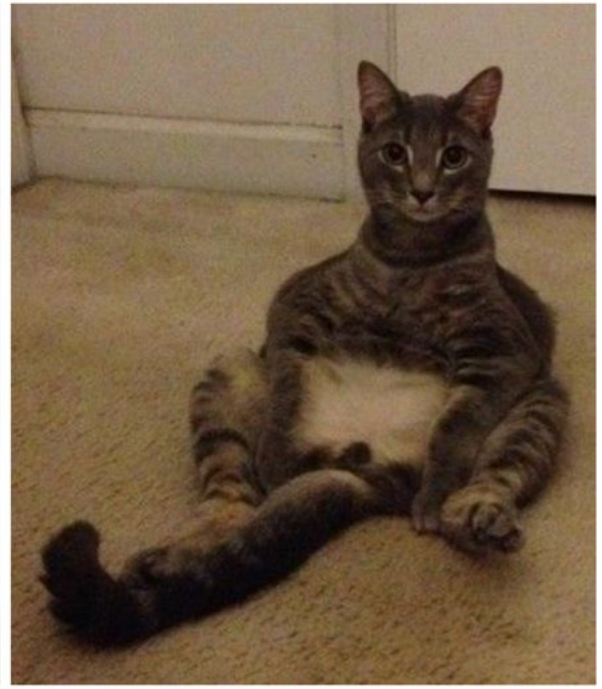
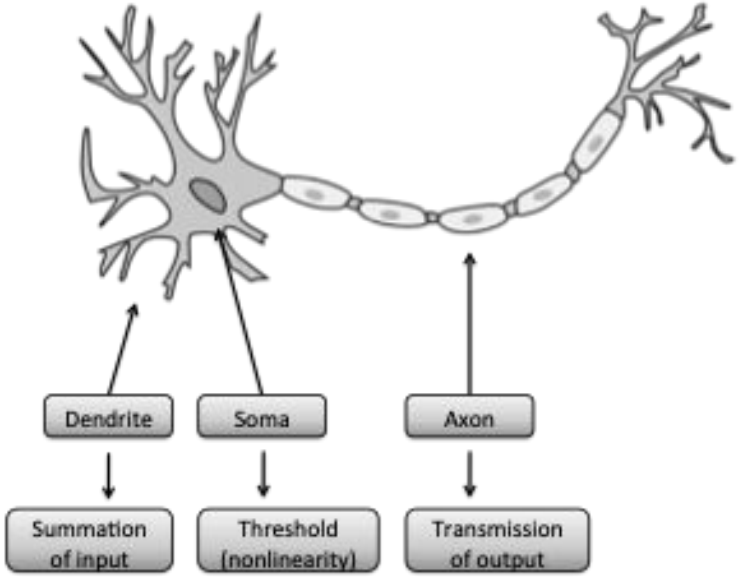
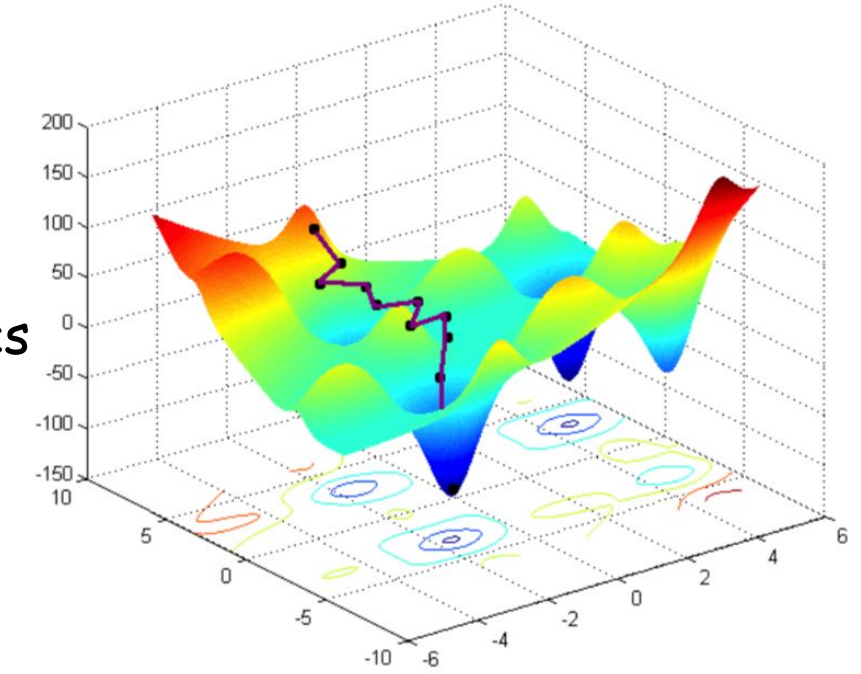
Deep Learning



By Phil Harrison
philip.harrison@farmbio.uu.se
Dept. of Pharmaceutical bioinformatics
Uppsala University





...with some slides adapted/inspired by material in lectures given by:

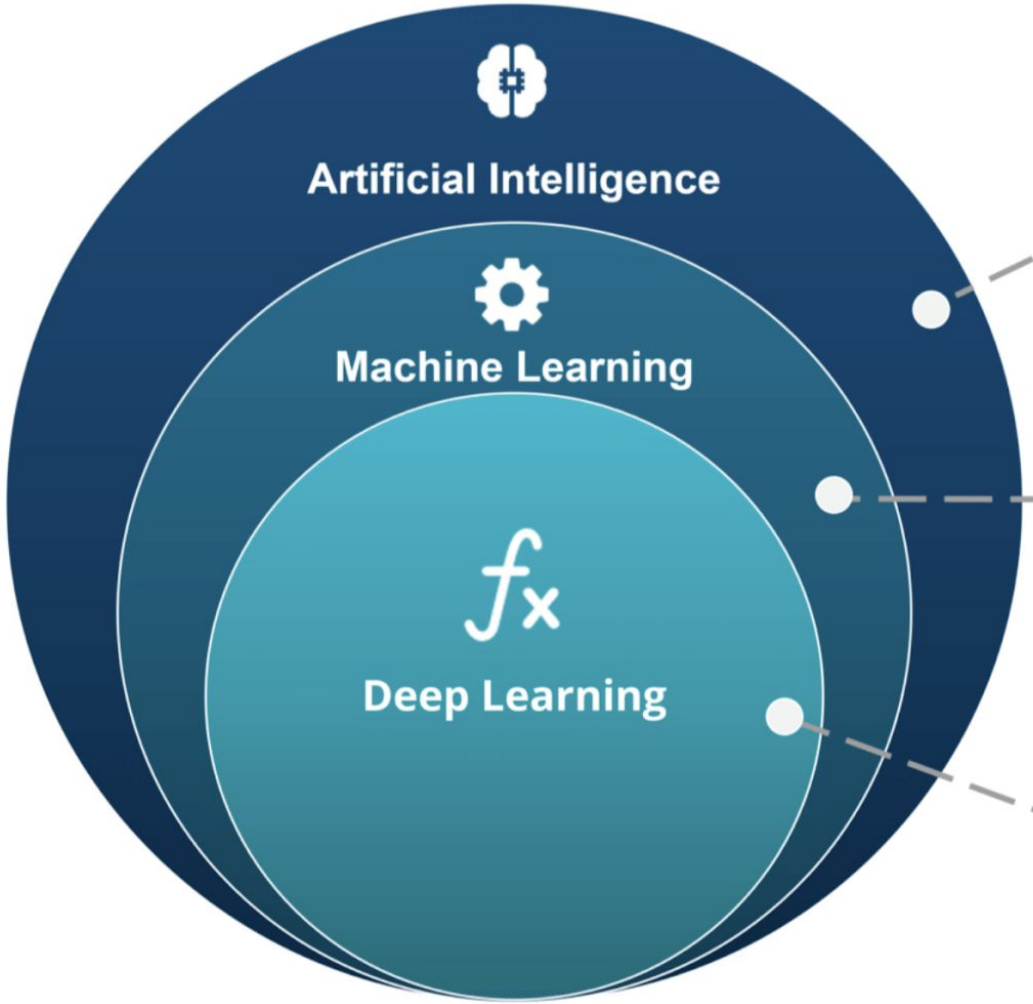
Anindya Gupta, Joakim Lindblad & Sajith Sadanandan

pharmb.io

Pharmaceutical Bioinformatics research group

Statistics and Prediction in the Pharmaceutical Sciences
20th May 2019

Deep learning and AI



ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

https://www.edureka.co/blog/ai-vs-machine-learning-vs-deep-learning/

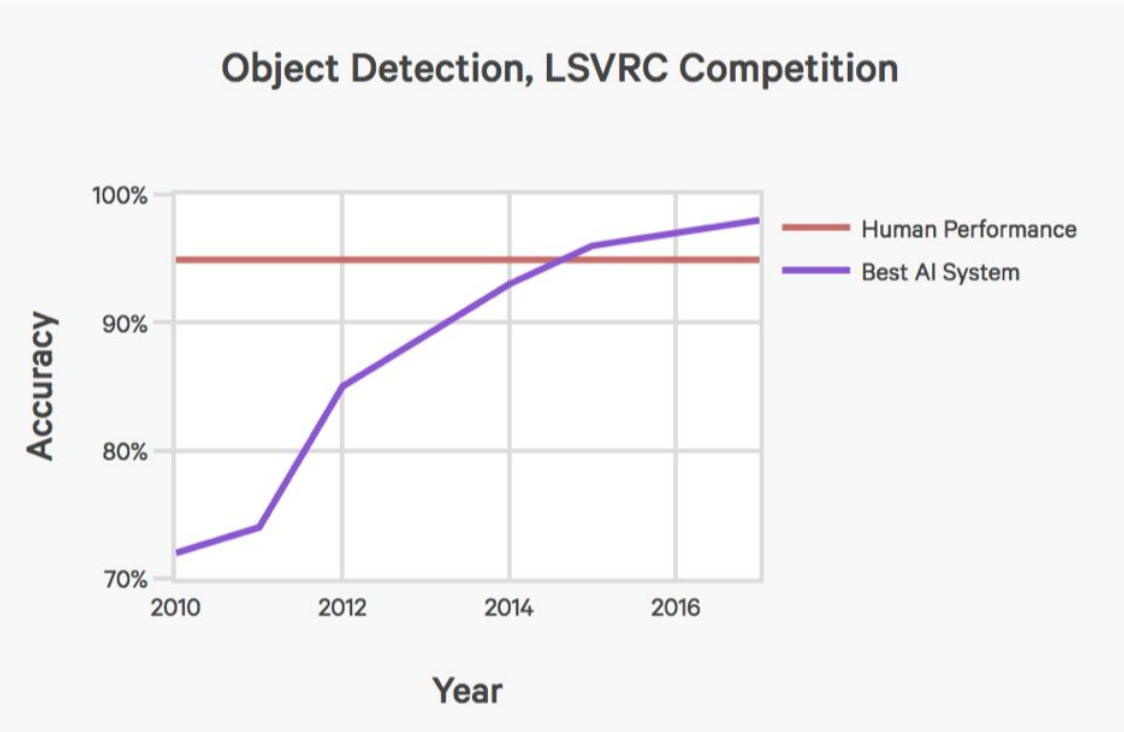
TECHNICAL PERFORMANCE

Vision

Object Detection

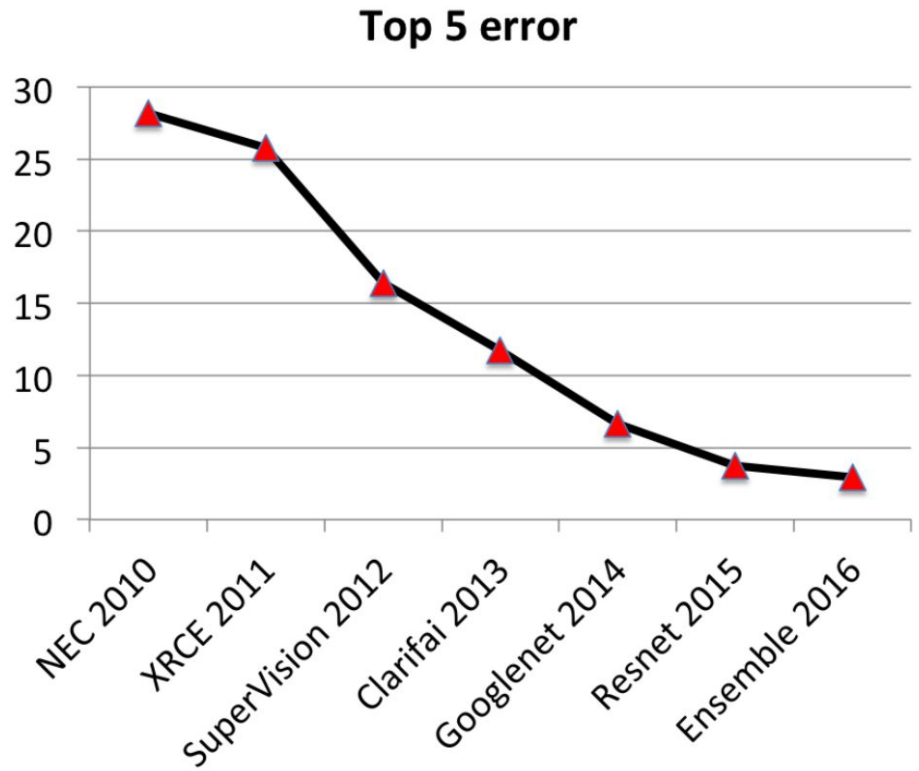
view more information in appendix A10

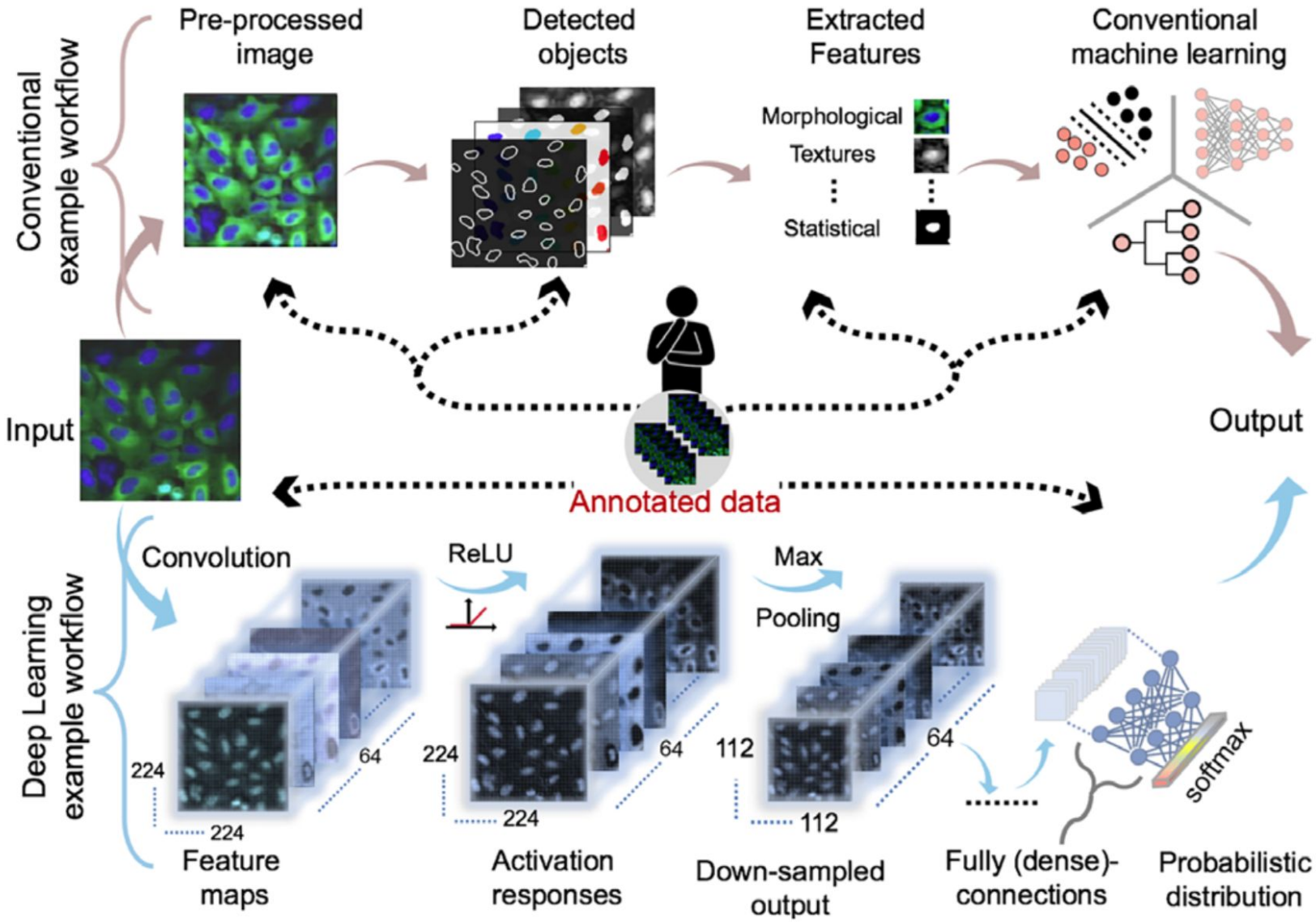
The performance of AI systems on the object detection task in the Large Scale Visual Recognition Challenge (LSVRC) Competition.



http://www.aiindex.org/2017-report.pdf

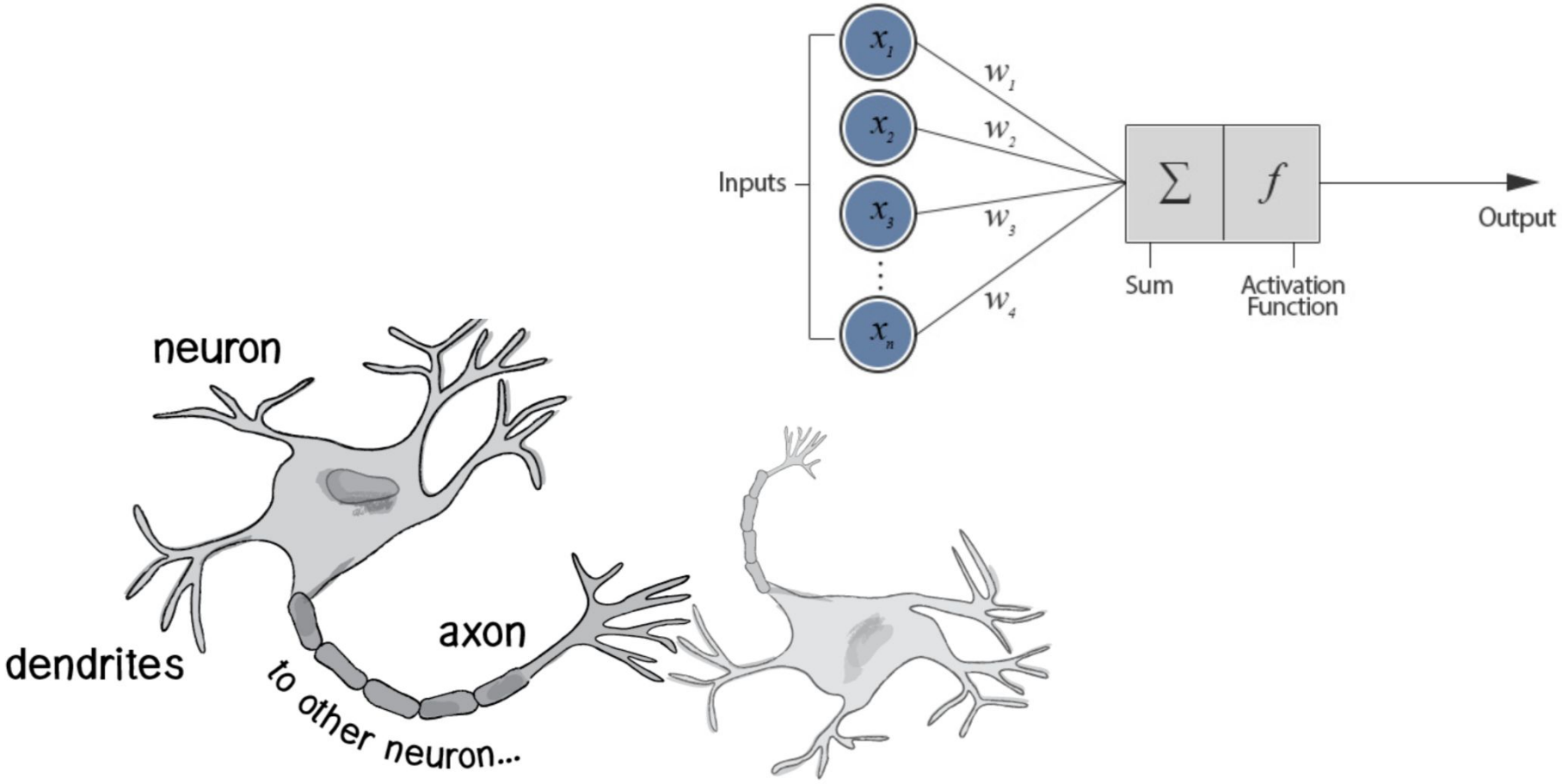
- subset of ImageNet dataset
- 1000 classes
- 1.2 million images
- From 2012 onwards won by CNNs



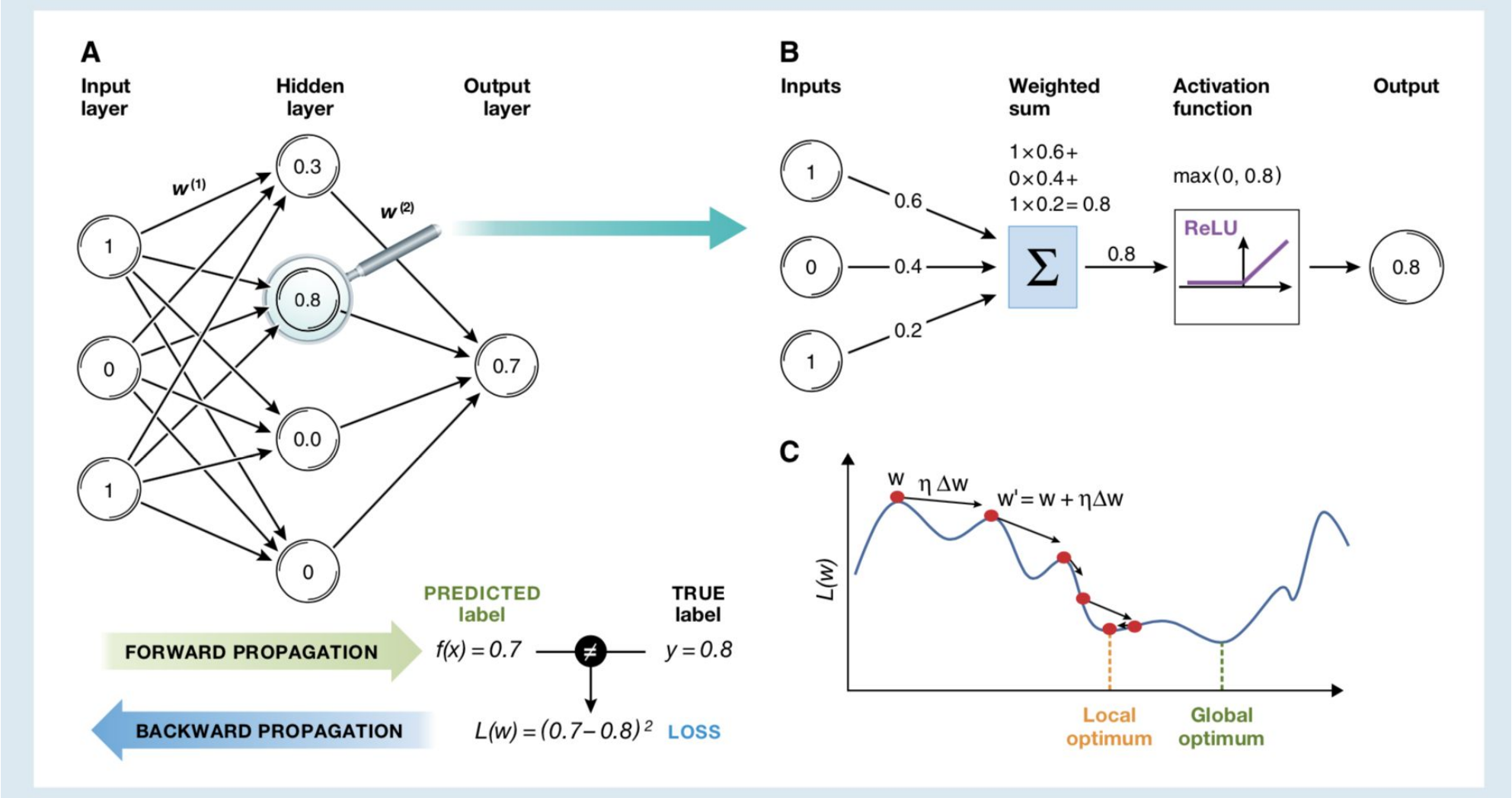


Overview of conventional versus deep learning workflows (from Gupta et al, 2019)

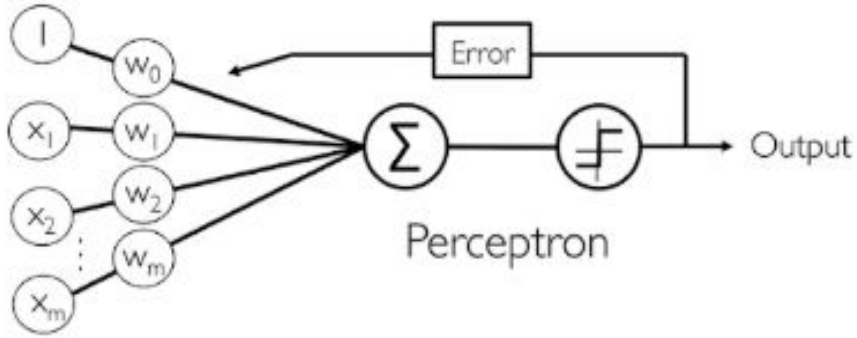
Biological connection

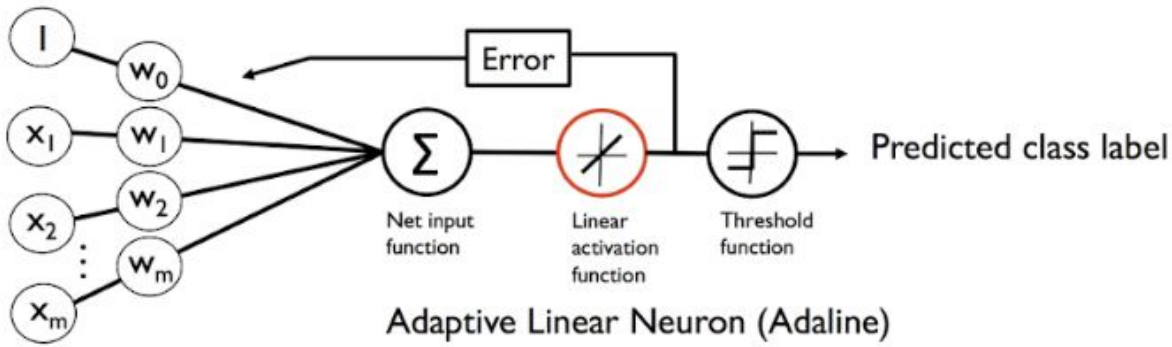


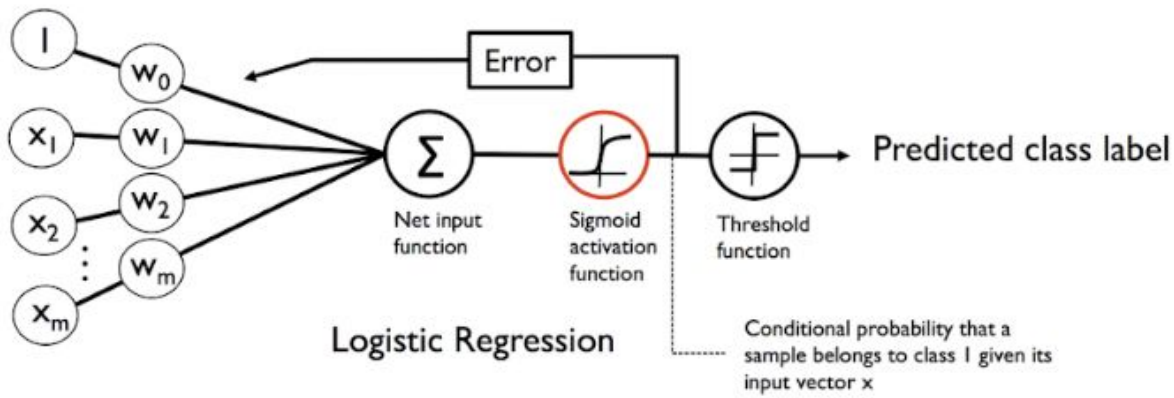
An introductory overview

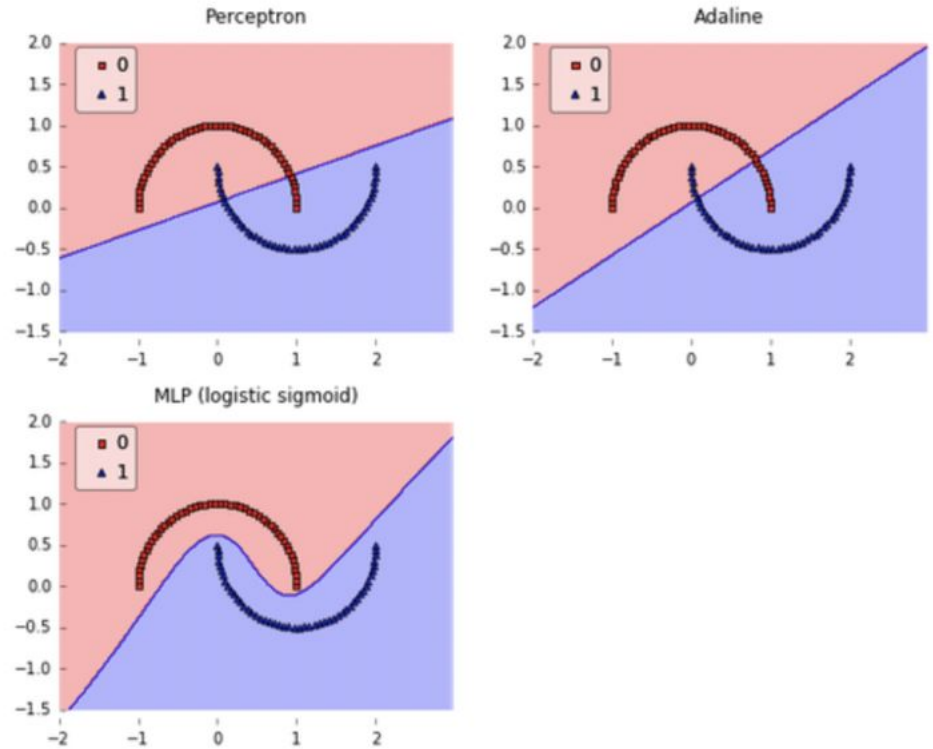


Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. Mol. Syst. Biol. *12*, 878.

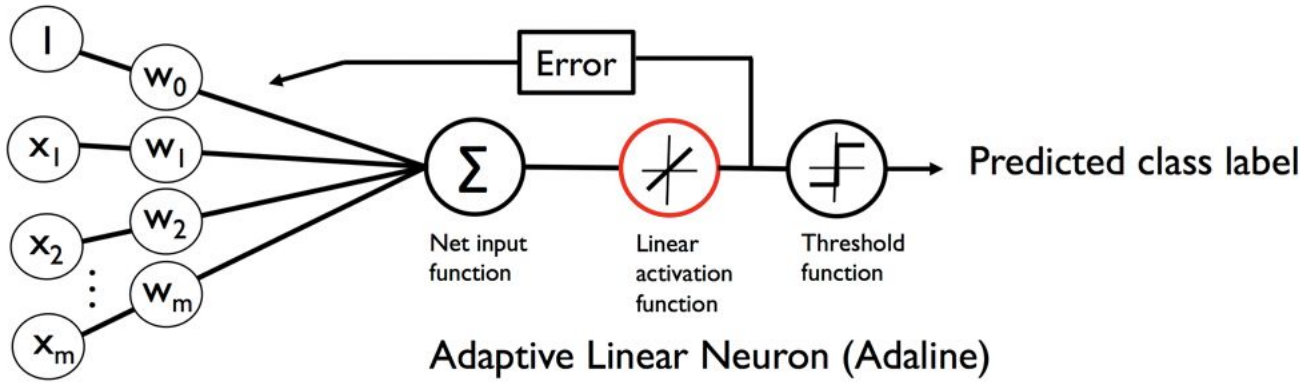




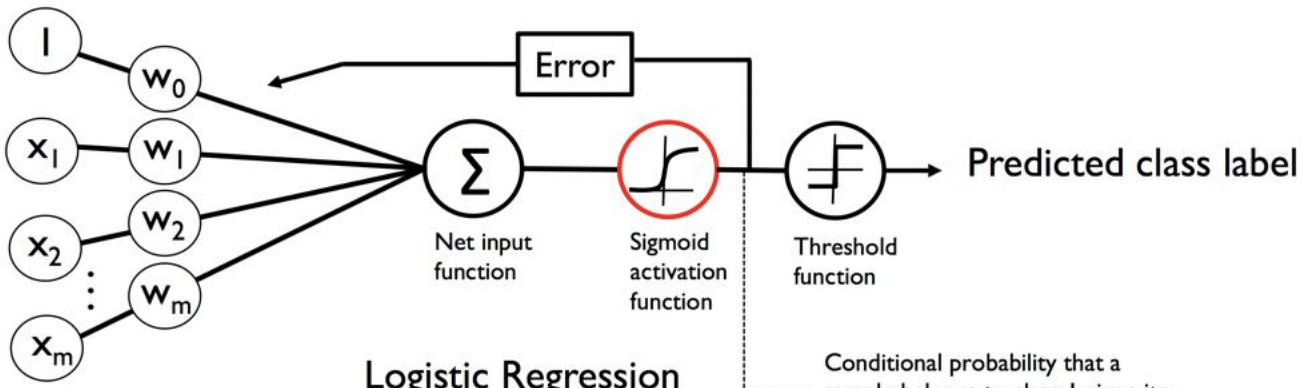




from Python Machine Learning (2nd Edition) by Ranschka & Mirjalili

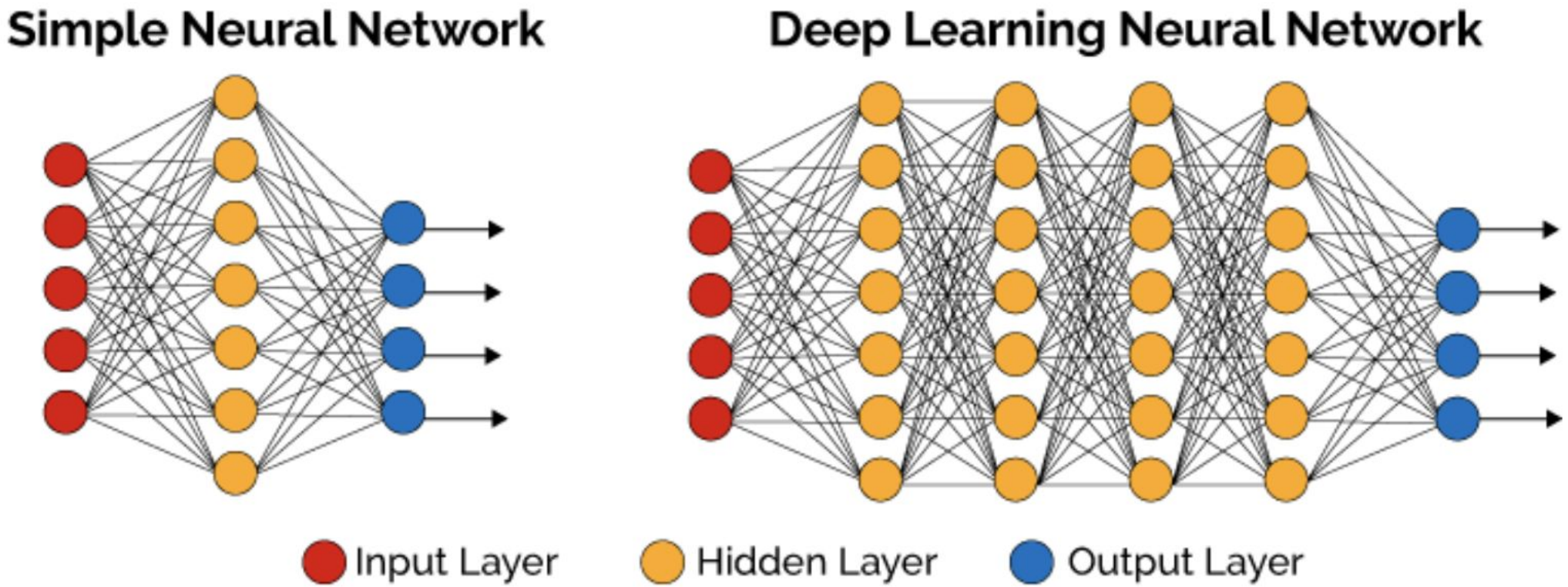


Adaptive Linear Neuron (Adaline)

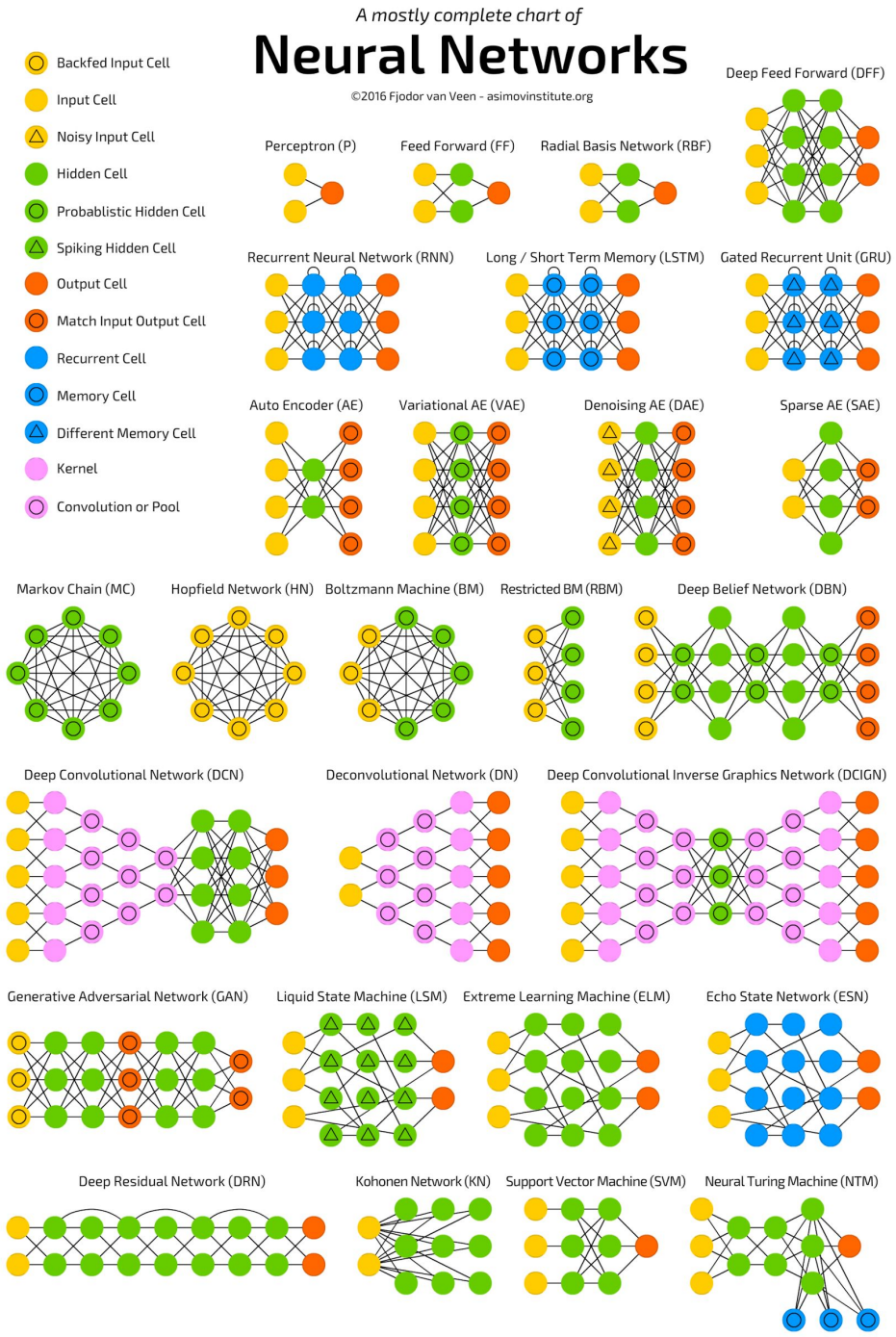


Logistic Regression

Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -½ \\ z + ½ & -½ \leq z \leq ½ \\ 1 & z \geq ½ \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	



https://www.kdnuggets.com/2017/12/deep-learning-made-easy-deep-cognition.html



A multitude of types

- Usually require input data and corresponding output labels for training
- Automatically learn the necessary features to label unseen data
- Choice of network depends on the problem at hand
- **Convolutional neural networks (CNNs)** are ideal for image data and will be the focus of this lecture and the lab tomorrow

https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Image classification

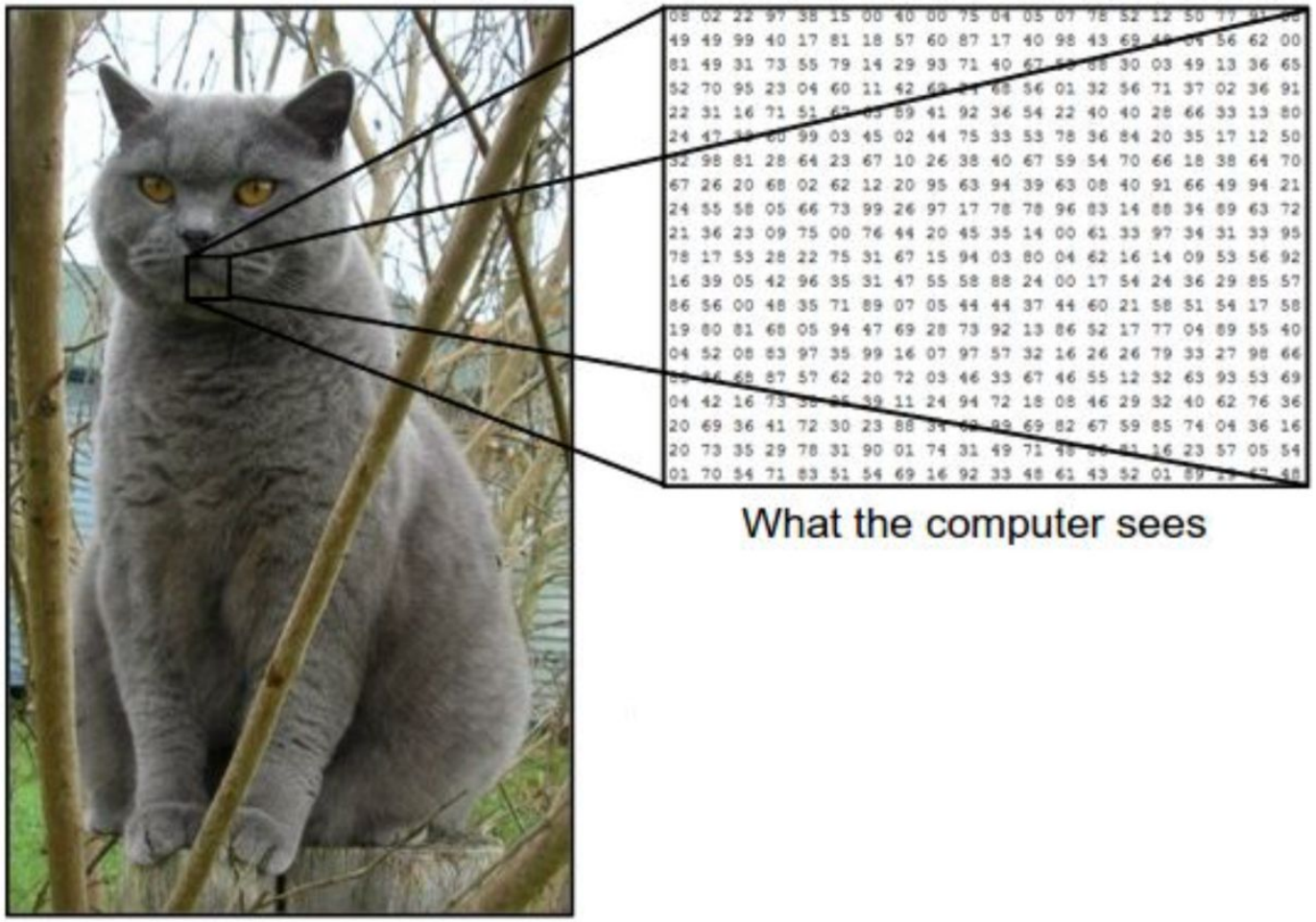
- Is a hard problem!
- Switching to Stanford slides for some feline friends to show us why!

Image Classification: a core task in Computer Vision

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

The problem:
semantic gap

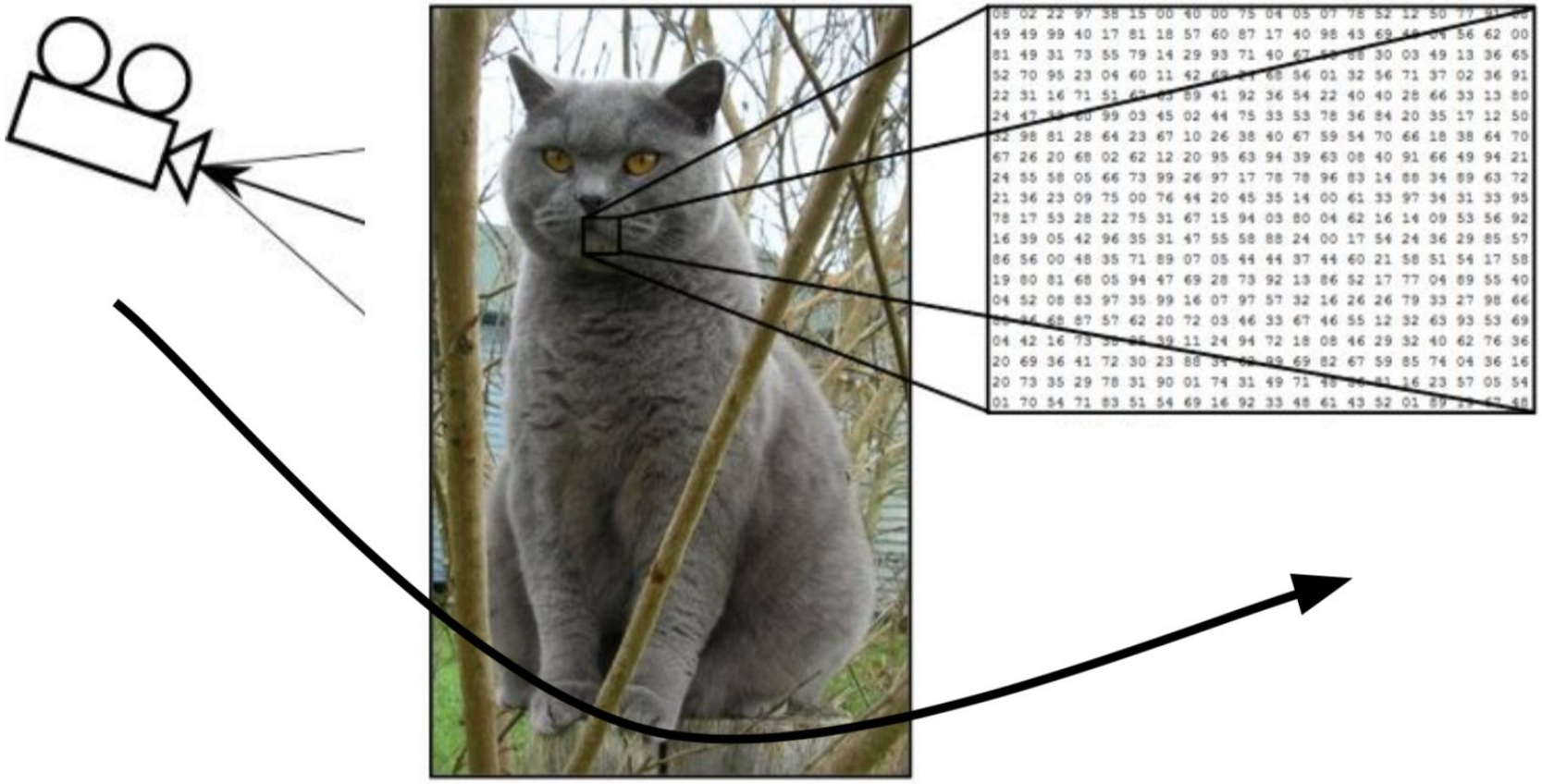


Images are represented as 3D arrays of numbers, with integers between [0, 255].

E.g.
300 x 100 x 3

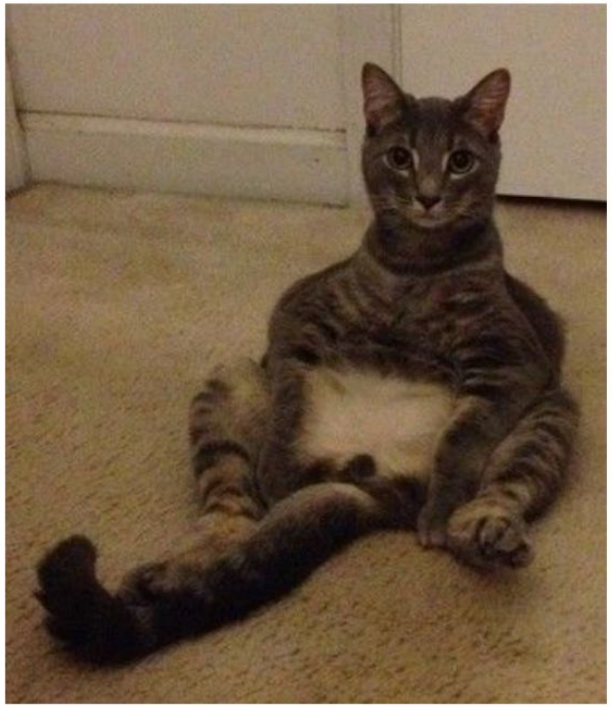
(3 for 3 color channels RGB)

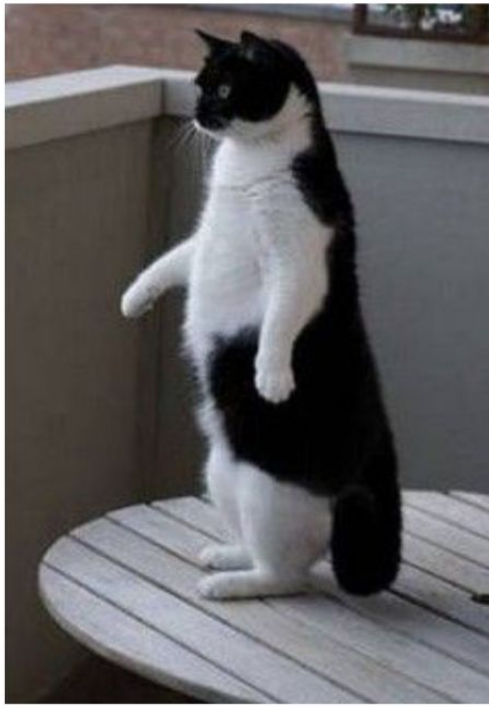
Challenges: Viewpoint Variation



Challenges: Illumination

Challenges: Deformation



Challenges: Occlusion

Challenges: Background clutter

Challenges: Intraclass variation

An image classifier

```
def predict(image):
    # ????
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm for recognizing a cat, or other classes.

Data-driven approach:

1. Collect a dataset of images and labels
2. Use Machine Learning to train an image classifier
3. Evaluate the classifier on a withheld set of test images

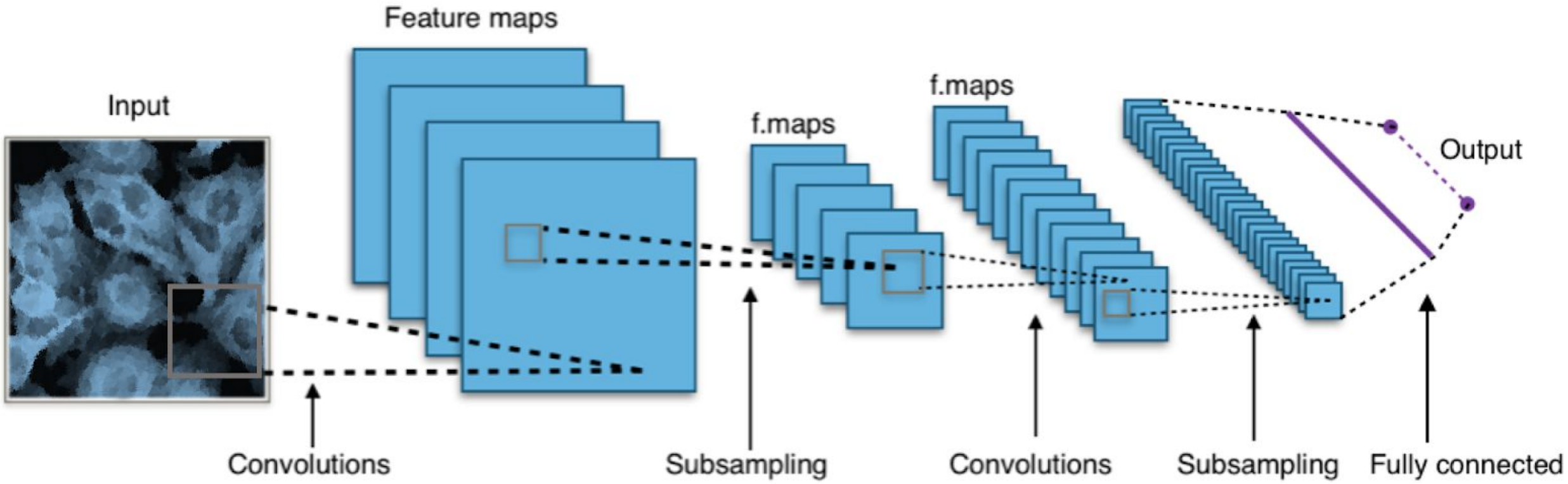
```
def train(train_images, train_labels):
  # build a model for images -> labels...
  return model

def predict(model, test_images):
  # predict test_labels using the model...
  return test_labels
```

Example training set

Convolutional Neural Networks (CNNs)

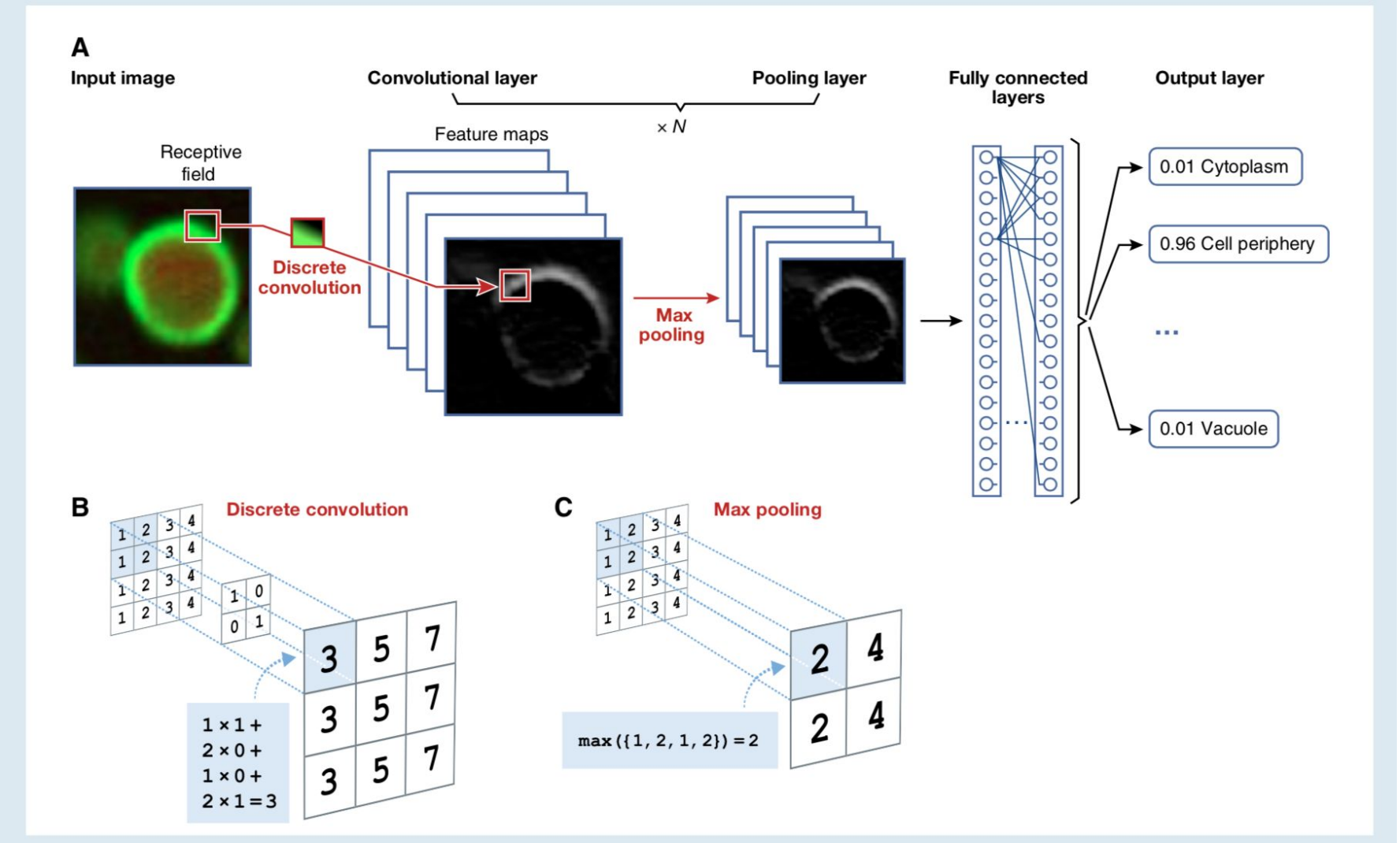
- Employ convolution operations
- Only local connectivity
- Spatial relationship is preserved
- Parameter sharing
- Translations equivariant
- Widely used in image analysis

Illustration of a typical CNN



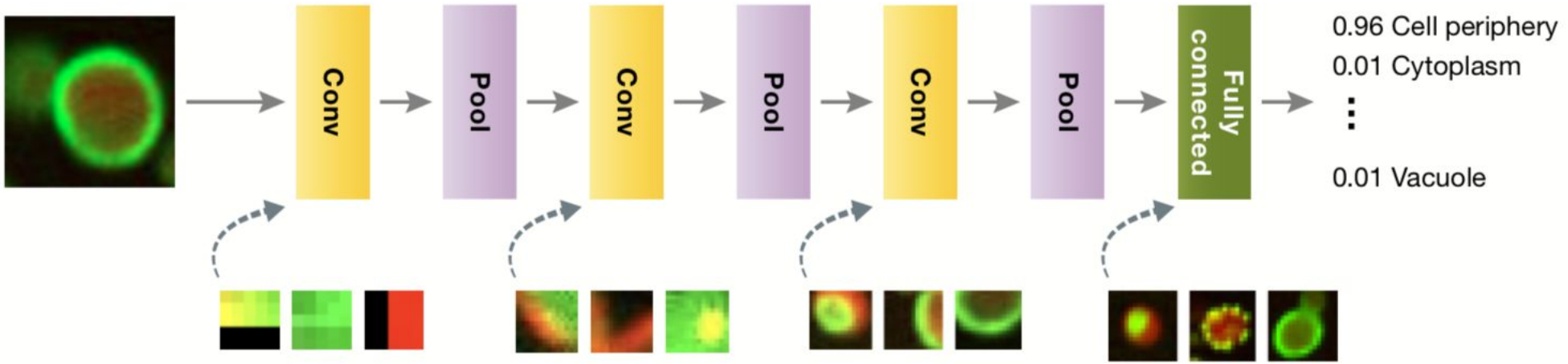
With today's computers much deeper networks are applied with great predictive power on image classification tasks. Figure modified (by Alexander Kensert) from Wikimedia Commons File:Typical cnn.png

An introduction to convolution and pooling



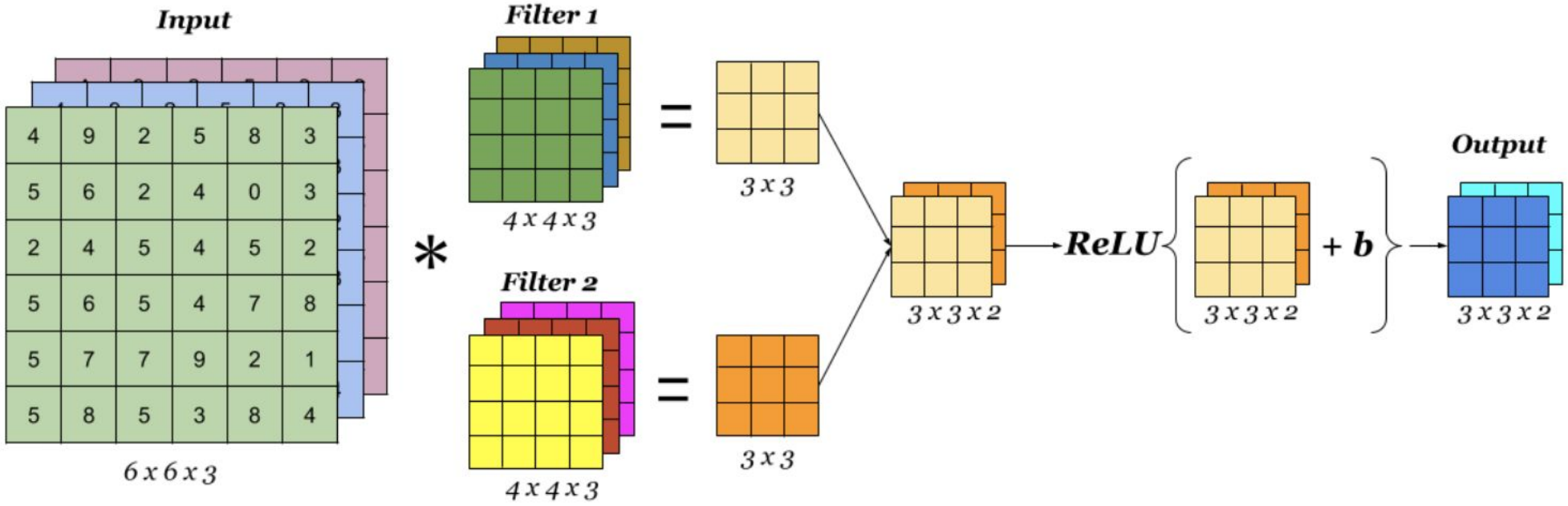
Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. Mol. Syst. Biol. *12*, 878.

What does it learn?



Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). Deep learning for computational biology. Mol. Syst. Biol. *12*, 878.

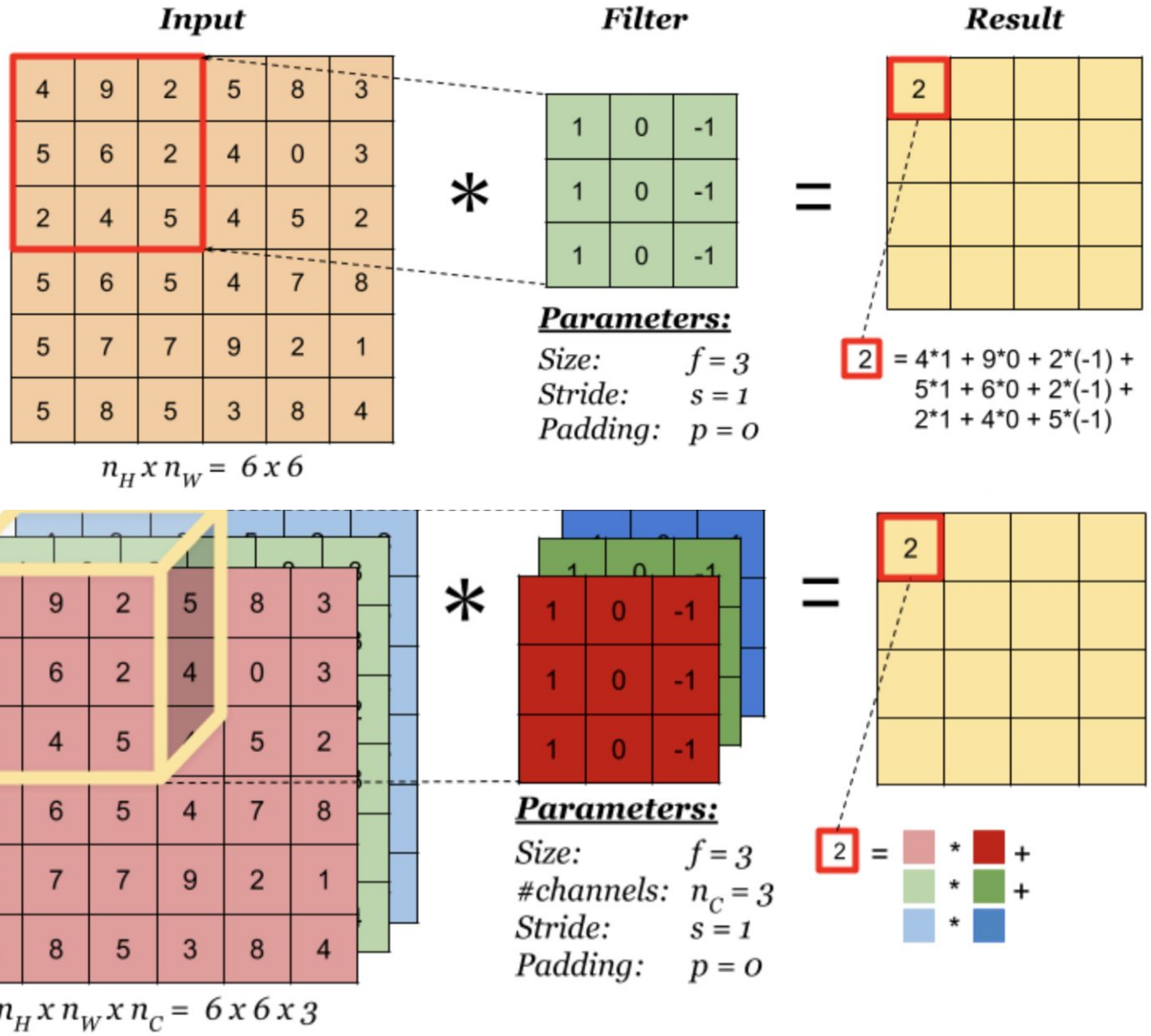
One convolutional layer



https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Convolution operation

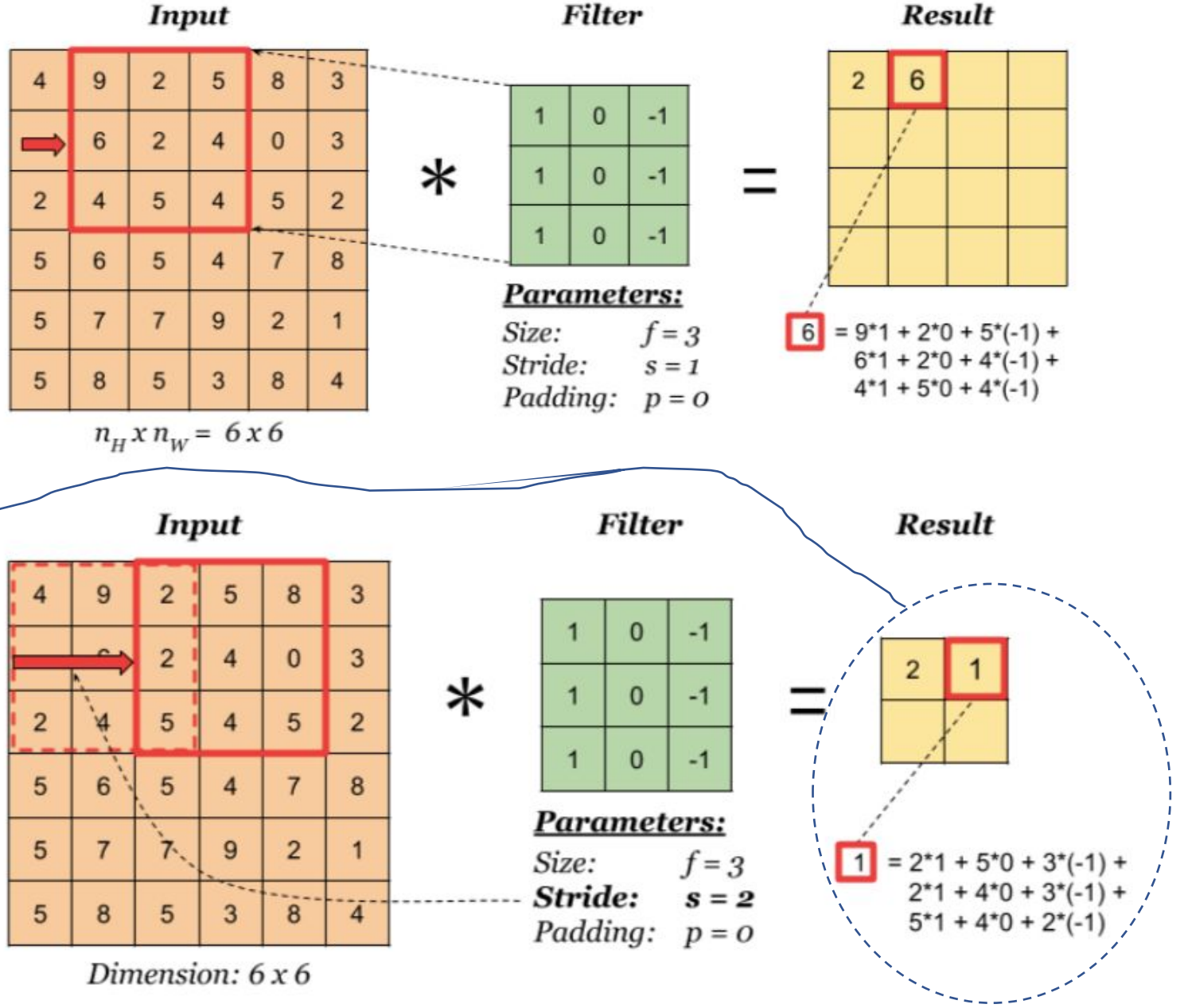
- Filters slide over the input to perform element wise multiplication and addition to generate feature maps
- Filter values corresponds to the parameters (or weights)
- Hyper parameters:
  - number of filters
  - stride
  - padding



https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Stride

- Stride governs how many cells the filter is moved across the input to calculate the next cell in the result
- Note the mistake made in this figure! It should have 8, 0 and 5 multiplied by (-1), not 3, 3, and 2, and the result should be -4



https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Another borrowed slide

Sobel filter

Approximates the first derivatives: $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$

1	0	-1
2	0	-2
1	0	-1

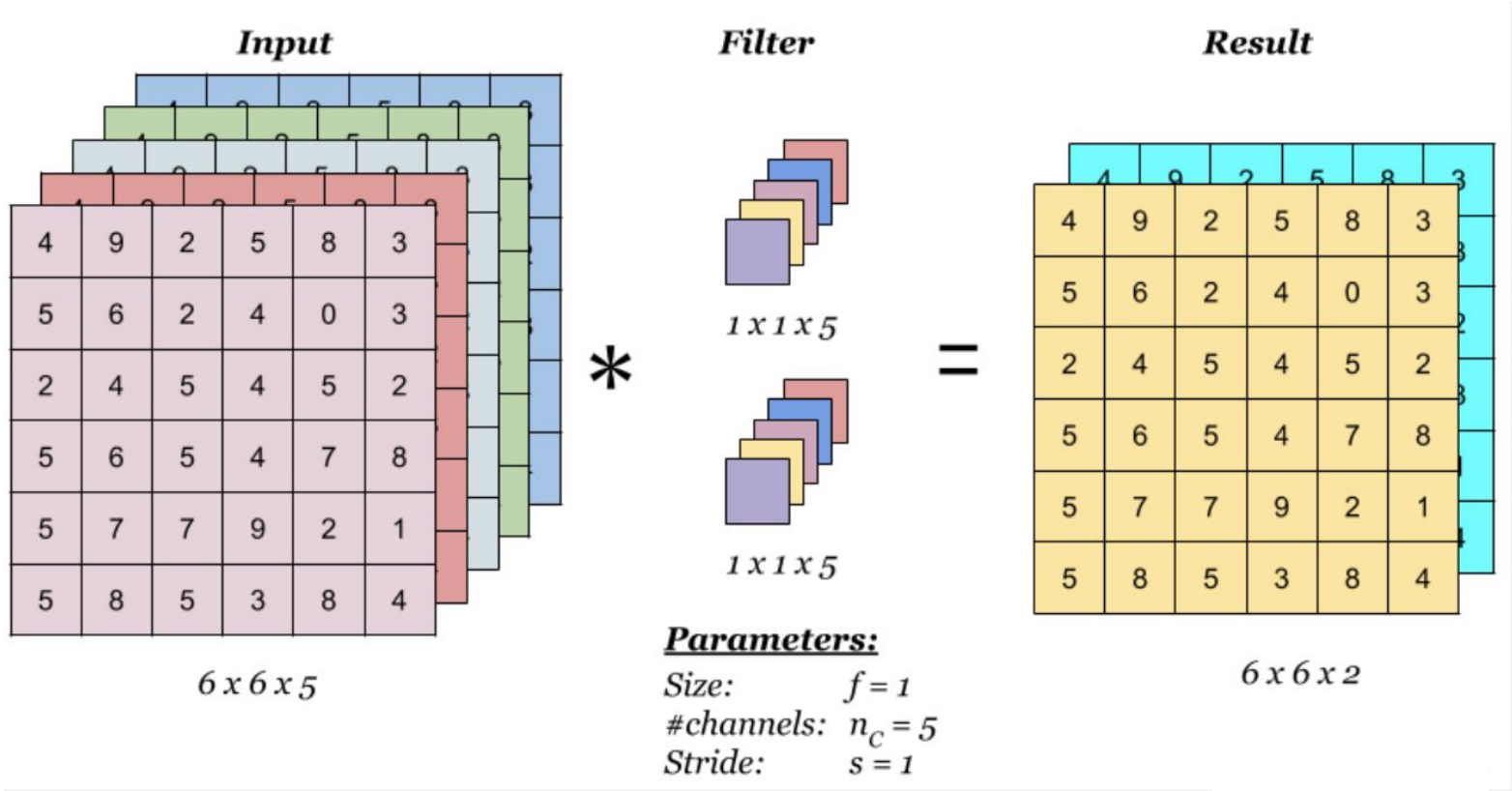
S_x

1	2	1
0	0	0
-1	-2	-1

S_y

1 x 1 convolutional filter

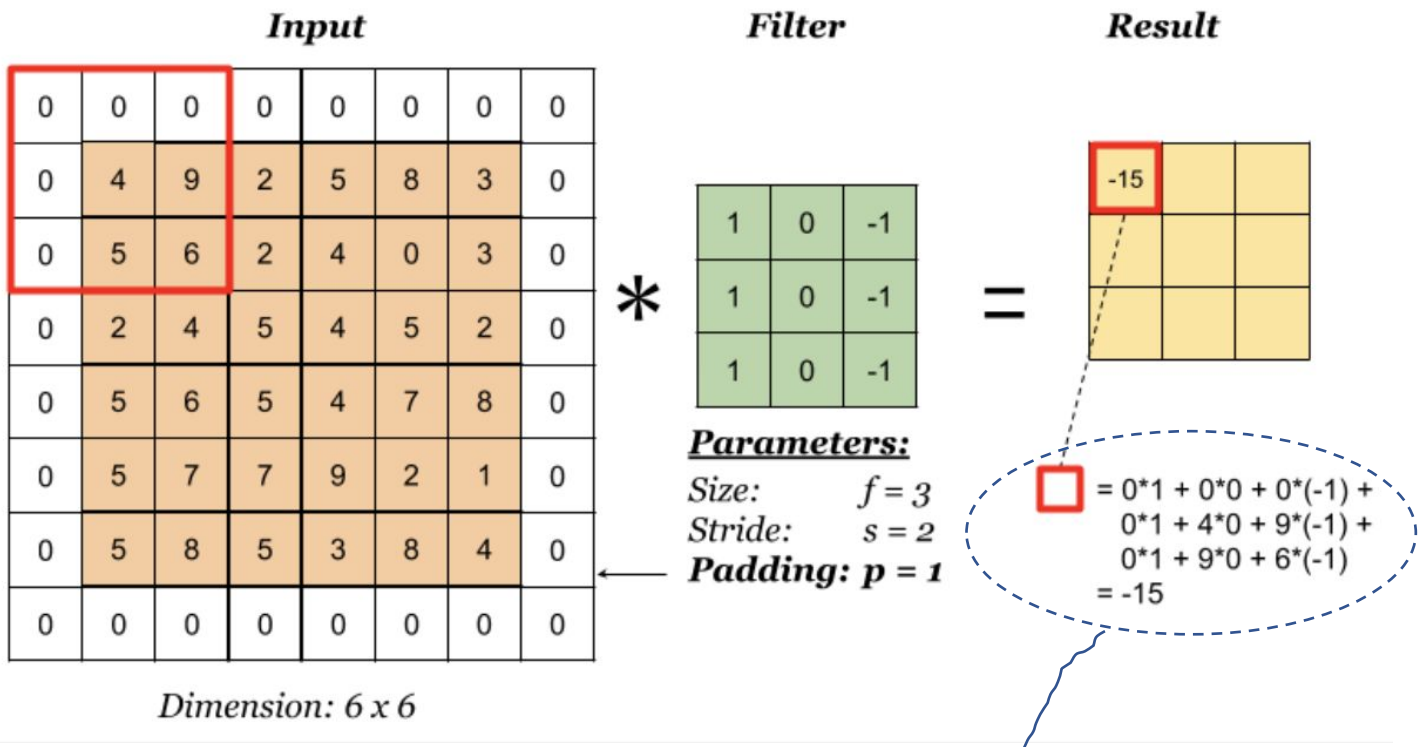
The effect of a 1 x 1 convolutional filter is to flatten or "merge" channels together, which can save computations later in the network



https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Padding

- Padding helps to keep the information at the borders of an image
- Without padding very few values at the next layer would be affected by pixels at the edges of an image

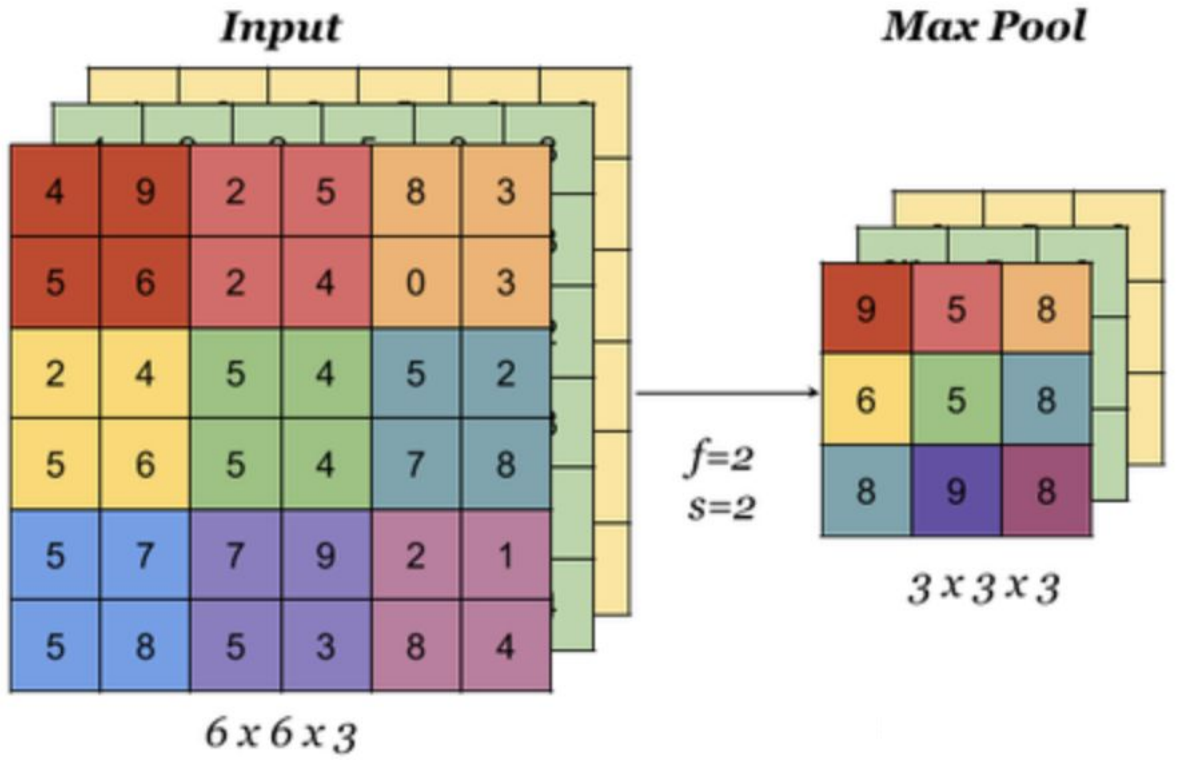


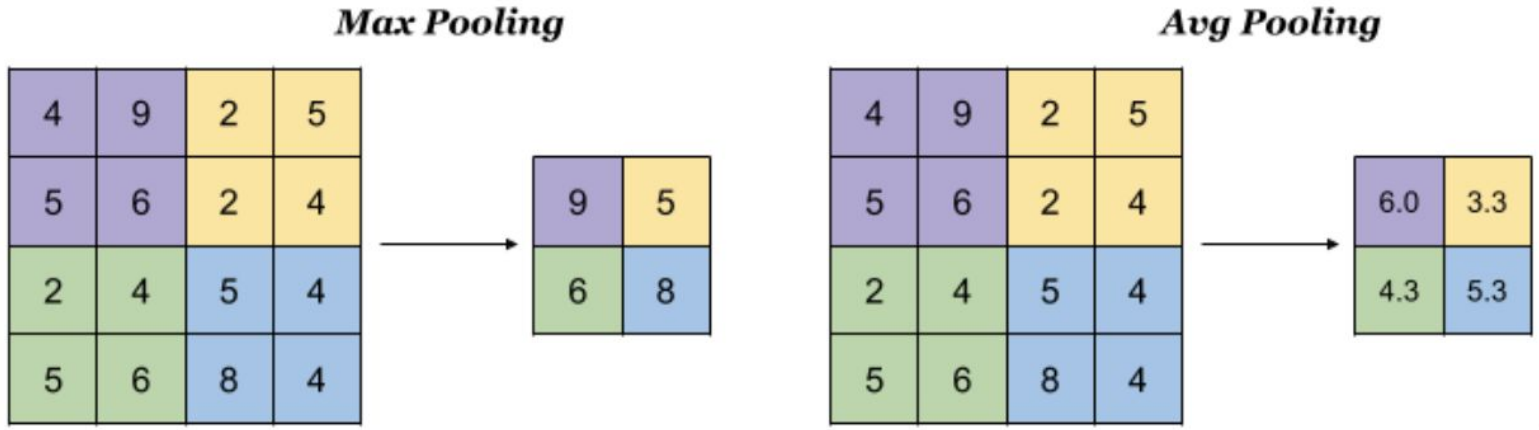
Another mistake!! Can anybody spot it?

https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Pooling layers

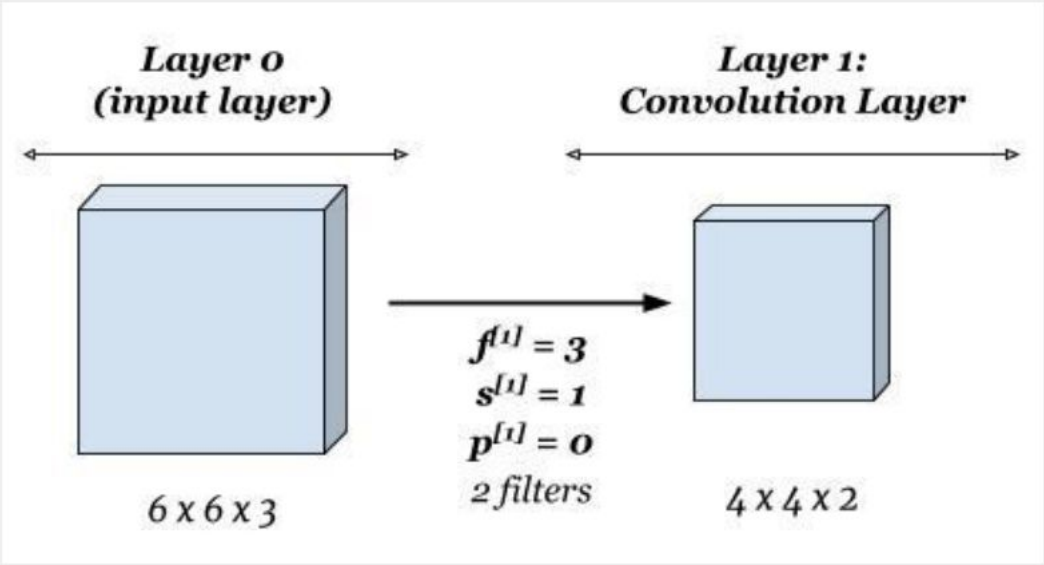
- Pooling layers reduce the size of the representations
- Reduce the number of trainable parameters
- Give translation-invariance
- Control overfitting
-
- Hyper-parameters:
  - size (f)
  - stride (s)
  - type (max or avg)





https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

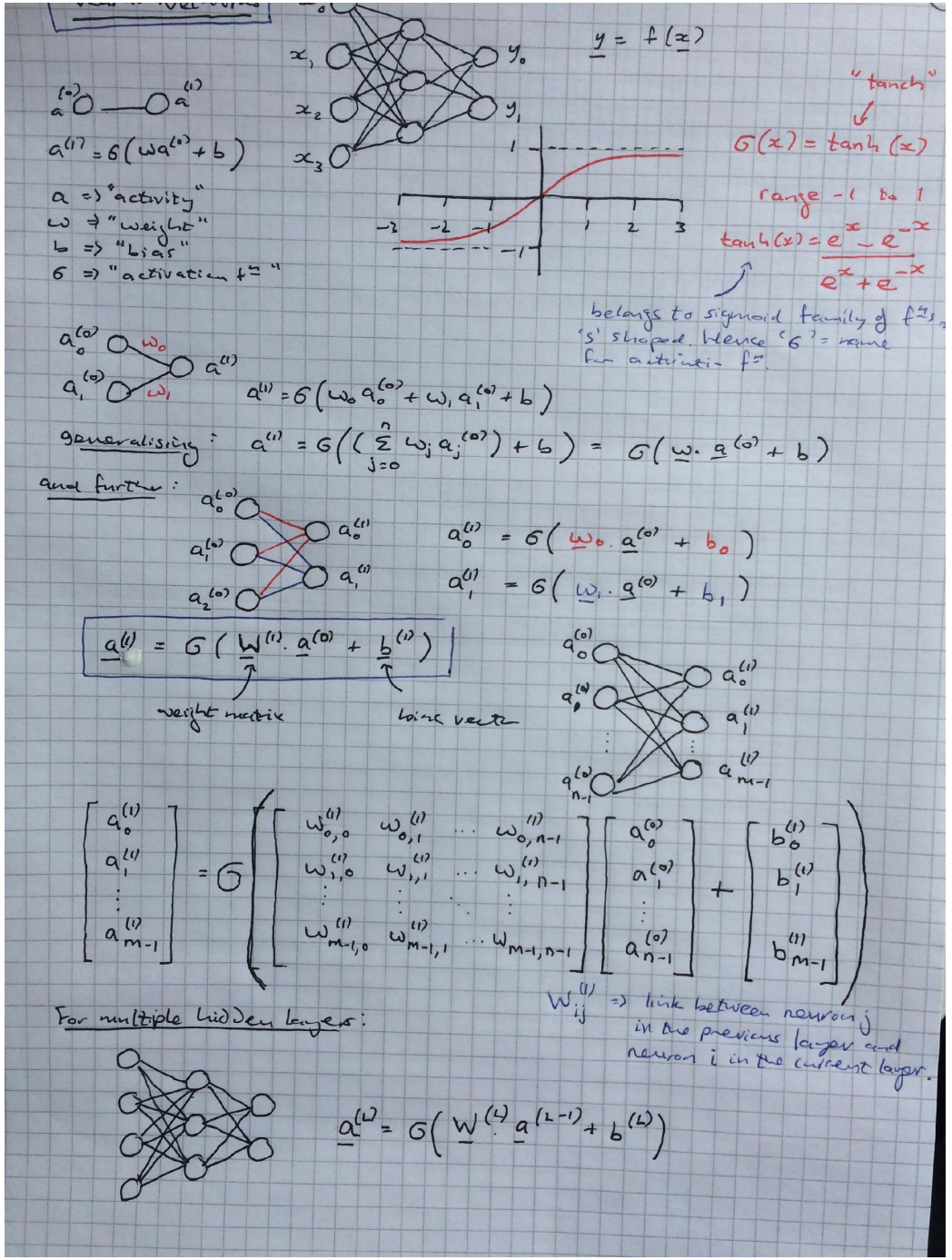
Shorthand representation



https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Notes

- Filter weights are learned from data
- Can be implemented as matrix multiplication (faster)
-
- Efficient **GPU** implementations are possible. We will use train CNNs in the lab but unfortunately only on CPUs
- Implemented as tensor multiplications/additions. [Hence the name "TensorFlow" – we will use TensorFlow, via Keras, in the lab]



my notes from a Coursera course showing why neural networks need matrix algebra

Famous CNNs

1. LeNet-5
2. AlexNet
3. VGG-16
4. ResNet
5. Inception -> GoogLeNet

LeNet-5

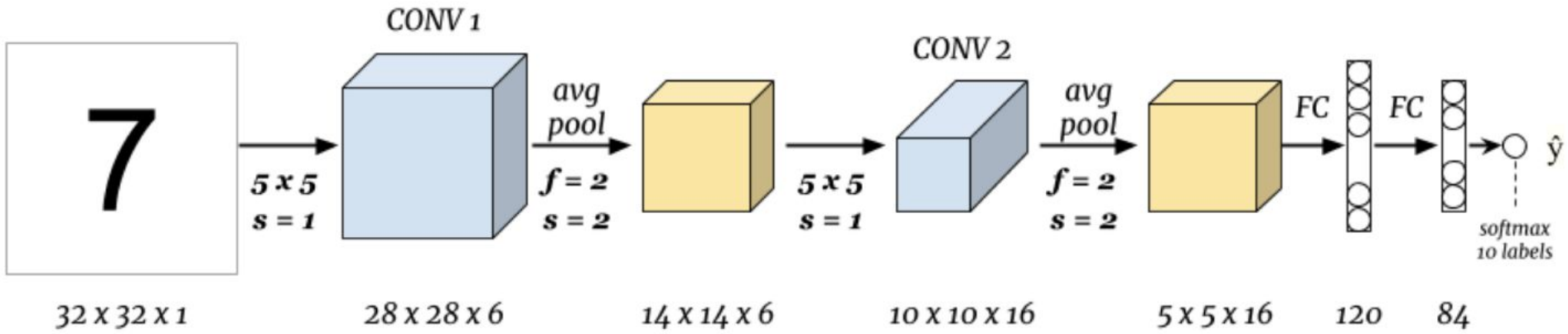


image from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

AlexNet

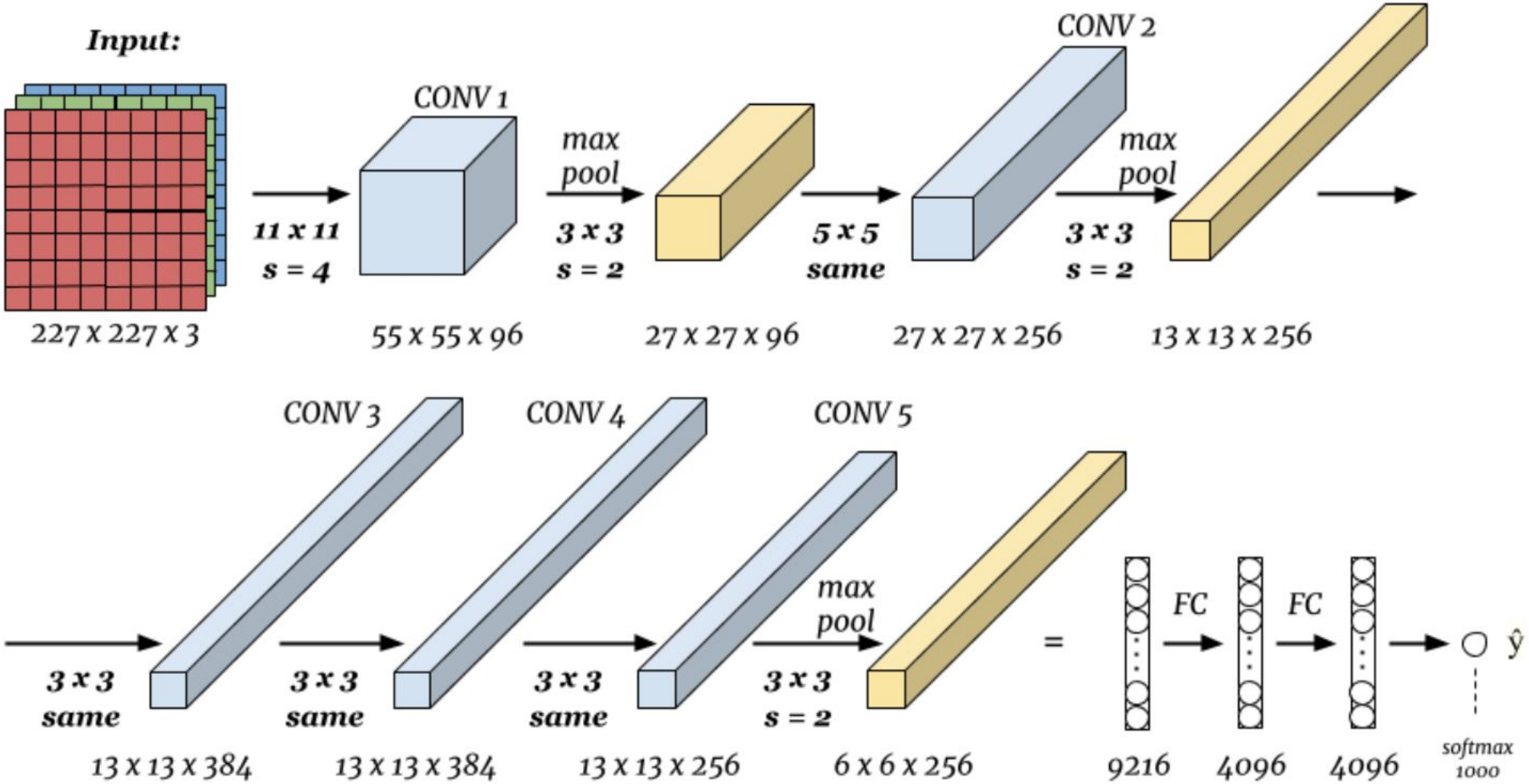


image from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

VGG-16

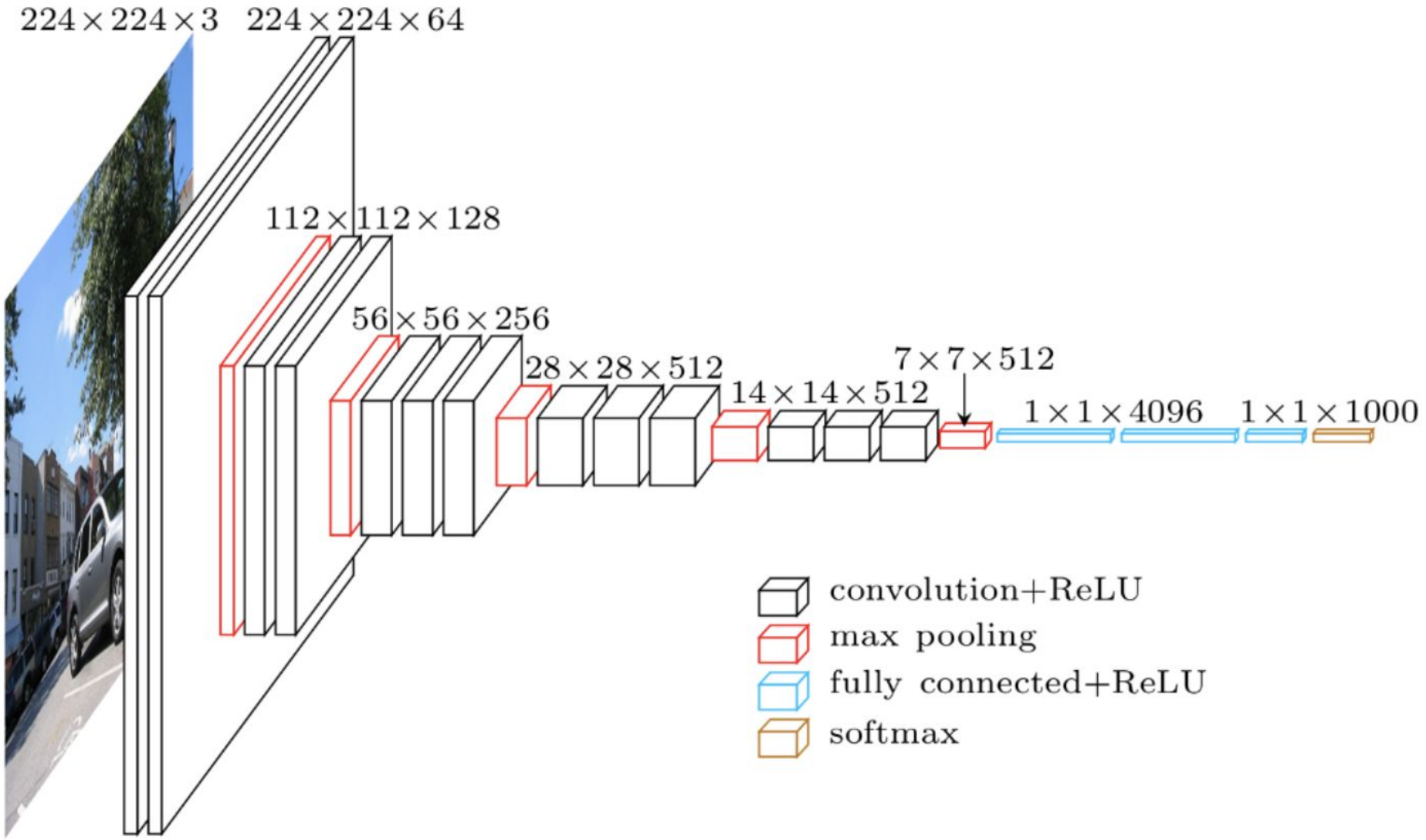


image from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

ResNet

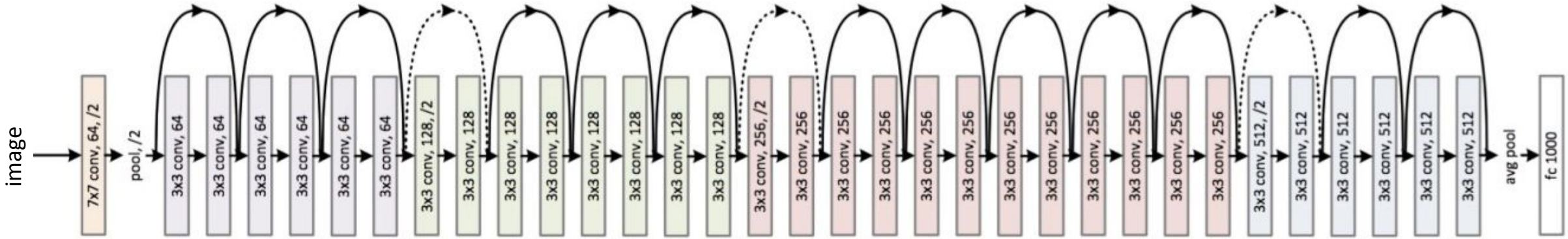
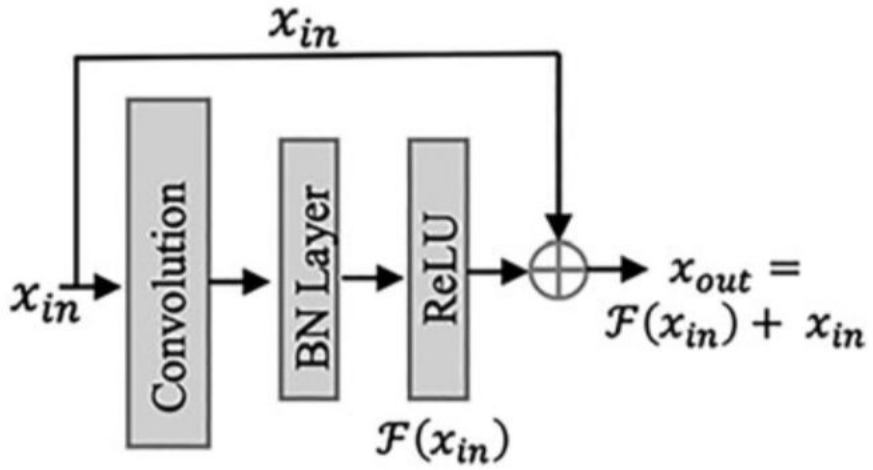


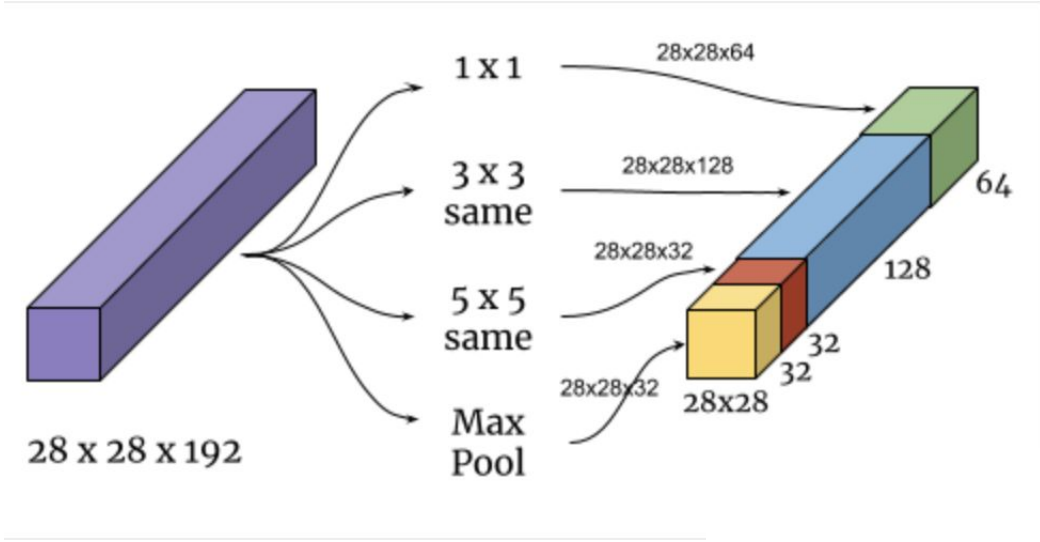
image from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

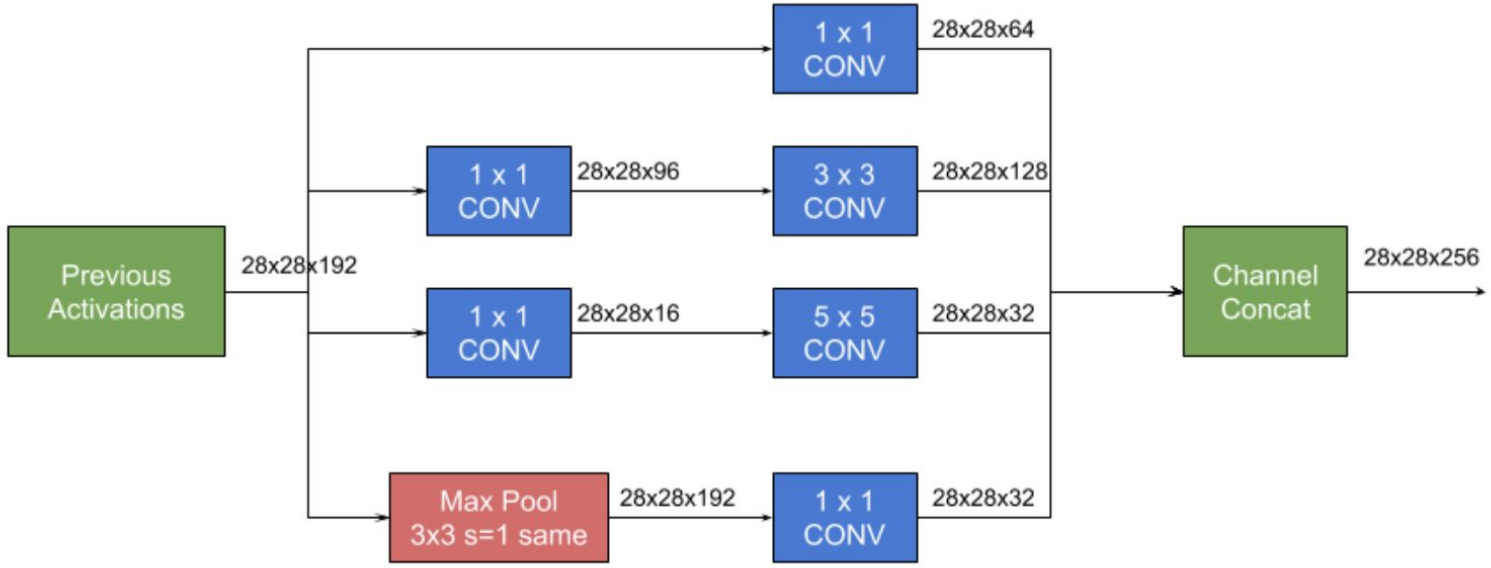
Skip/residual connections



An example of a skip connection, connecting the input with the output of one convolution block (consisting of a convolutional layer, a batch normalization layer, and a ReLU activation function) (from Gupta et al, 2019)

Inception block





images from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

GoogLeNet

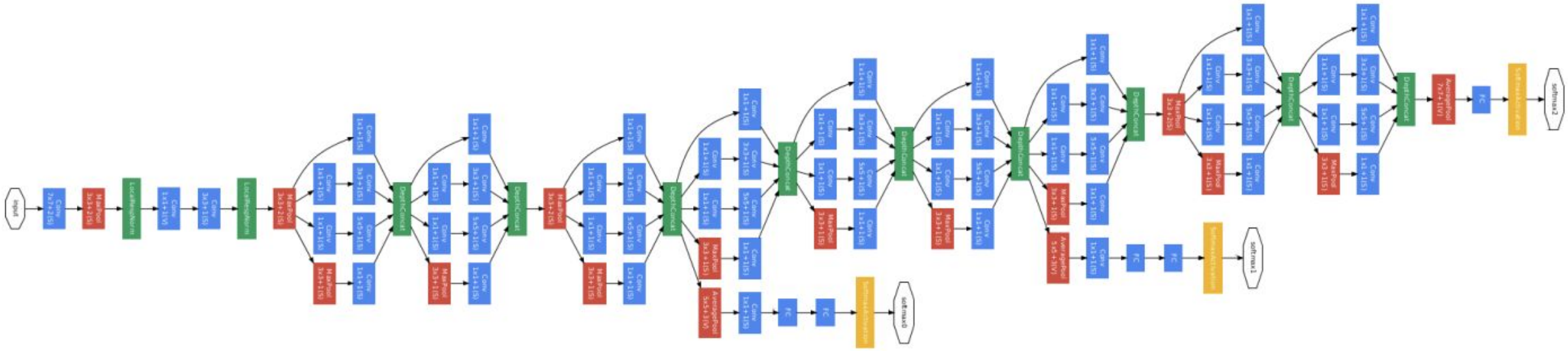
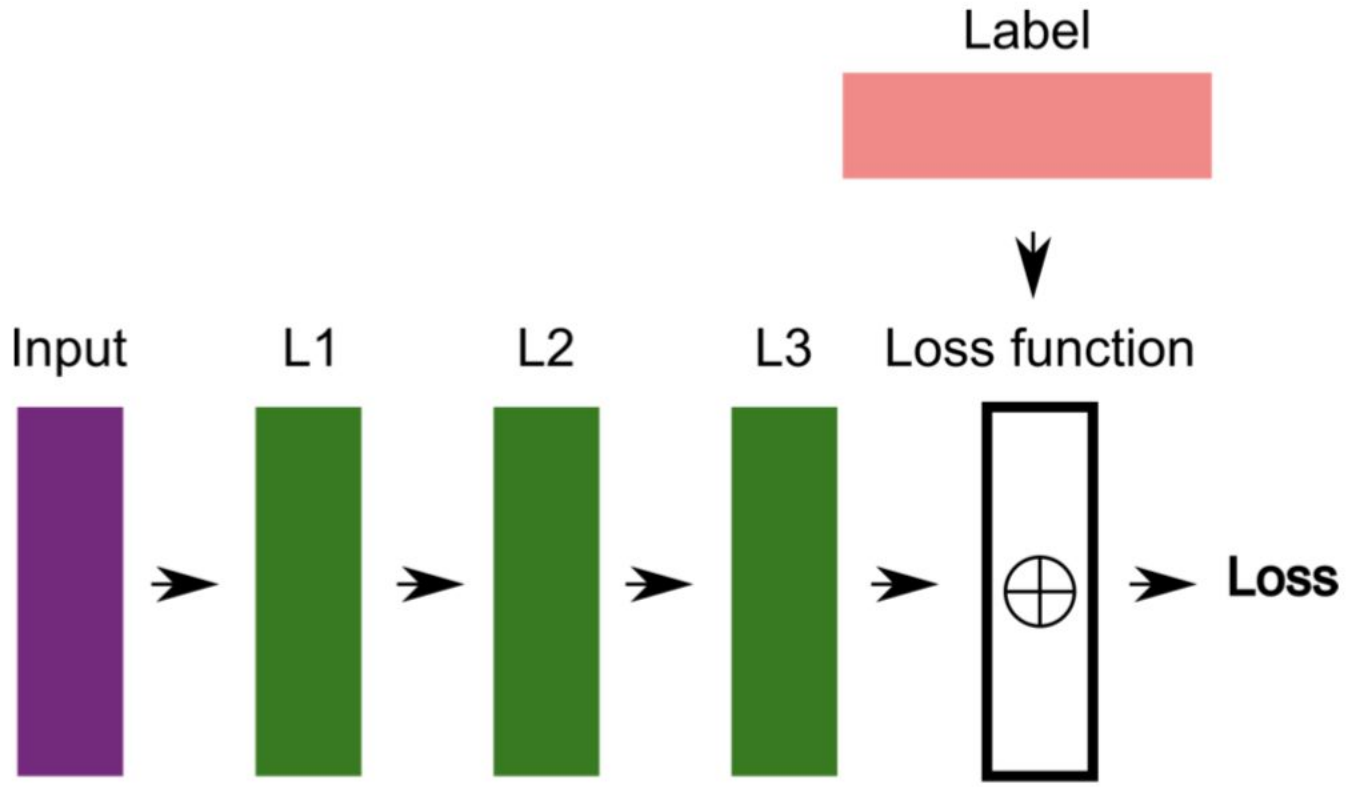


image from https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

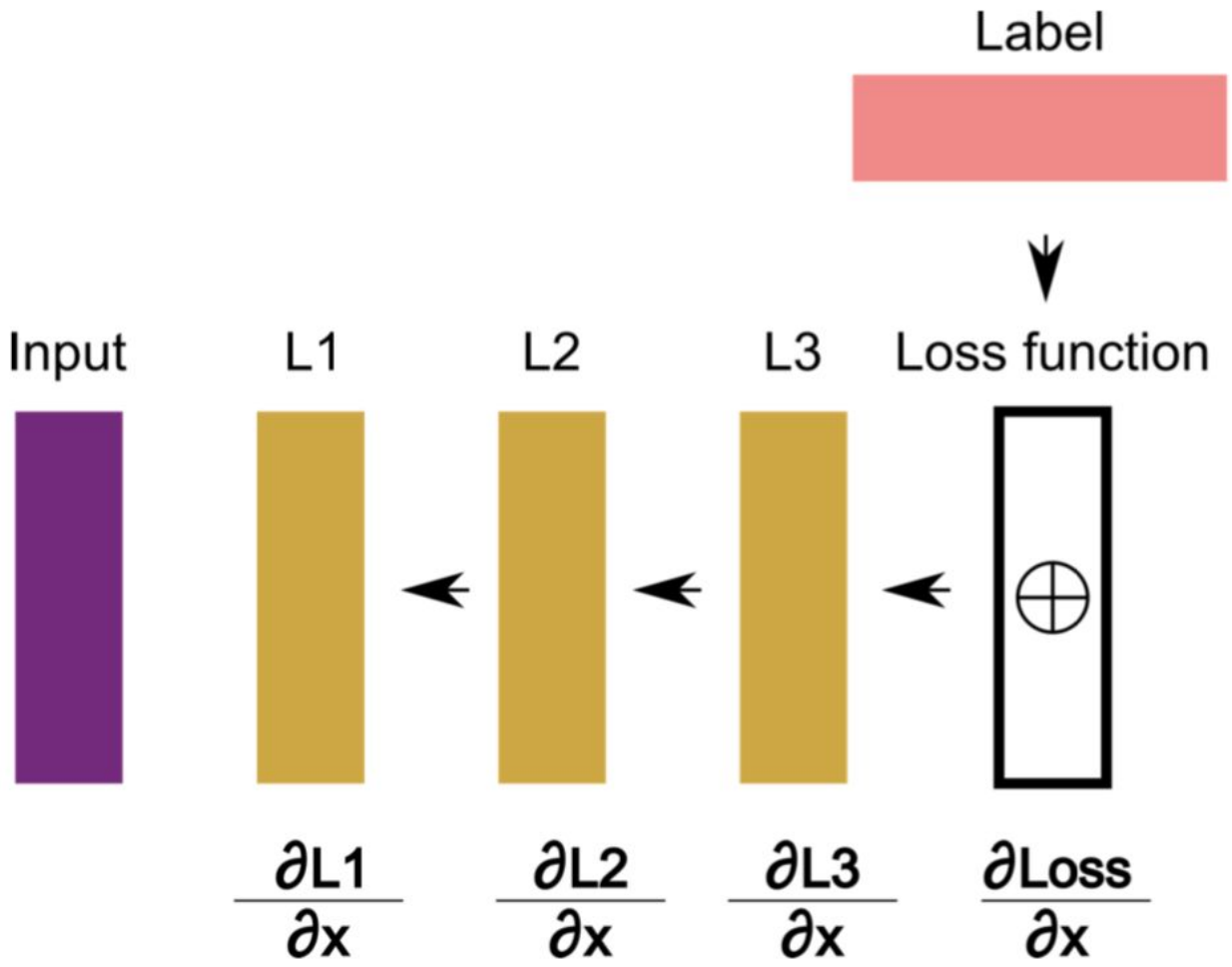
How does the network learn?

- Learns from its mistakes using a loss function
- Contains hundreds of parameters/variables
- Find the effect of each parameter in making mistakes using back propagation
- Increase/decrease the parameter values so as to make less mistakes using Gradient Descent
- Do all the above several times iteratively

Forward pass



Backward pass



Deep neural network working

- Training
- Loss function
- Forward pass
- Backward pass/Back propagation
- Gradient descent optimization

Training

- Data partitioning - training, validation, testing
- Preprocessing
- Network architecture
- Hardware
- Weight initialization

Loss function

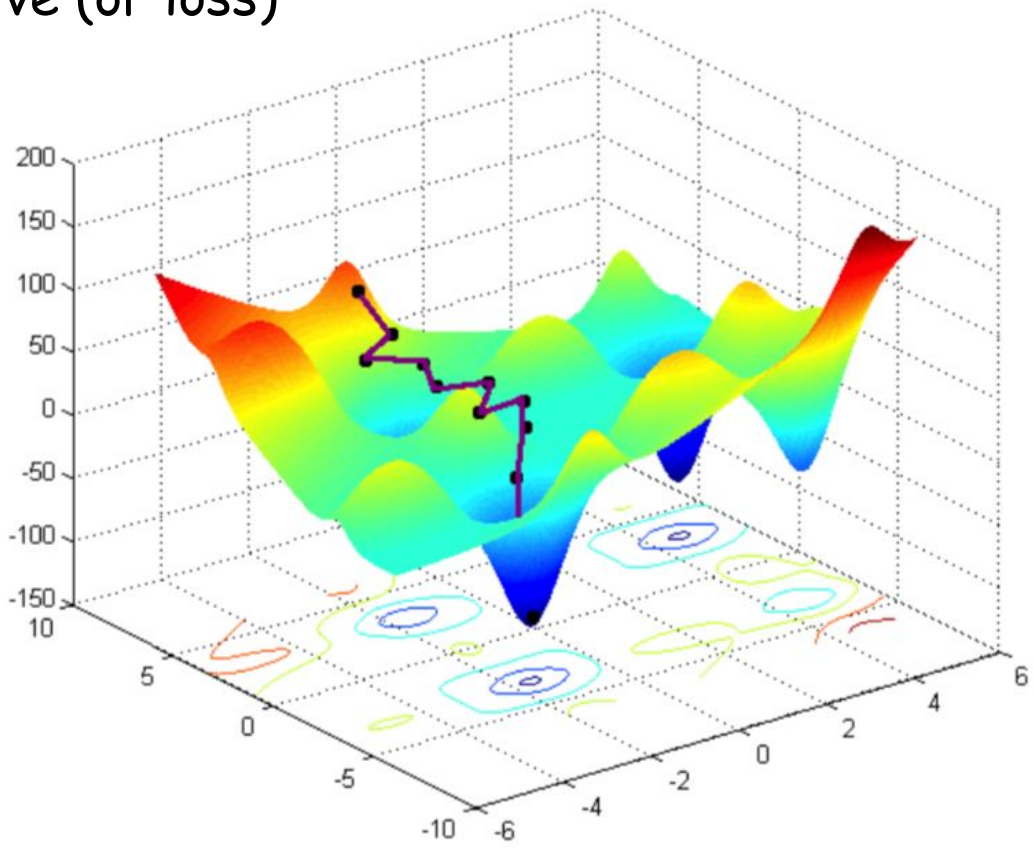
- Mean square error loss
- Softmax cross-entropy loss
- etc

Back propagation

- Follow chain rule
- Find the analytical gradient of each components
- Find derivative of activation functions
- Total gradient is the product of the gradient at that node multiplied with the gradient coming to the node
- If connected to multiple nodes, add the gradients
- For ReLU activations gradient pass to positive outputs only

Gradient descent

- Gradient descent is a way to minimize an objective (or loss) function, $J(\theta)$
- Parameters θ
- Gradient of the objective function $\nabla\theta J(\theta)$
- Update in the opposite direction of gradient
- Learning rate η
- Its variants:
  - Batch gradient descent
  - Stochastic gradient descent
  - Mini-batch gradient descent



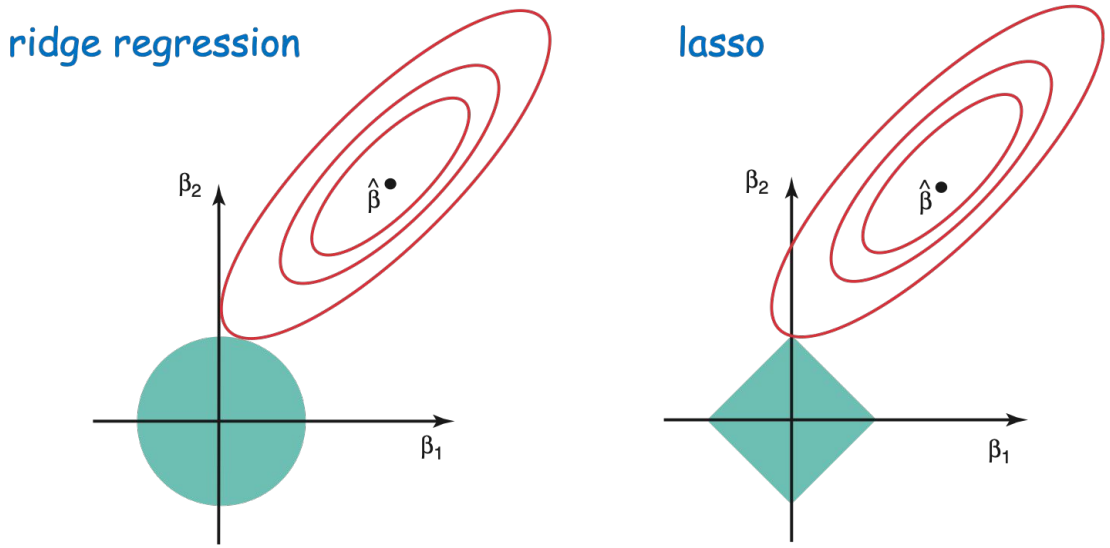
http://www.phoenix-int.com/software/benchmark_report/bird.php

Regularization

To reduce over-fitting to training data:

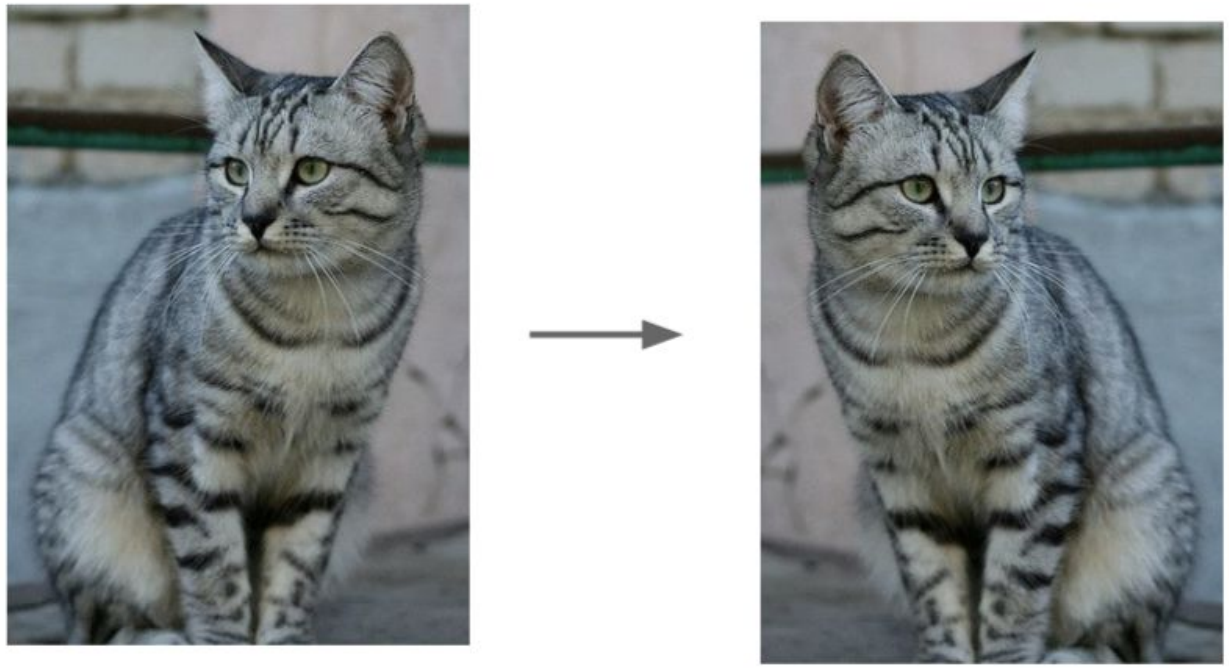
- Data augmentation
- Drop out
- Batch normalization
- *L2, L1* regularization of weights

recall from Ola's slides:



Data Augmentation

Horizontal Flips



Data Augmentation

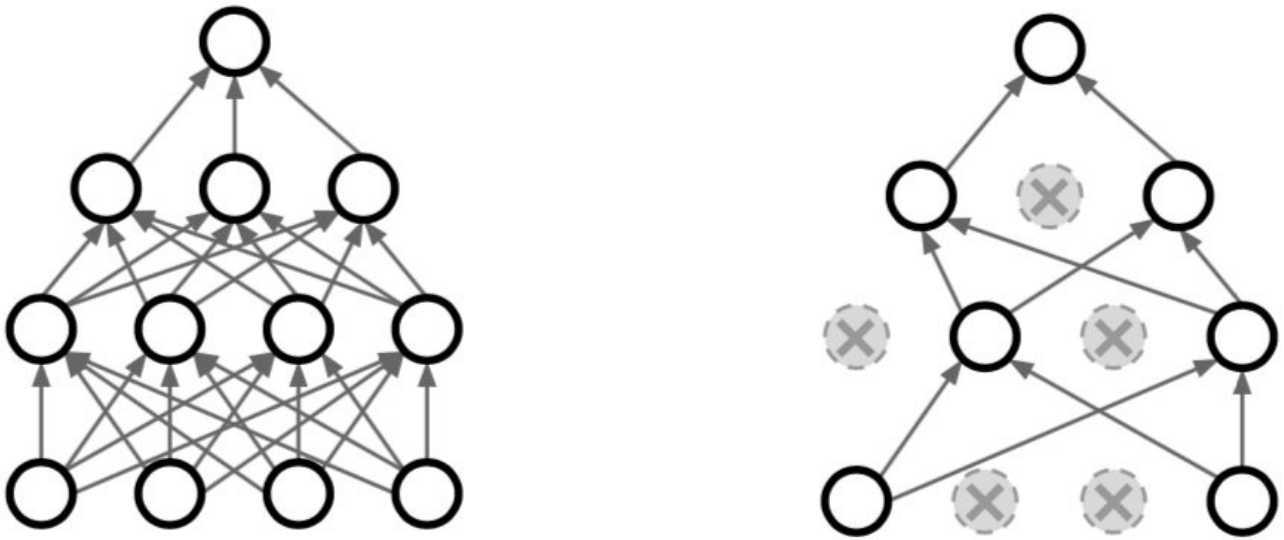
Get creative for your problem!

Random mix/combinations of :

- translation
- rotation
- stretching
- shearing,
- lens distortions, … (go crazy)

Regularization: Dropout

In each forward pass, randomly set some neurons to zero
Probability of dropping is a hyperparameter; 0.5 is common



Srivastava et al, "Dropout: A simple way to prevent neural networks from overfitting", JMLR 2014

The problem of insufficient dataset size

To fit deep CNNs to data you need lots of data! More than we often have for cell image data ☹

solution = **transfer learning**

- use weights from a pretrained model with a similar architecture on a large dataset (e.g. ImageNet)

- either:
  - freeze the initial layers (basic image feature extractor) with the pretrained weights. Then train the weights in the final layer(s);

  - use the pretrained weights as good starting values and train the entire network

Additional hyper parameters

- Learning rate
- Learning rate decay
- Momentum
- Network depth
- Network width
- Mini batch size

Deep learning tools

- Tensorflow
- Keras
-
- Caffe
- Torch
- PyTorch
- Theano
- Neon
- Chainer

Deep learning (Keras) workflow

1. read in required packages
2. read in data – inspect/plot
3. standardize or normalize data & one-hot encoding for categorical responses
4. train-validation-test split (or use cross validation)
5. define & summarize model
6. compile model (choose loss function, optimizer and metrics to record)
7. fit model (train for several epochs)
8. evaluate model (accuracy on test set, inspect confusion matrices etc)

Bonus material:
Cell segementation with U-Net

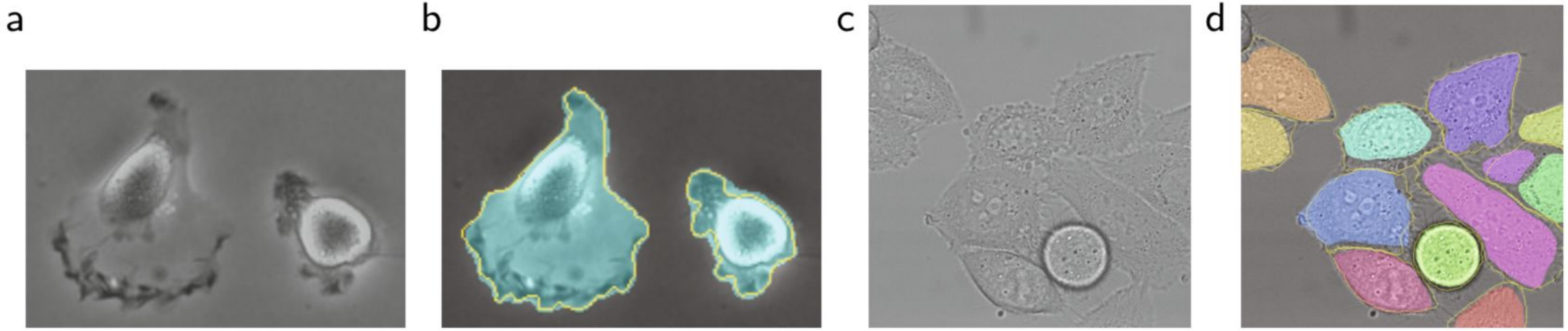
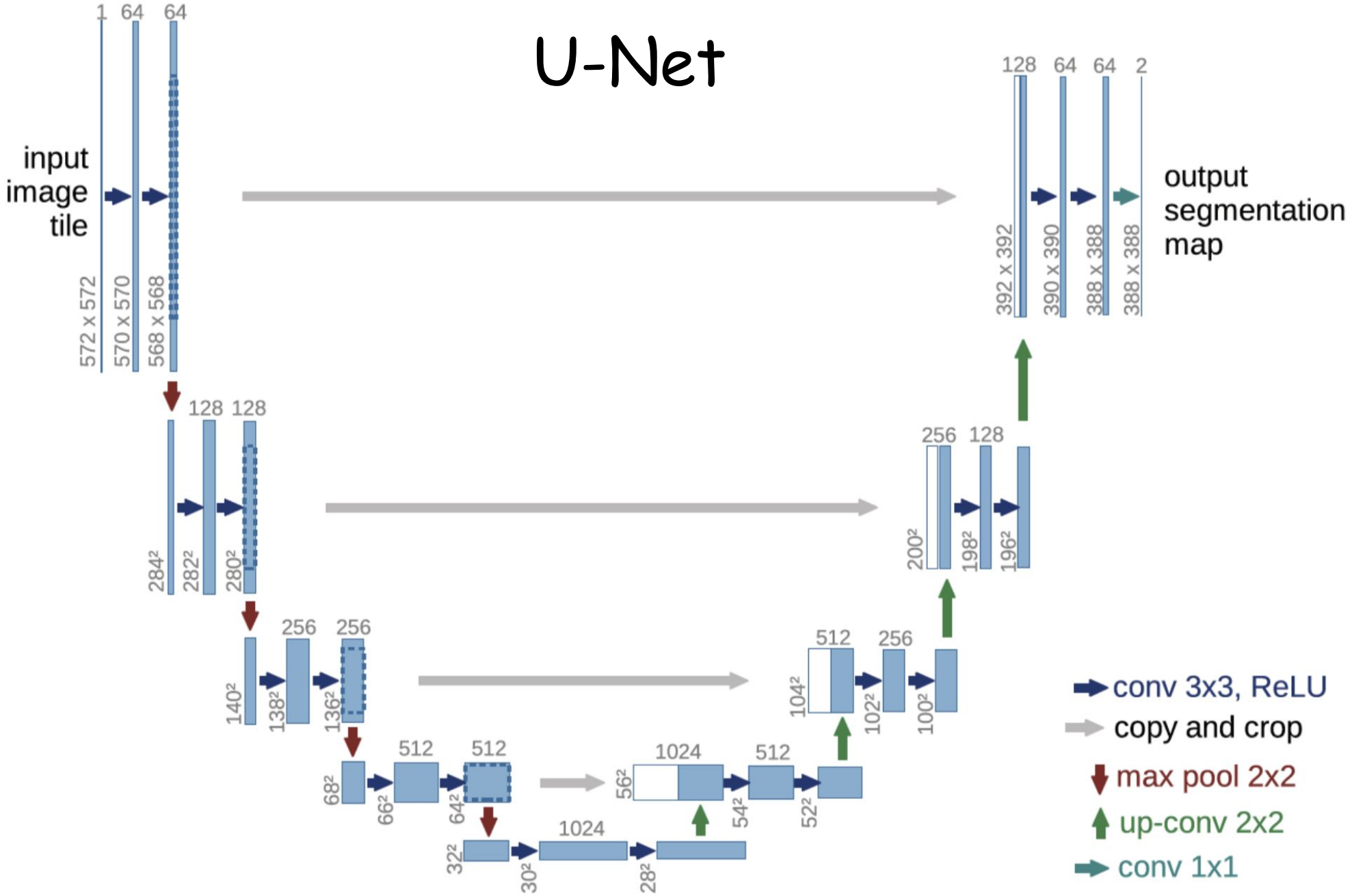
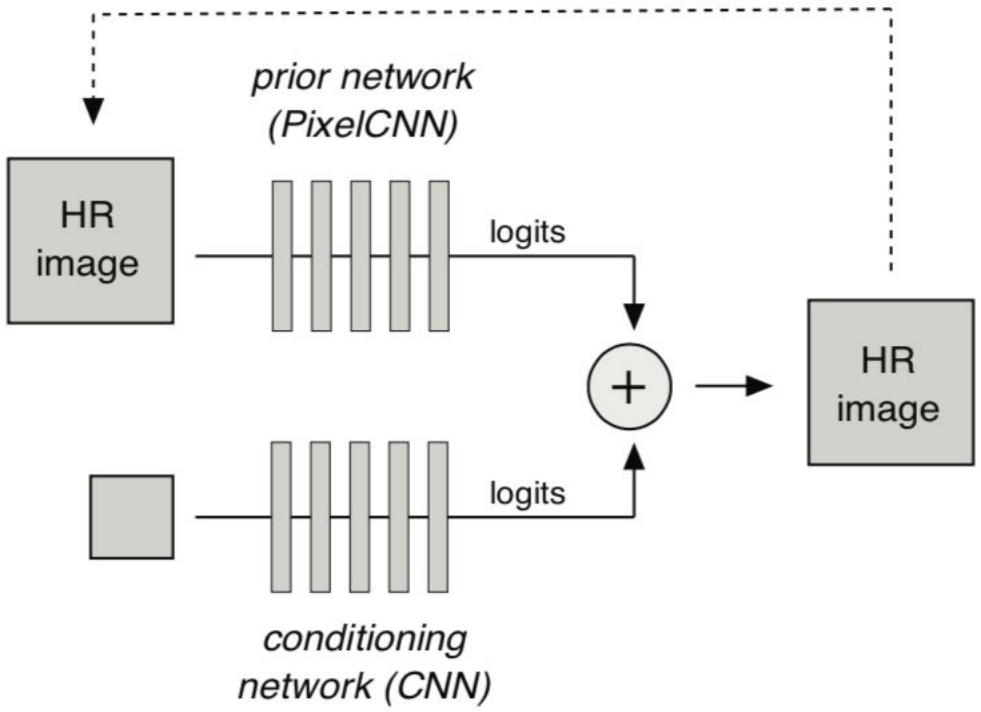


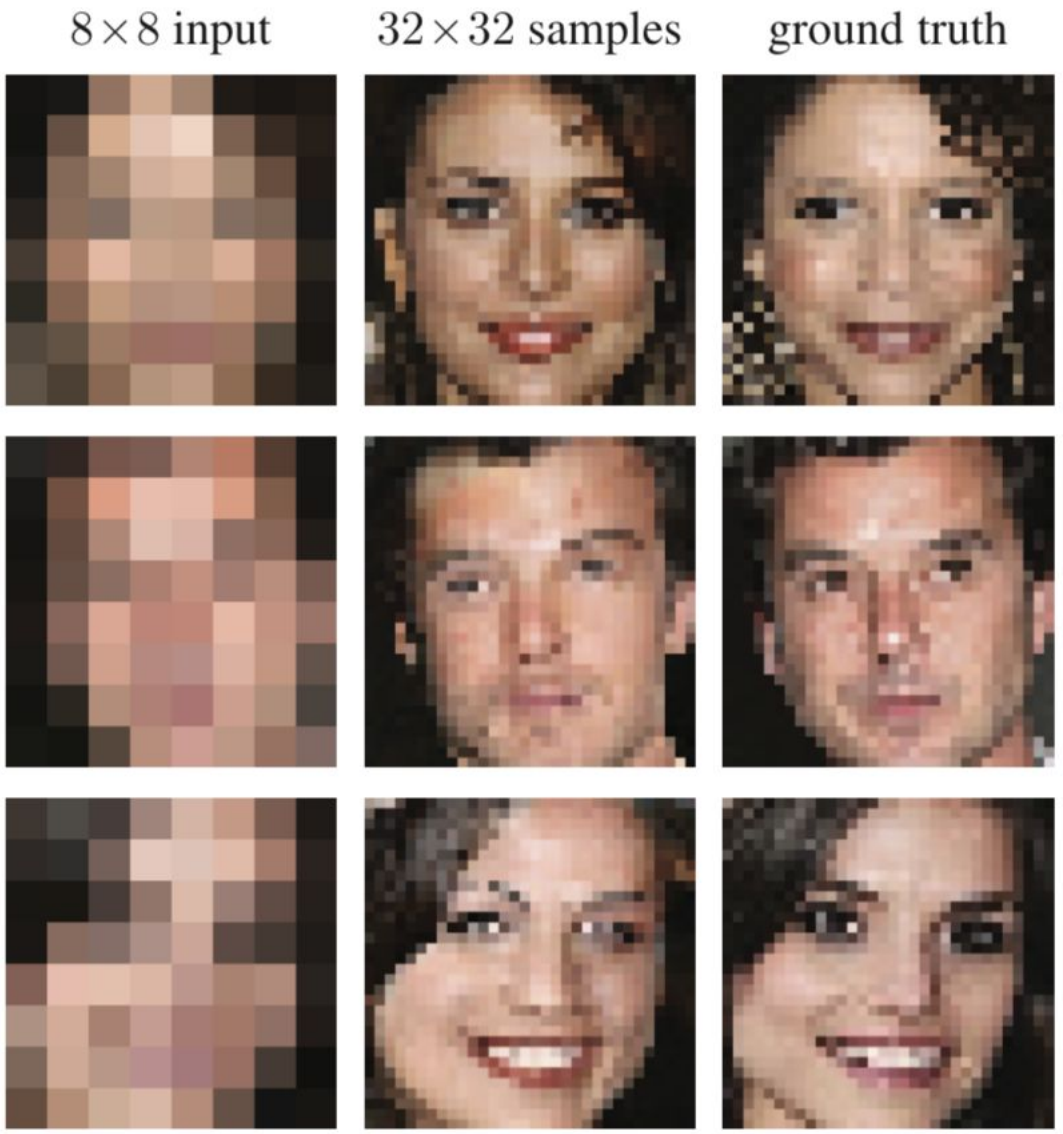
Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

U-Net


input image tile
572 x 572
570 x 570
568 x 568
1 64 64
128 128
284² 282² 280²
256 256
140² 138² 136²
512 512
68² 66² 64²
1024
32² 30² 28²
1024 512
56² 54² 52²
512 256
104² 102² 100²
256 128
200² 198² 196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

Super resolution





Dahl, R., Norouzi, M., and Shlens, J. (2017). Pixel Recursive Super Resolution. In 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5449-5458.

Other cool deep learning stuff

- Recurrent Neural Networks (RNNs) & LSTMs
- Autoencoders and variational autoencoders
- Generative Adversarial Networks (GANs)
- etc.

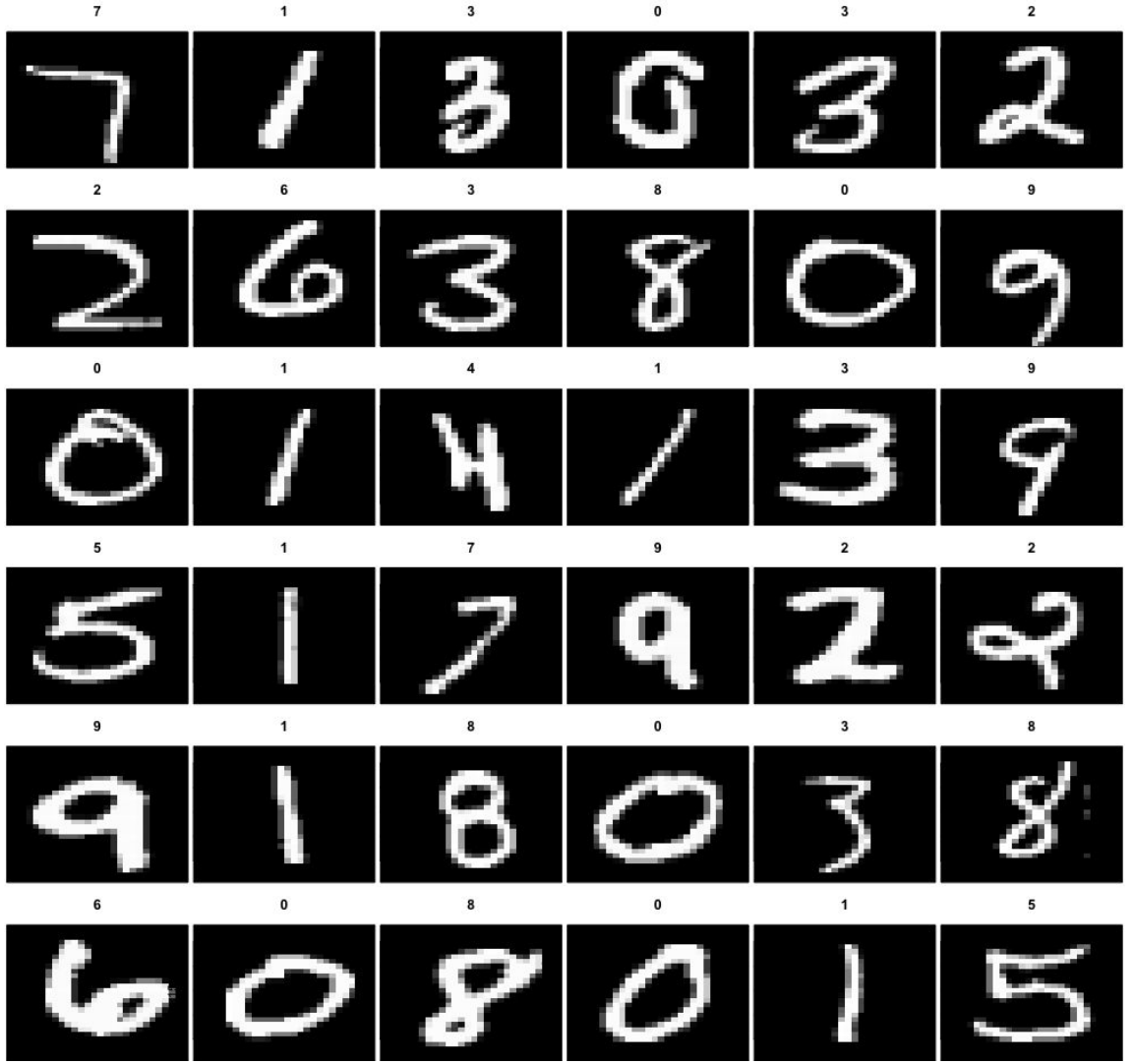
Details about tomorrow's computer lab

Warm-up exercise:

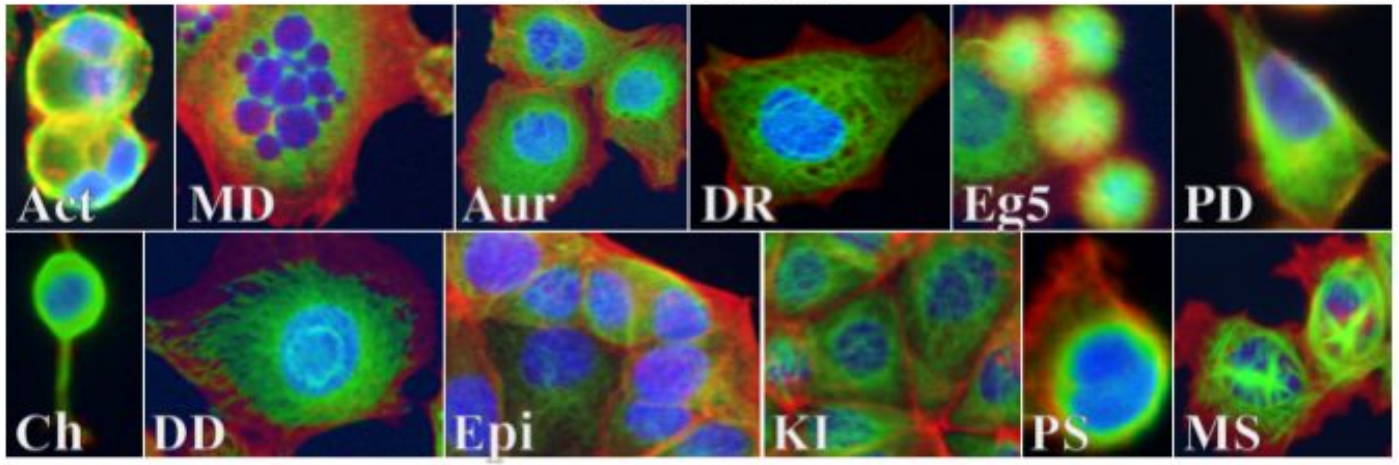
Train a simple CNN on the MNIST data

- MNIST = database of handwritten digits
- 60,000 training images and 10,000 test images
- Digits size-normalized and centered in a 28 x 28 pixel image.
- A classic benchmark dataset for machine learning!
- Comes pre-installed in Keras!

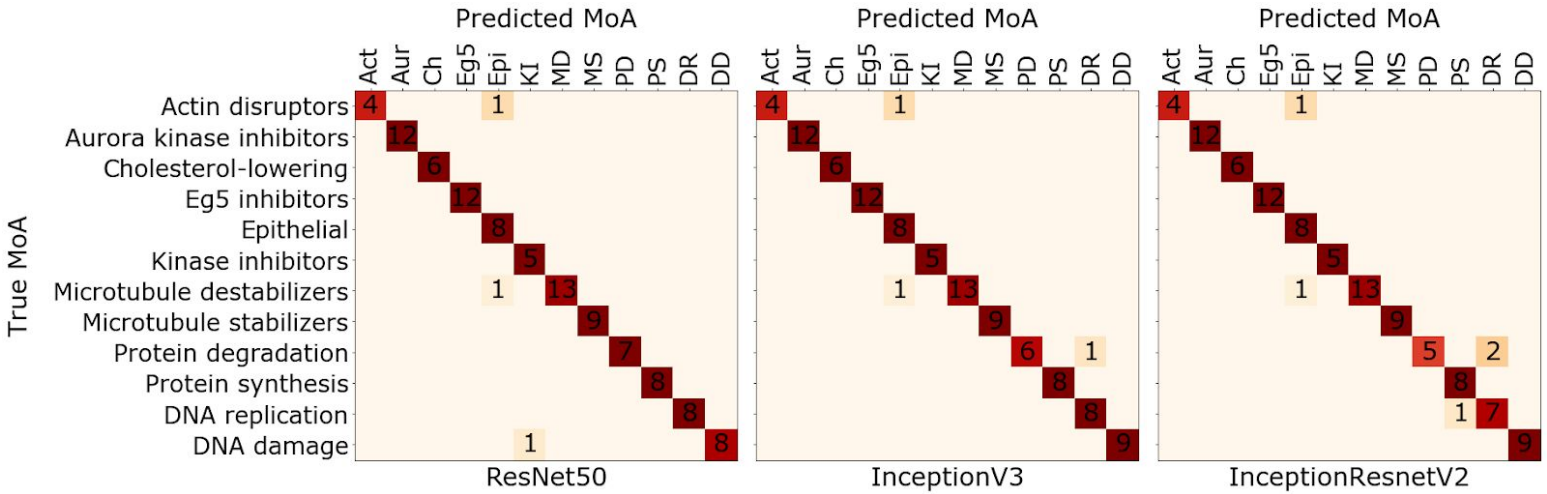
Go to 



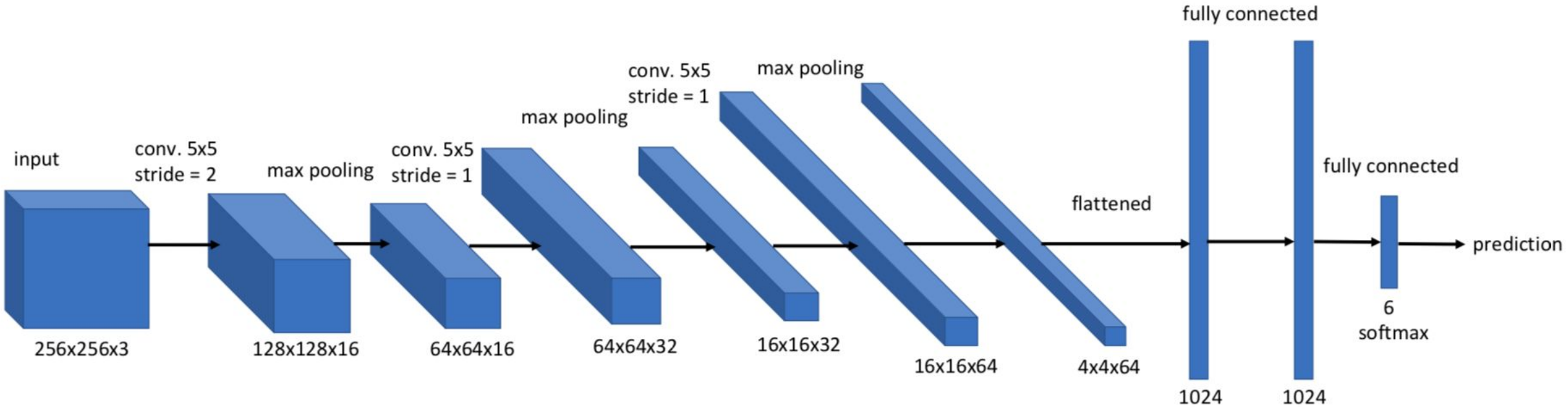
Main lab exercise:
Classification of cell morphological changes with CNNs



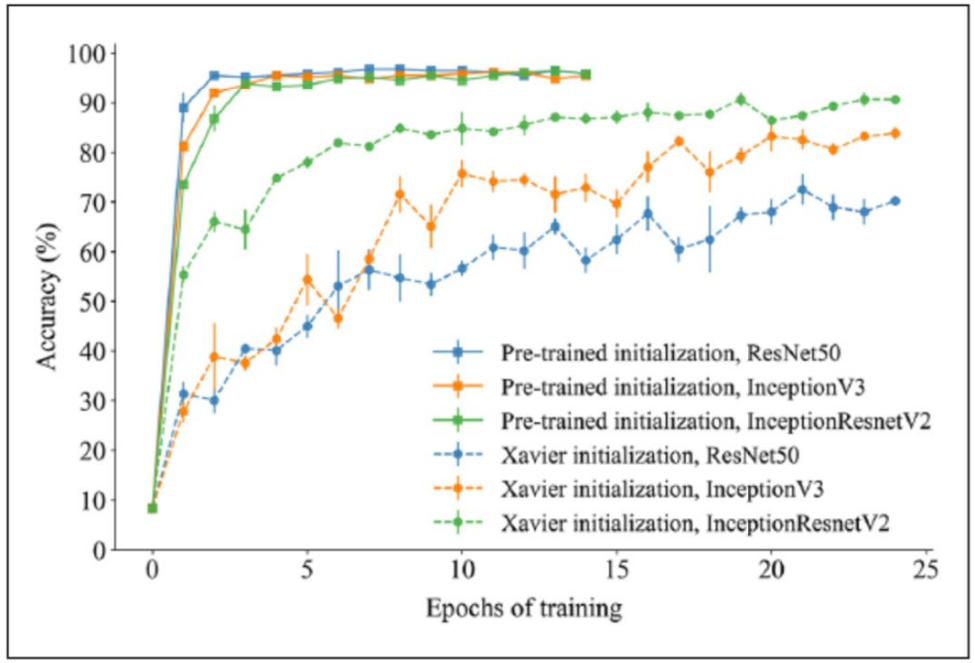
Kensert, A., Harrison, P.J., and Spjuth, O. (2019). Transfer Learning with Deep Convolutional Neural Networks for Classifying Cellular Morphological Changes. SLAS DISCOVERY: Advancing Life Sciences R&D

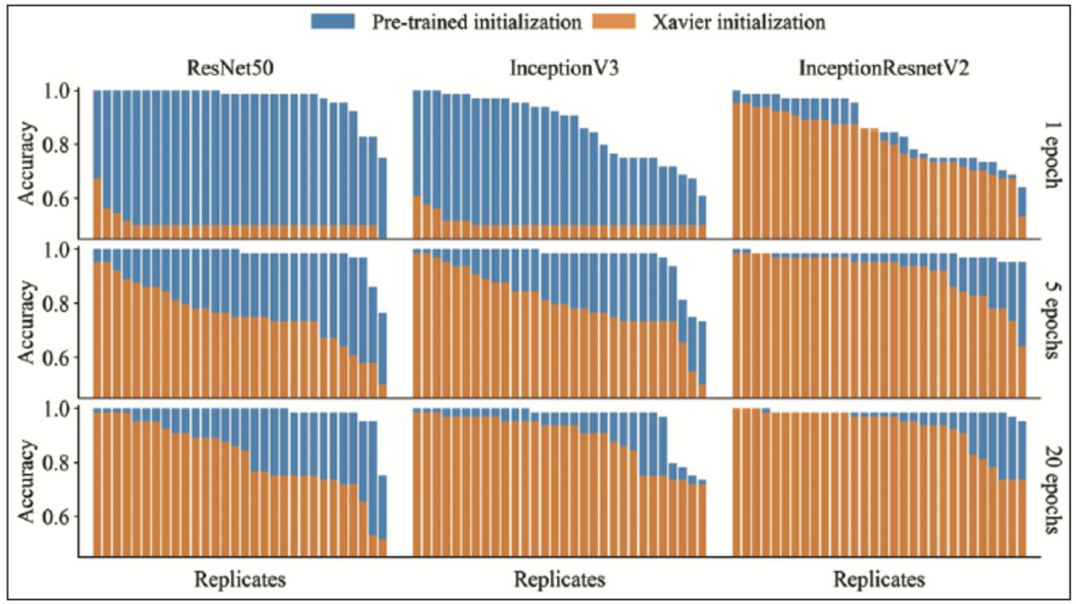


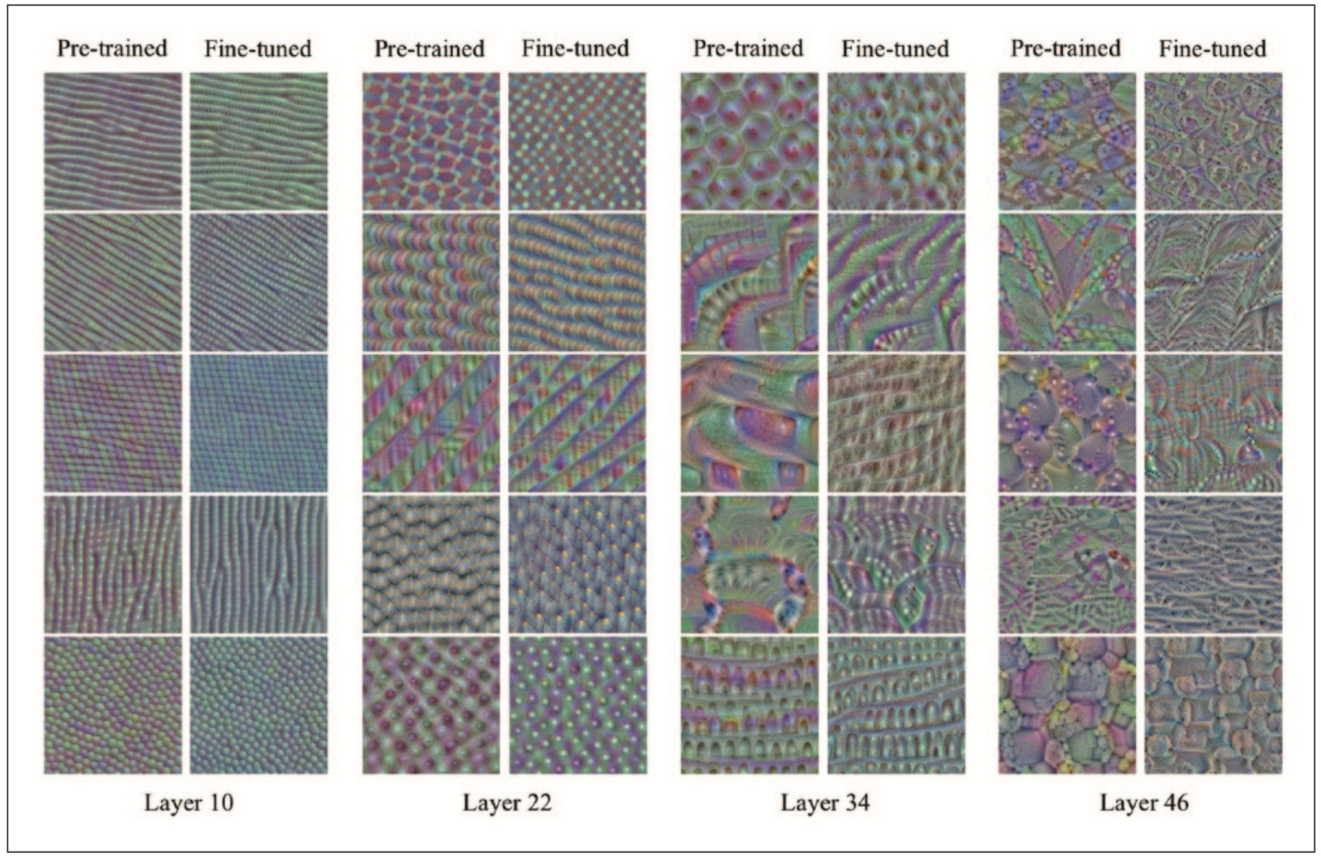
LeNet inspired CNN that you will define and fit to the MoA dataset during the lab



More figures from Kensert, A., Harrison, P.J., and Spjuth, O. (2019).







THE END