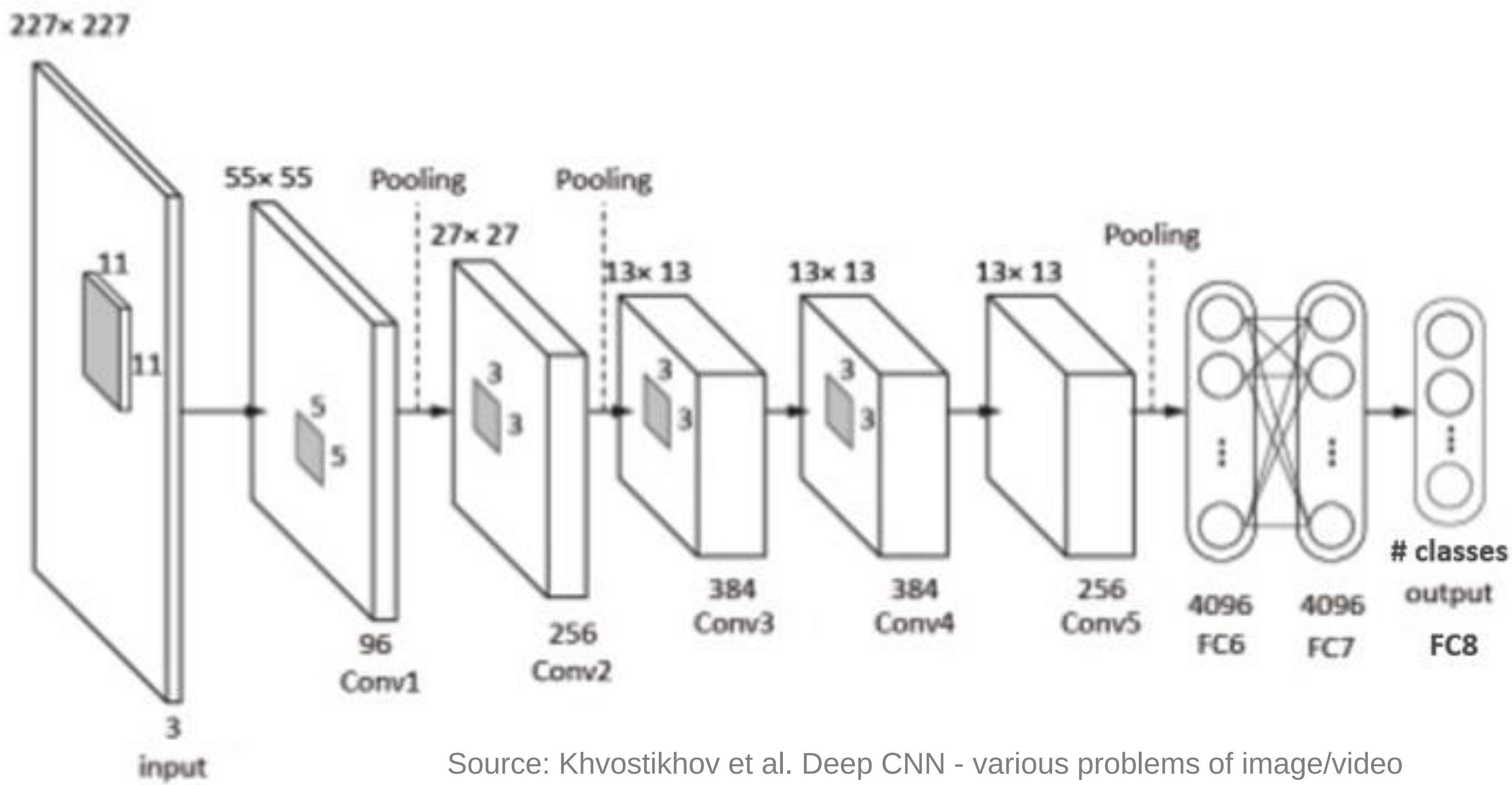
Timeline of CNN (for classification)



Source: towardsdatascience.com

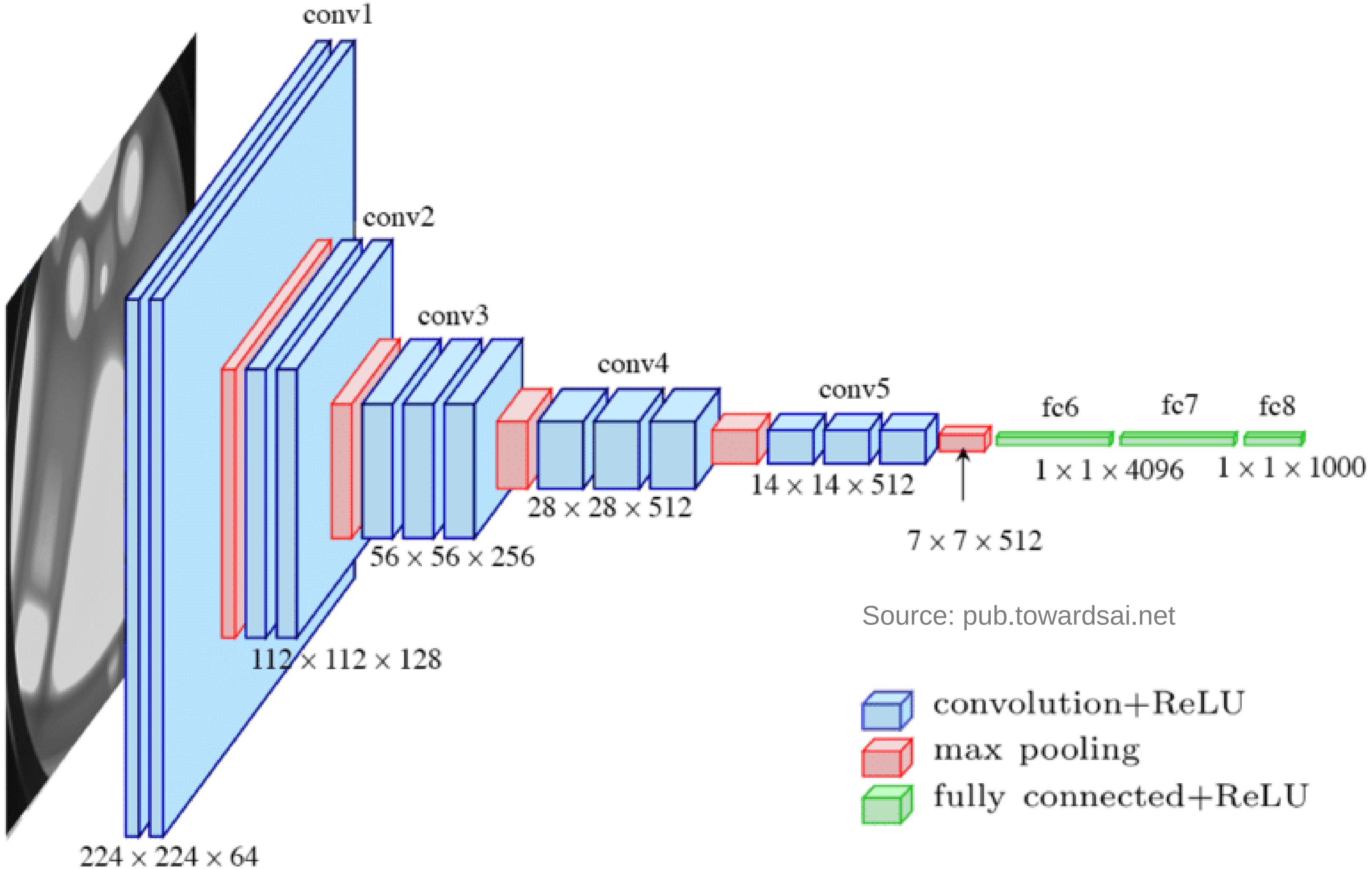
- AlexNet was the winner of 2012 ImageNet challenge
  - 5 convolution + 3 fully connected layers
  - Better than 2nd place by more than 10 percentage points

AlexNet



Source: Khvostikhov et al. Deep CNN - various problems of image/video classification Alzheimer disease studies (2018)

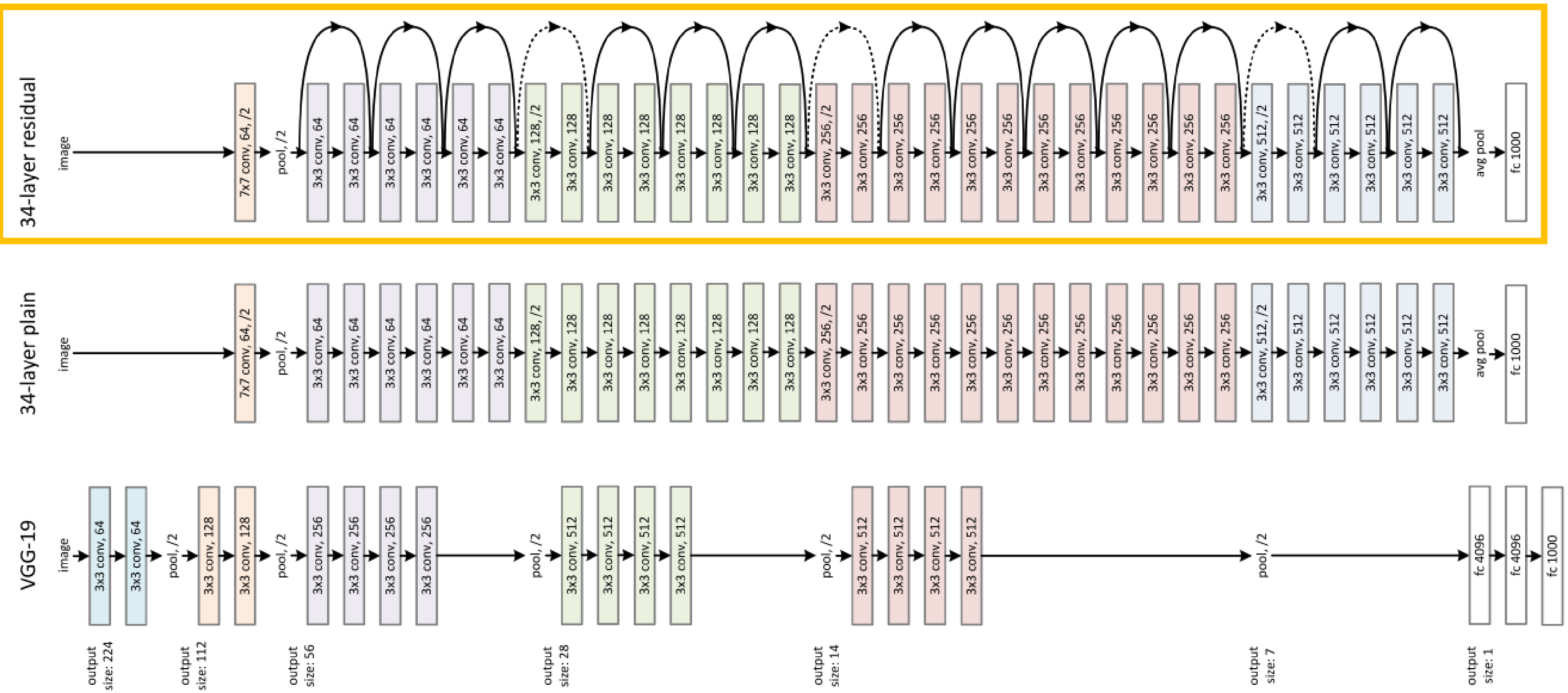
VGG-16



Source: pub.towardsai.net

- Decrease image size, increase number of channels

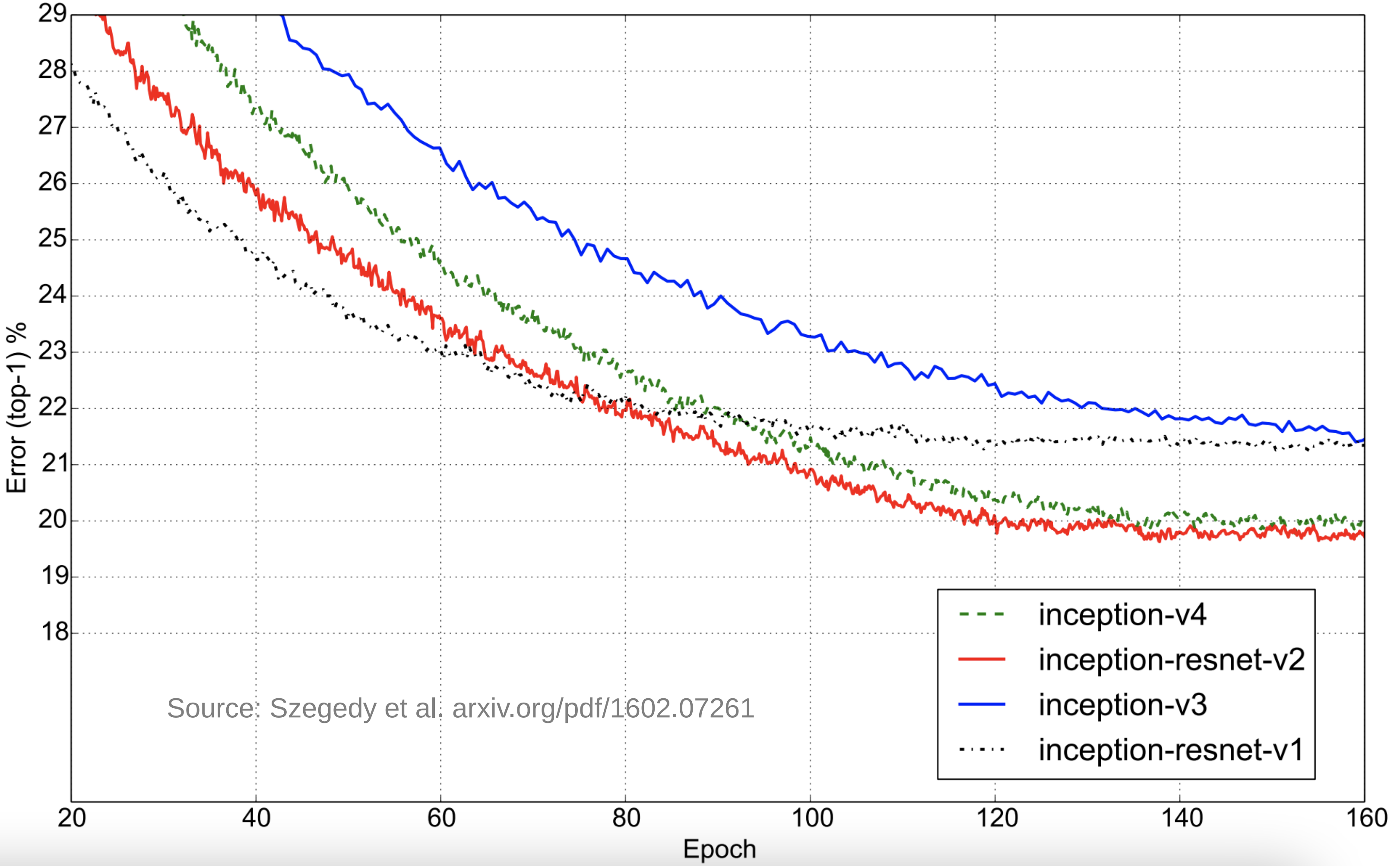
Residual Network (ResNet)



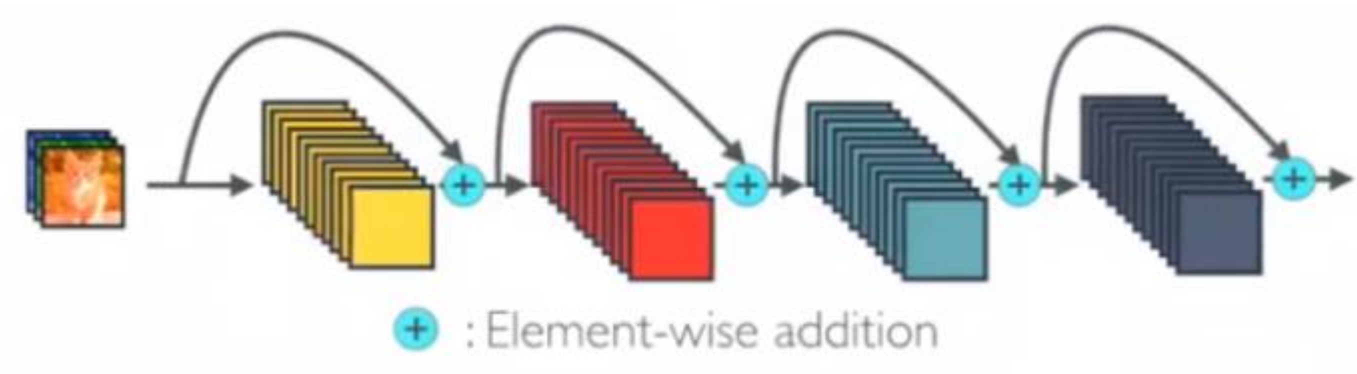
Source: medium.com

- Add bypasses over multiple convolutional layers
  - Reduce the number of multiplication terms in chain rule during backpropagation for early layers

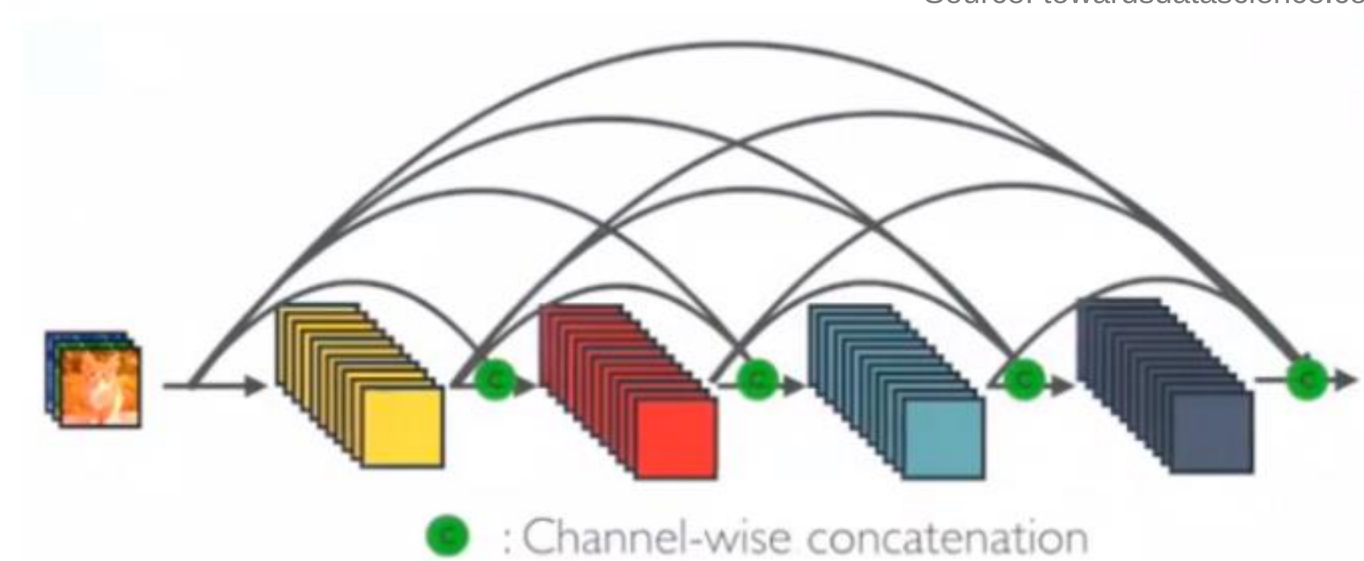
Residual connection improves training



DenseNet

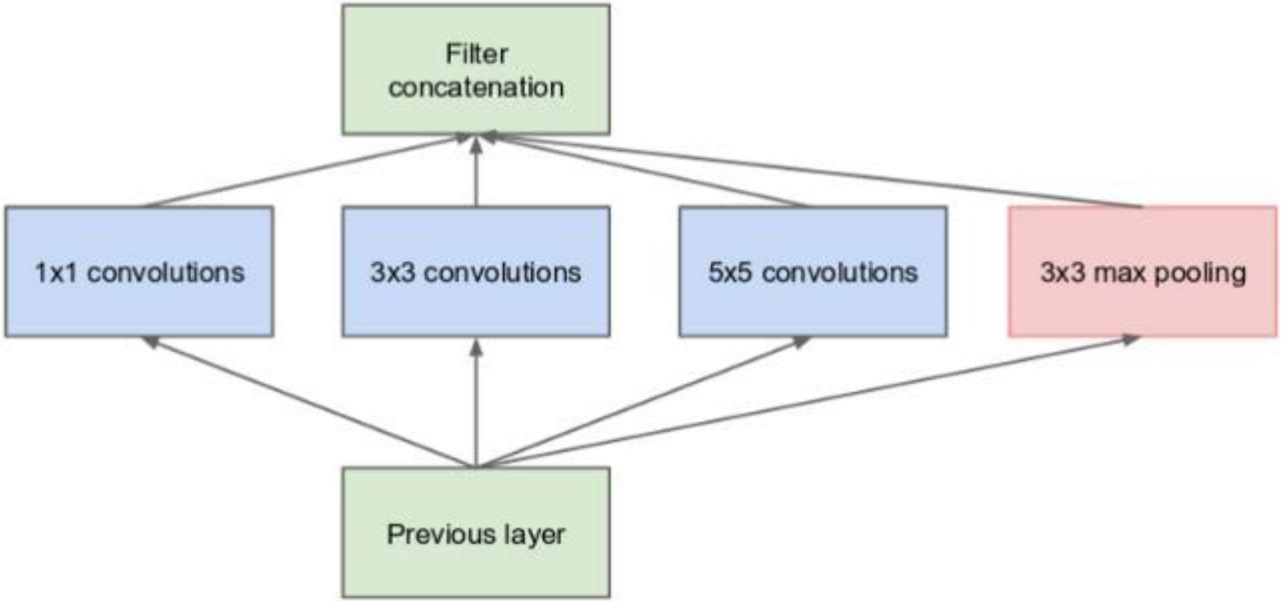


Source: towardsdatascience.com



- Feedforward bypasses

Inception = multi-resolution CNN



From left: A dog occupying most of the image, a dog occupying a part of it, and a dog occupying very little space (Images obtained from Unsplash).

CNN design in a nutshell

Module: tf.keras.applications

Modules

`densenet` module: DenseNet models for Keras.

`efficientnet` module: EfficientNet models for Keras.

`imagenet_utils` module: Utilities for ImageNet data preprocessing & prediction decoding.

`inception_resnet_v2` module: Inception-ResNet V2 model for Keras.

`inception_v3` module: Inception V3 model for Keras.

`mobilenet` module: MobileNet v1 models for Keras.

`mobilenet_v2` module: MobileNet v2 models for Keras.

`mobilenet_v3` module: MobileNet v3 models for Keras.

`nasnet` module: NASNet-A models for Keras.

`resnet` module: ResNet models for Keras.

Any question?

Pyception video, AnacondaCon 2018