UNIVERSITY of HAWAIʻI® at MĀNOA

Image Classification using Convolutional Neural Networks (CNNs)

MSc. Jonas Krause

Prof. Dr. Lipyeow Lim
Prof. Dr. Kyungim Baek

Agenda

Introduction

•What we see vs. What computers see (MNIST and CIFAR Datasets)

•Hand-Crafted Features for Image Classification

Deep Learning

•Convolutional Neural Networks (CNNs)

- Architecture (Convolutional, Pooling, and Fully Connected Layers)
- Successful CNN Architectures

Training

•Backpropagation

•Overfitting, Regularization and Dropout

Experiments

Transfer Learning

Complex Networks

Agenda

Introduction

- What we see vs. What computers see (MNIST and CIFAR Datasets)
- Hand-Crafted Features for Image Classification

Deep Learning

- Convolutional Neural Networks (CNNs)
  - Architecture (Convolutional, Pooling, and Fully Connected Layers)
  - Successful CNN Architectures

Training

- Backpropagation
- Overfitting, Regularization and Dropout

Experiments

Transfer Learning

Complex Networks

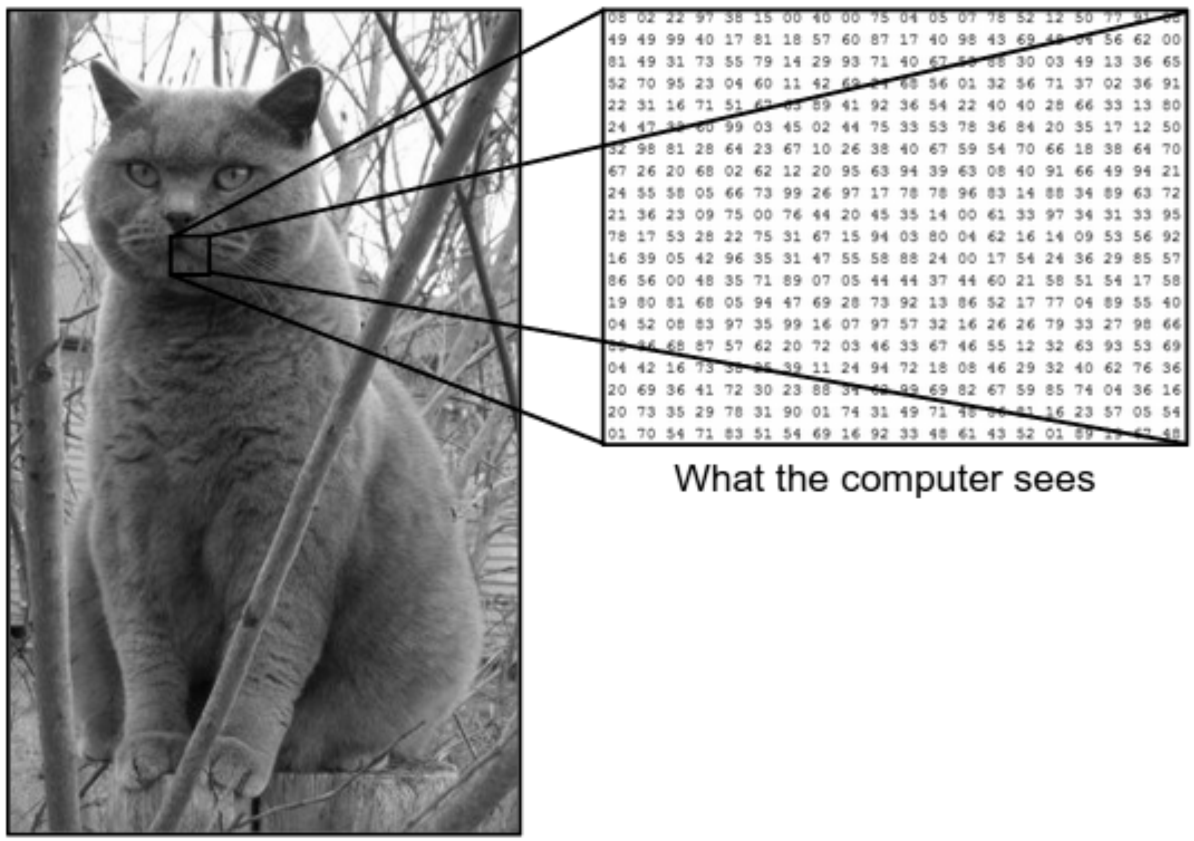
Introduction

- Image classification is the task of taking an input image and outputting a class or a **probability of classes** that best describes the image
  - For humans, this task is one of the first skills we learn and it comes **naturally** and **effortlessly** as adults

  - Being able to quickly **recognize patterns**, generalize from **prior knowledge**, and adapt to **different image environments** are difficult tasks for machines

Introduction

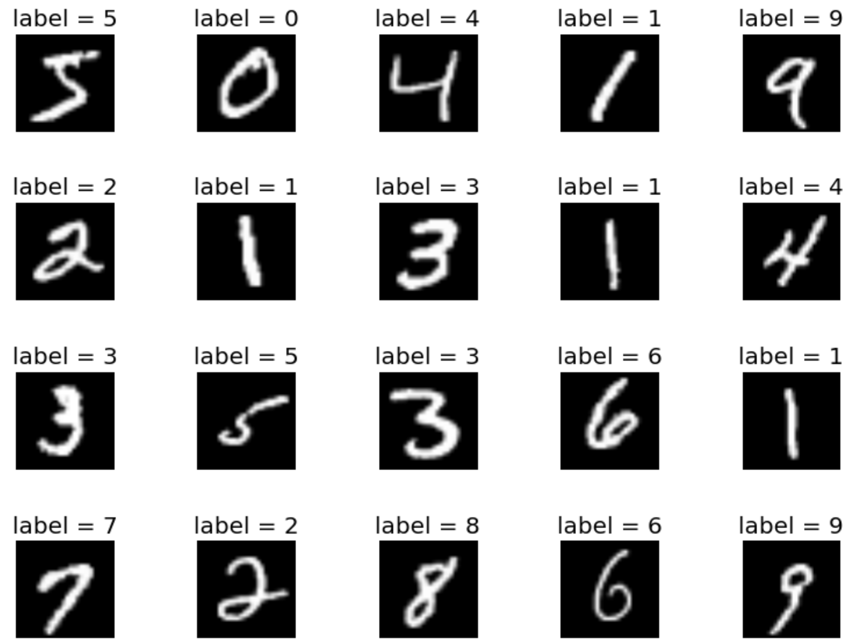
What we see vs. What computers see



Introduction

MNIST Dataset (*http://yann.lecun.com/exdb/mnist/*)

•60,000 training examples



•10,000 test examples

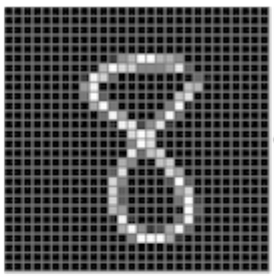
•Rank of best Classifiers and Errors

•Currently Best Accuracy:

- Ciresan et al. CVPR 2012 ➔ **99.77%**

Introduction

MNIST Dataset



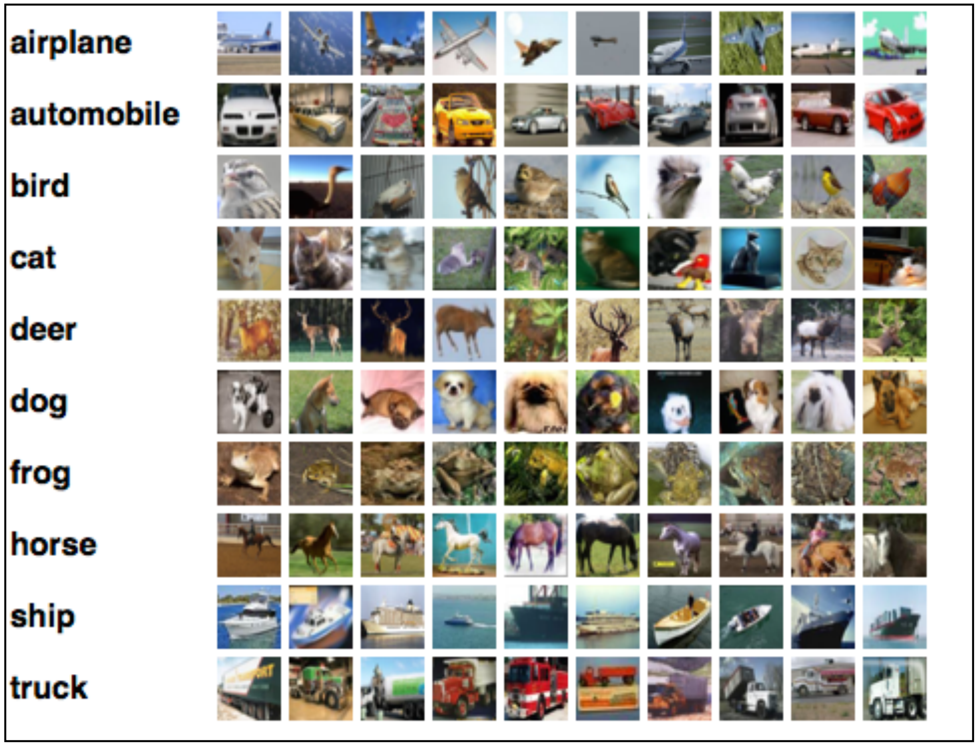
28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Introduction

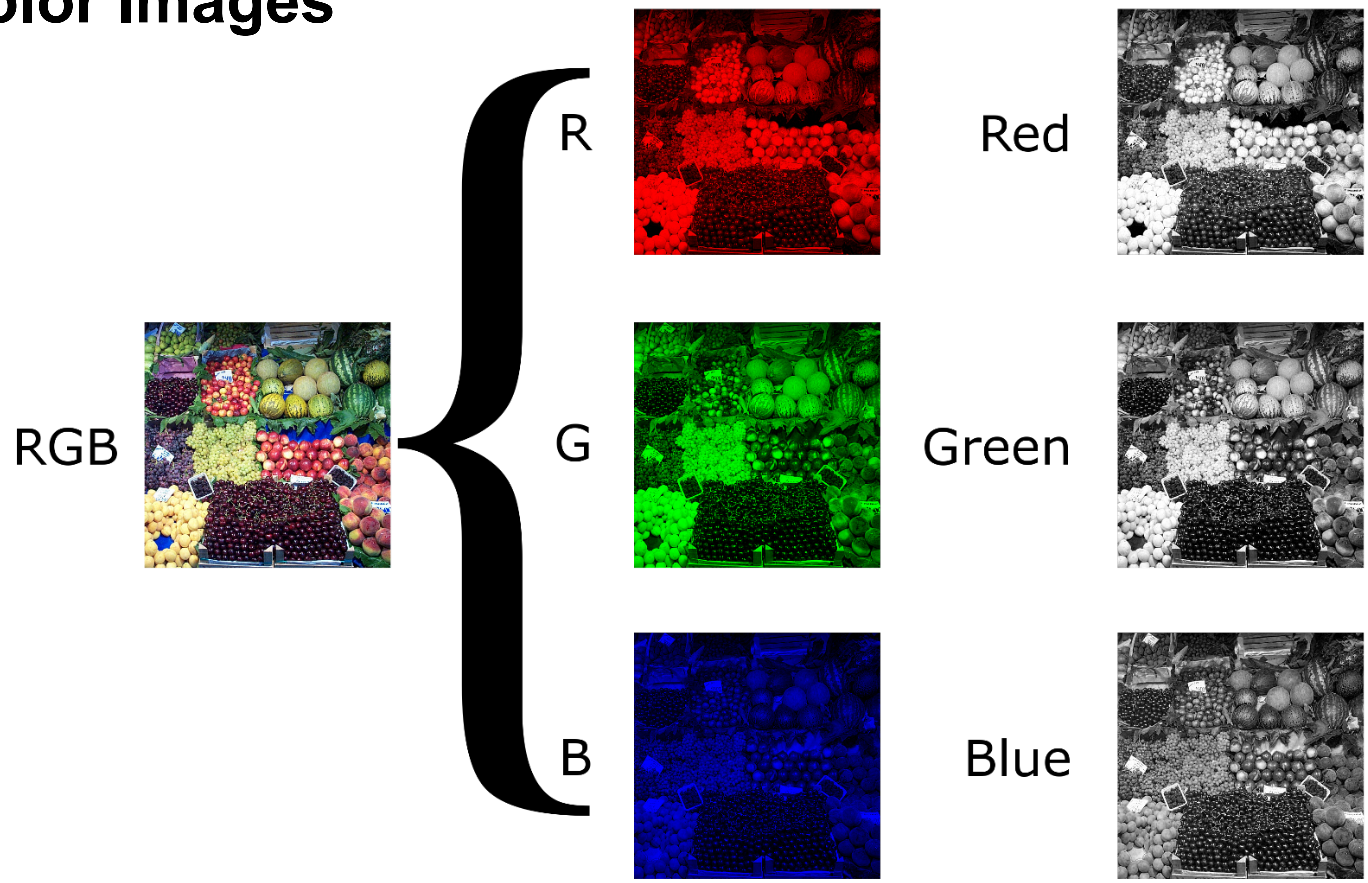
CIFAR-10 Dataset (*https://www.cs.toronto.edu/~kriz/cifar.html*)

•Consists of 60,000 32x32 **color** images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.



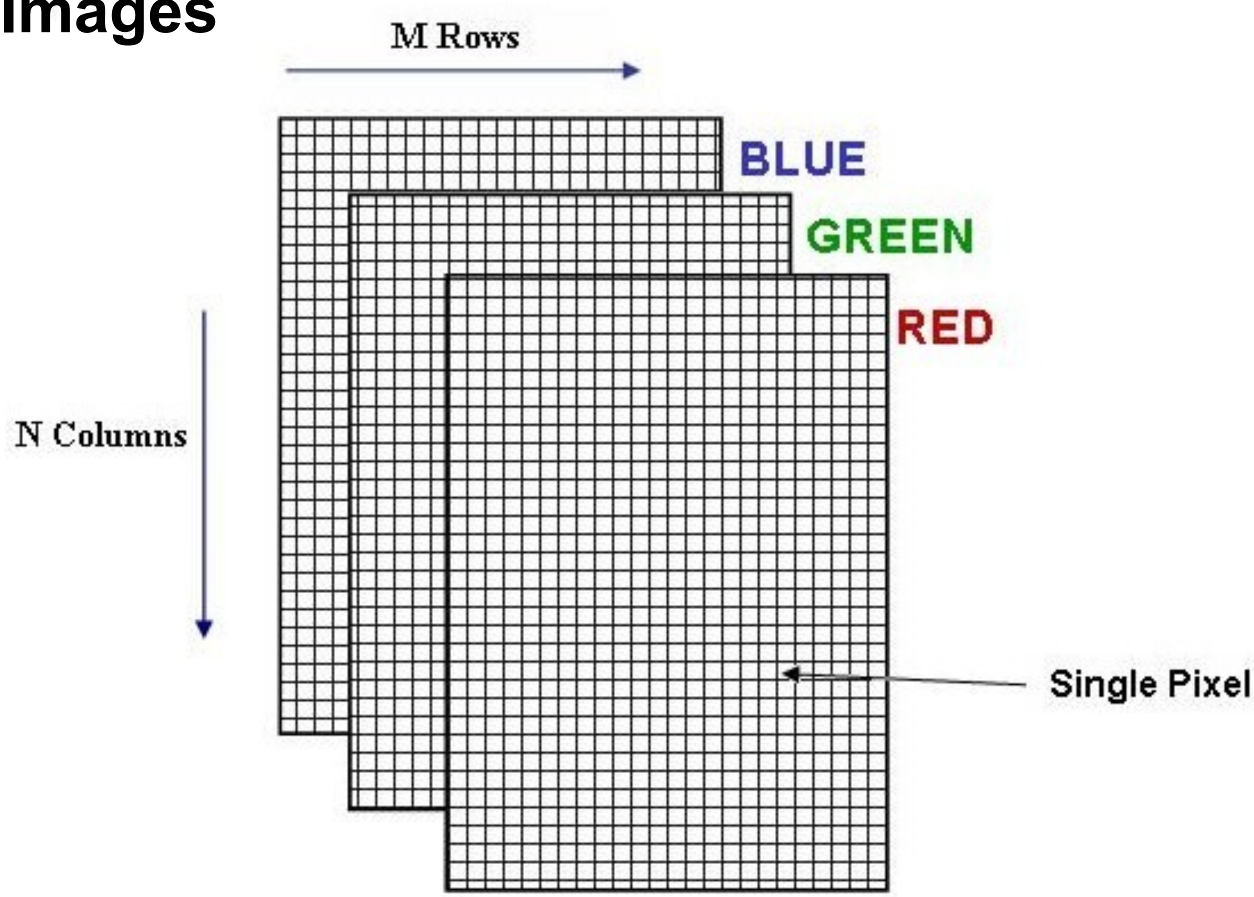
Introduction

Color Images



Introduction

Color Images



Introduction

Image Classification (previous Deep Learning)

•Hand-Craft Features

•**Texture Features**: Histogram based, Entropy, Haralick features (Co-occurrence matrix), Gray-level run length metrics, Local Binary Pattern, Fractal, etc.

•**Morphological Features**: Hu's moments, Shape features, Granulometry, Bending Energy, Roundness ratio, etc.

Agenda

Introduction

- What we see vs. What computers see (MNIST and CIFAR Datasets)
- Hand-Crafted Features for Image Classification

Deep Learning

- Convolutional Neural Networks (CNNs)
  - Architecture (Convolutional, Pooling, and Fully Connected Layers)
  - Successful CNN Architectures

Training

- Backpropagation
- Overfitting, Regularization and Dropout

Experiments

Transfer Learning

Complex Networks

Deep Learning (DL)

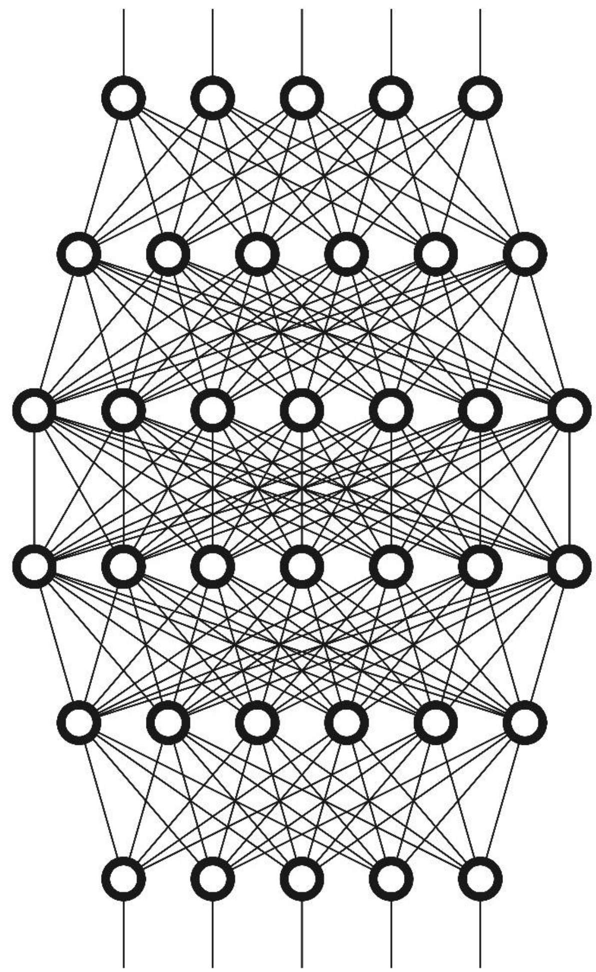
"Deep Learning is a new area of Machine Learning, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.” *http://deeplearning.net/*

- Key Concepts of Deep Neural Networks
  - Deep-learning networks are distinguished from the more common single-hidden-layer neural networks by their **depth**
  - More than **three** layers (including input and output) qualifies as “deep” learning
  - In deep-learning networks, each layer of nodes trains on a distinct set of features based on the **previous** layer’s output
  - The further you advance into the neural net, the more **complex** the features your nodes can recognize, since they aggregate and recombine features from the previous layer

Deep Learning (DL)

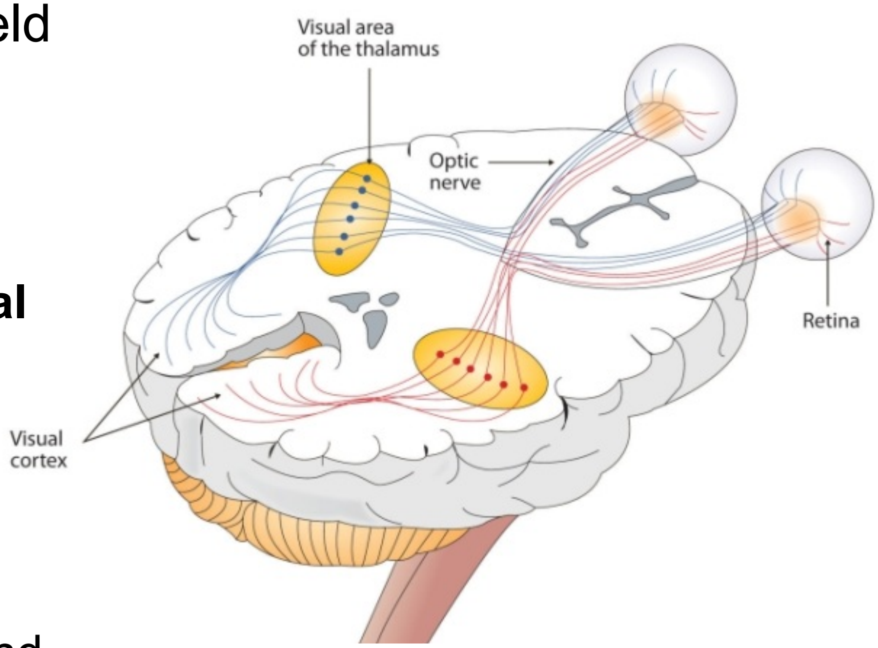
Different DL Models:

- Deep Neural Network
- Deep Boltzmann Machine
- Restricted Boltzmann Machine
- Deep Belief Networks
- Deep Autoencoders
- Recurrent Neural Networks
- Convolutional Neural Networks



Convolutional Neural Networks (CNNs)

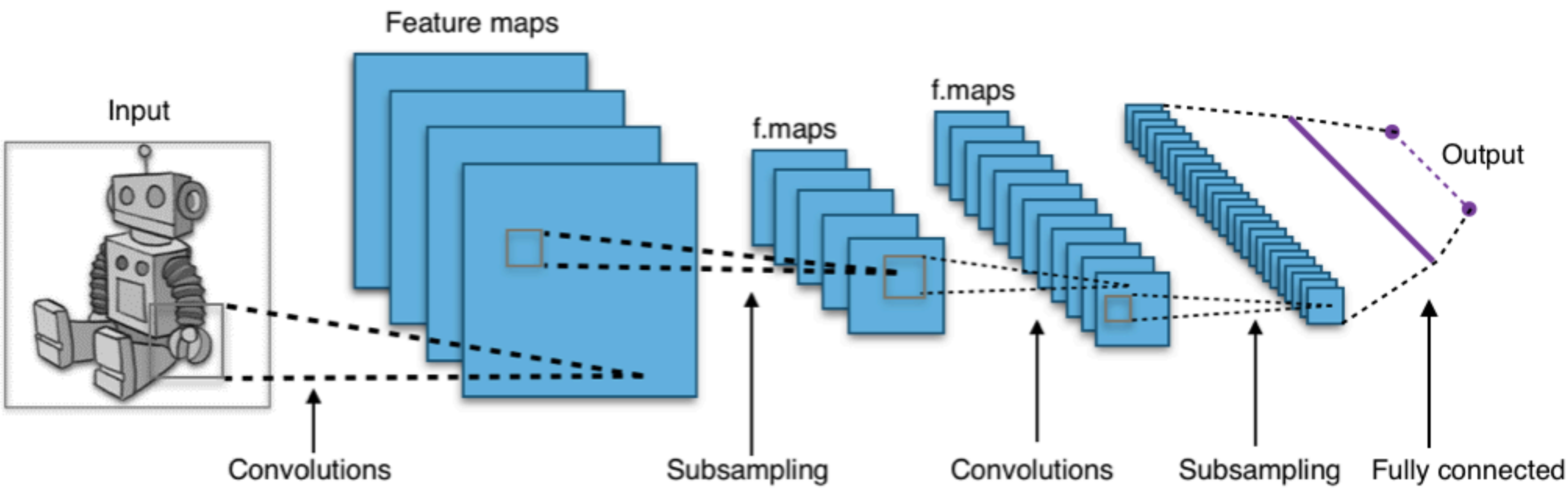
- CNNs take a **biological** inspiration from the visual cortex
- The visual cortex has small regions of cells that are sensitive to **specific regions** of the visual field
  - For example, some neurons fired when exposed to **vertical** edges and some when shown **horizontal** or **diagonal** edges
  - Having the neuronal cells in the visual cortex looking for **specific characteristics** is the basis behind CNNs



Convolutional Neural Networks (CNNs)

Network Architecture

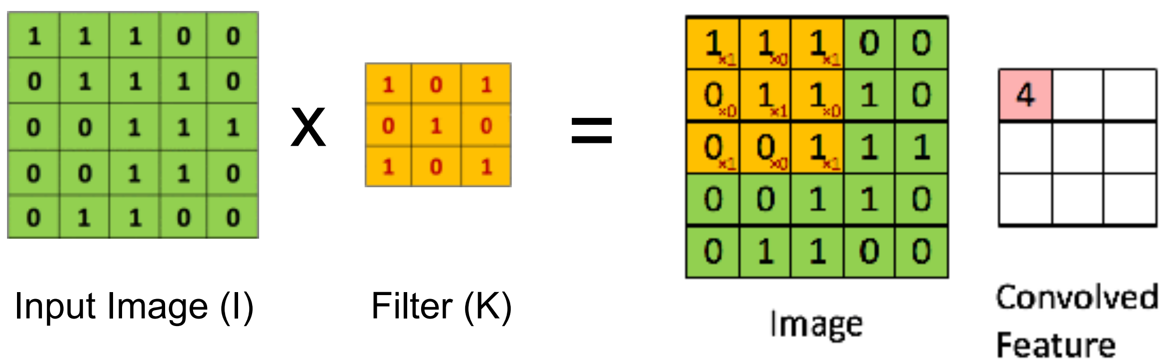
•Convolutional Layer, Pooling Layer, Fully Connected Layer



Convolutional Neural Networks (CNNs)

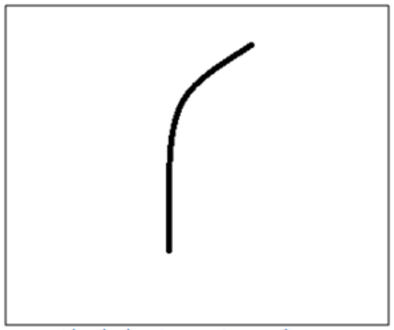
Convolution Operator

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \cdot I_{x+i-1,y-j-1}$$



•The 3×3 matrix (K) is called a '**filter**' or '**kernel**' or '**feature detector**' and the matrix formed by sliding the filter over the image and computing the dot product is called the '**Convolved Feature**' or '**Activation Map**' or the '**Feature Map**'.

Convolutional Neural Networks (CNNs)

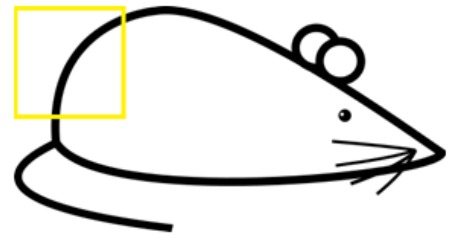


Visualization of a curve detector filter

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Original image



Visualization of the filter on the image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

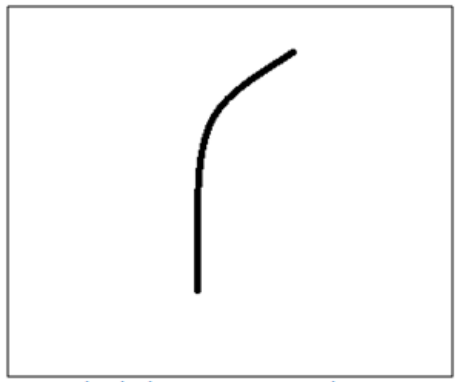
*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Convolutional Neural Networks (CNNs)



Visualization of a curve detector filter

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

Convolutional Neural Networks (CNNs)

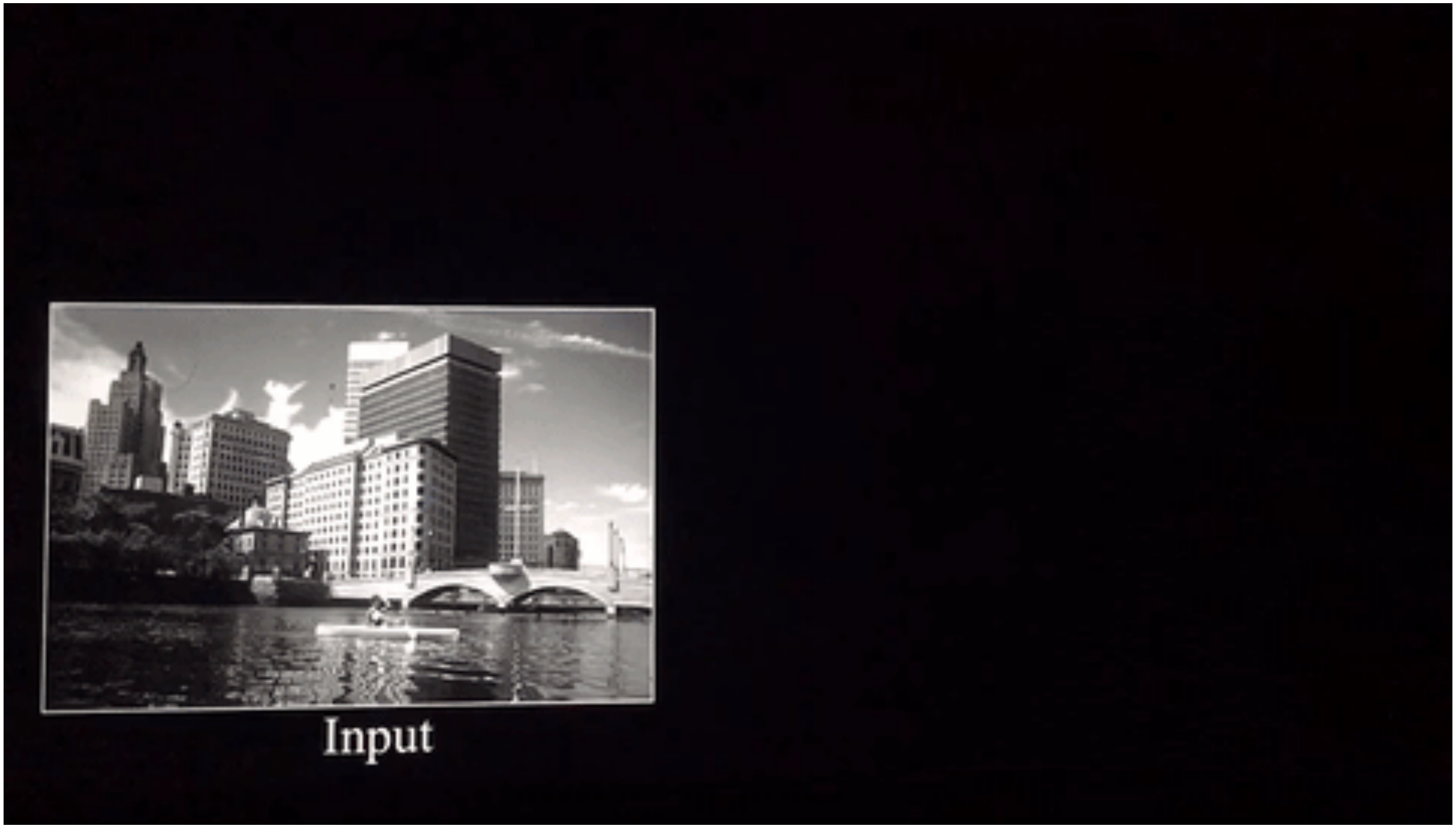
Convolution Operator

•Different filters will produce different **Feature Maps** for the same input image. For example:

Input Image

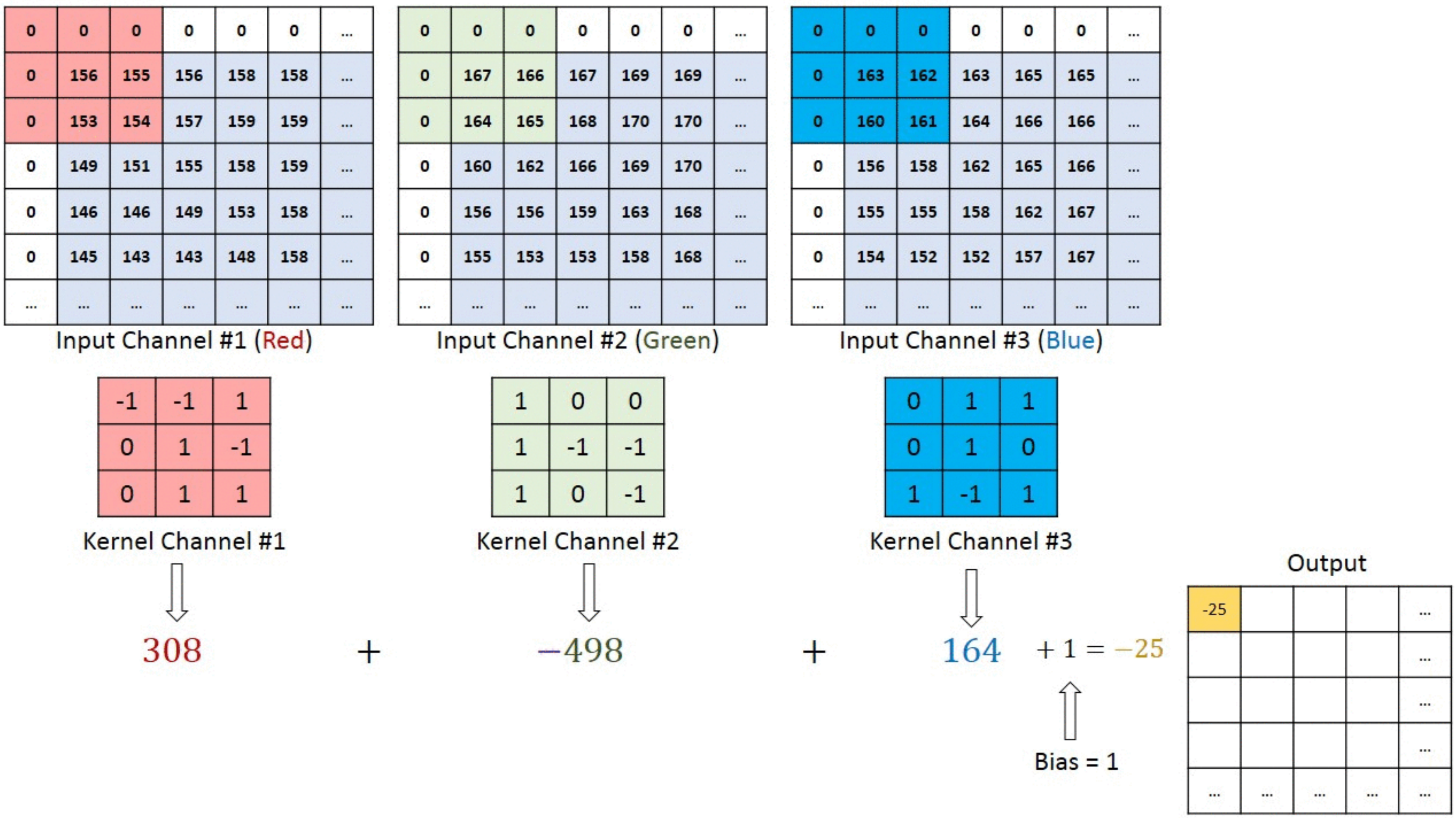
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Convolutional Neural Networks (CNNs)



Convolutional Neural Networks (CNNs)

$$\mathrm{conv}(I,K)_{xy} = \sigma\left(b + \sum_{i=1}^{h}\sum_{j=1}^{w}\sum_{k=1}^{d} K_{ijk} \cdot I_{x+i-1,y+j-1,k}\right)$$

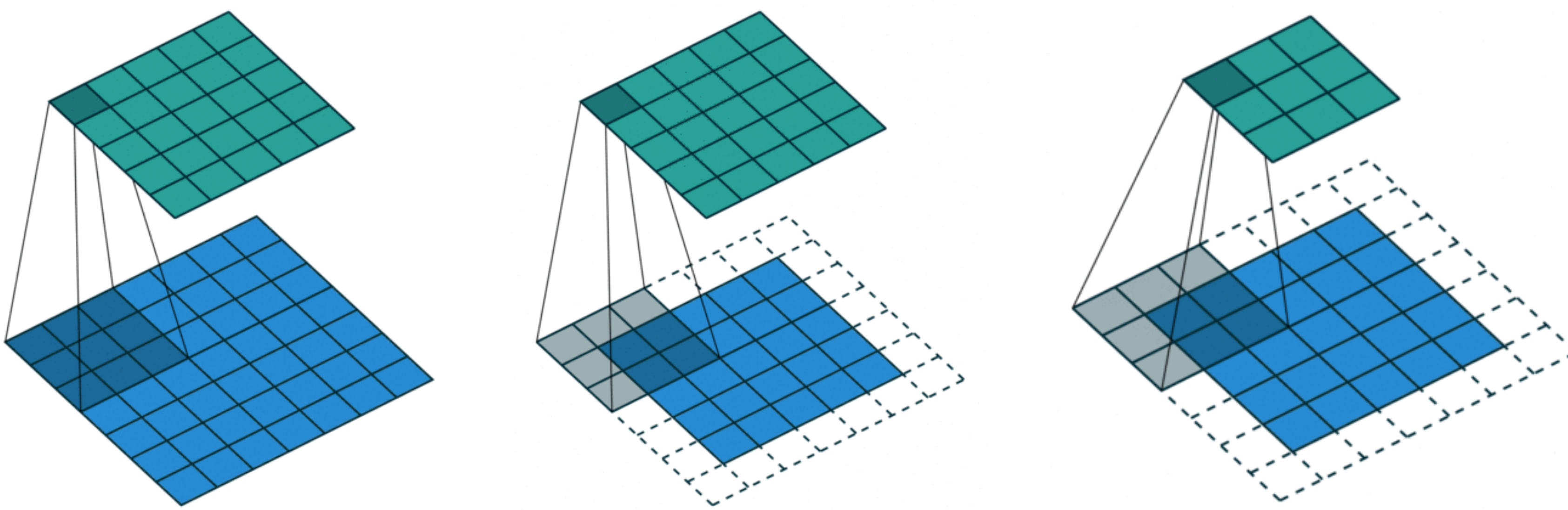


Convolutional Neural Networks (CNNs)

Convolutional Layer

•In practice, a CNN learns the values of these **filters** on its own during the **training process**

•Although we still need to specify parameters such as **number of filters**, **filter size**, **padding,** and **stride** before the training process

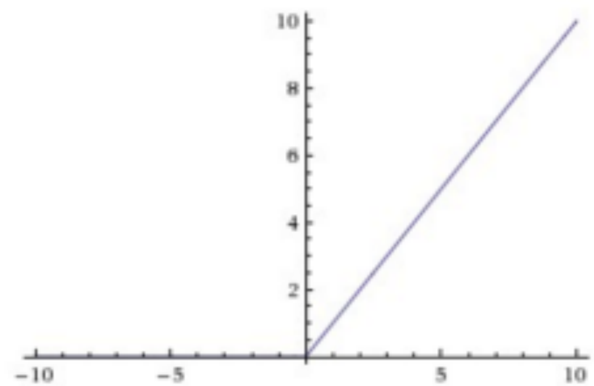


Convolutional Neural Networks (CNNs)

Activation Layer (ReLU)

•An additional operation called Rectified Linear Unit (ReLU) has been used after every Convolution operation

Output = Max(zero, Input)

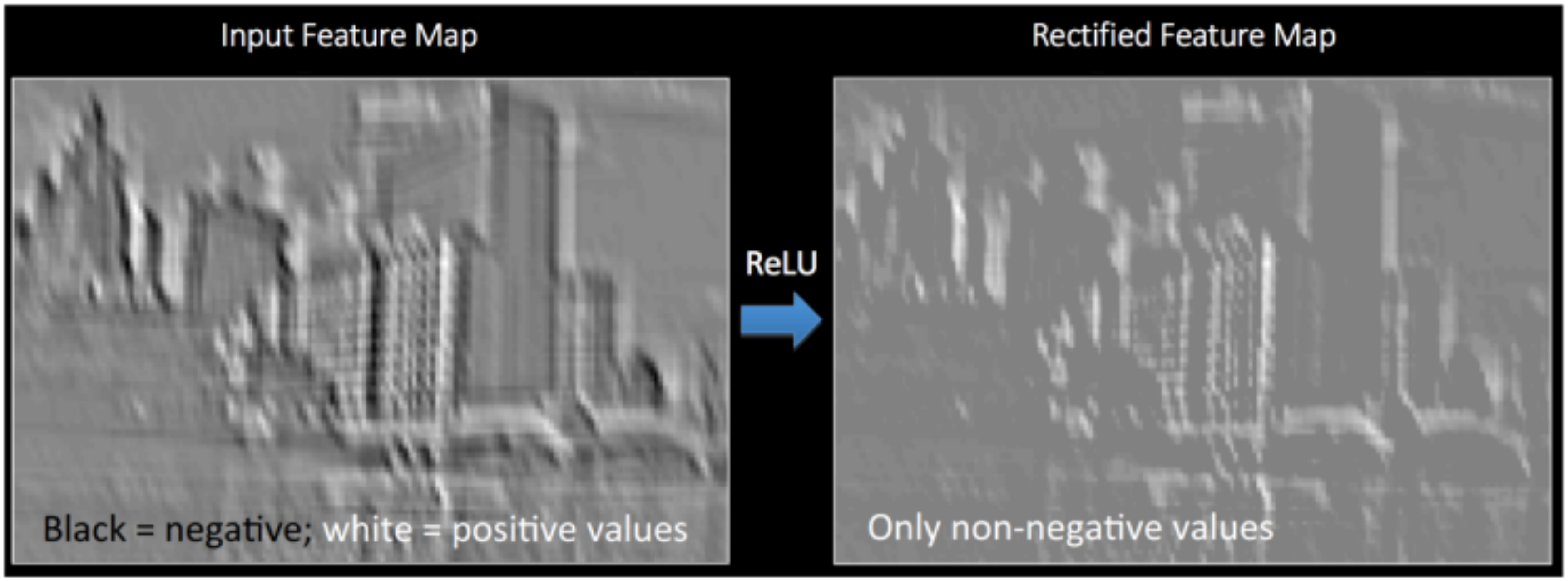


•Basically, ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero

•The purpose of ReLU is to introduce non-linearity to the network

Convolutional Neural Networks (CNNs)

Activation Layer (ReLU)

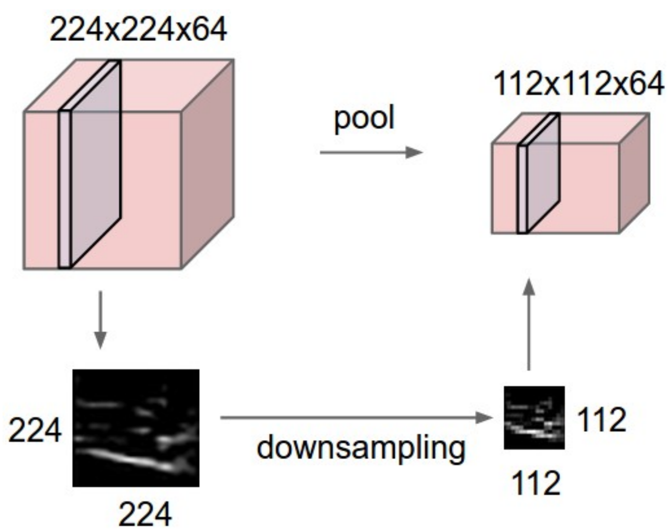


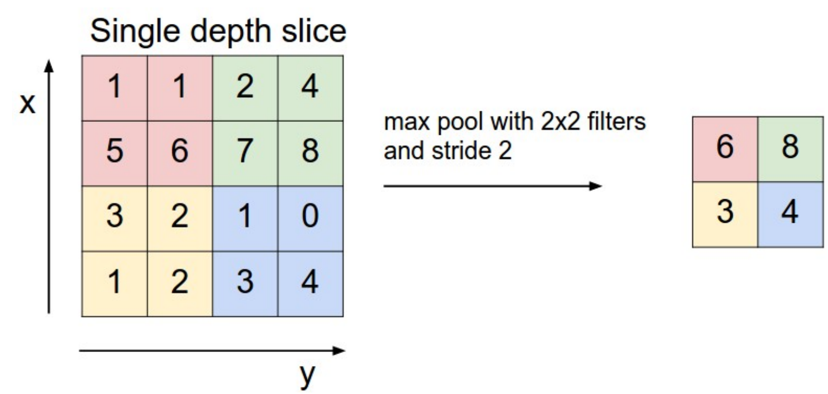
•Other non linear functions such as ***tanh*** or ***sigmoid*** can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

Convolutional Neural Networks (CNNs)

Pooling Layer

•Pooling layer **downsamples** the volume spatially, **independently** in **each depth** slice of the input



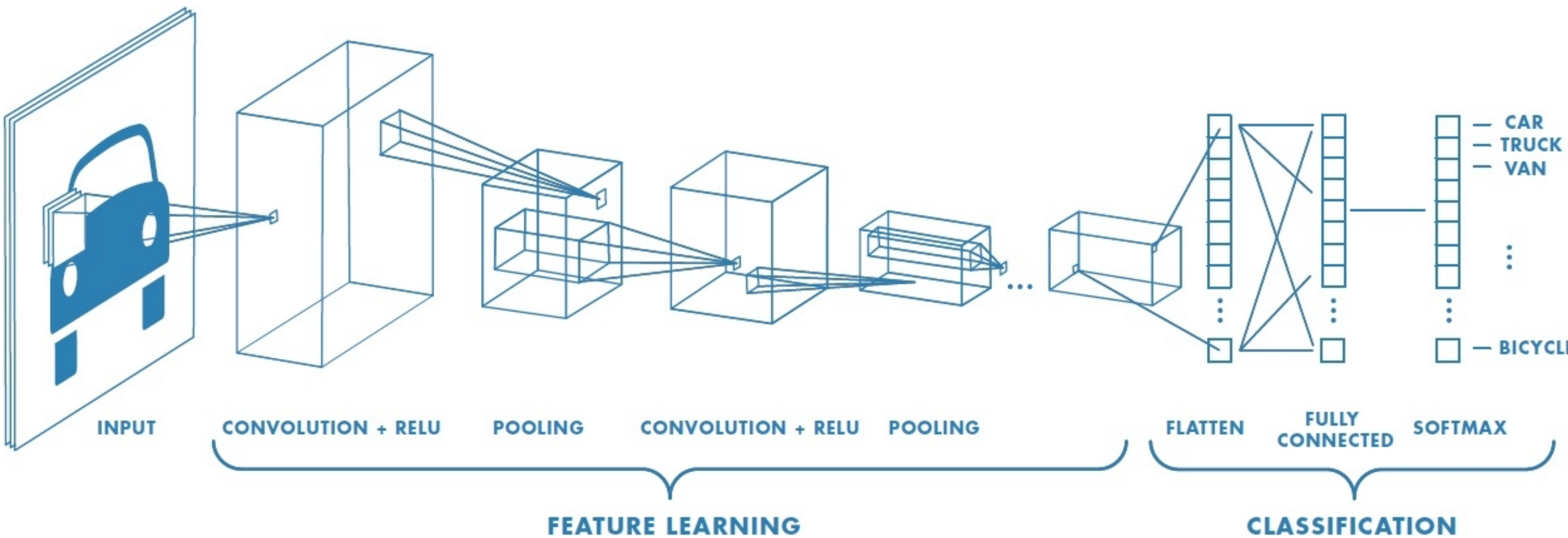


•The most common downsampling operation is **max**, giving rise to **max pooling**, here shown with a stride of 2

Convolutional Neural Networks (CNNs)

Fully Connected Layer

•Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular neural networks



Convolutional Neural Networks (CNNs)

Architectures

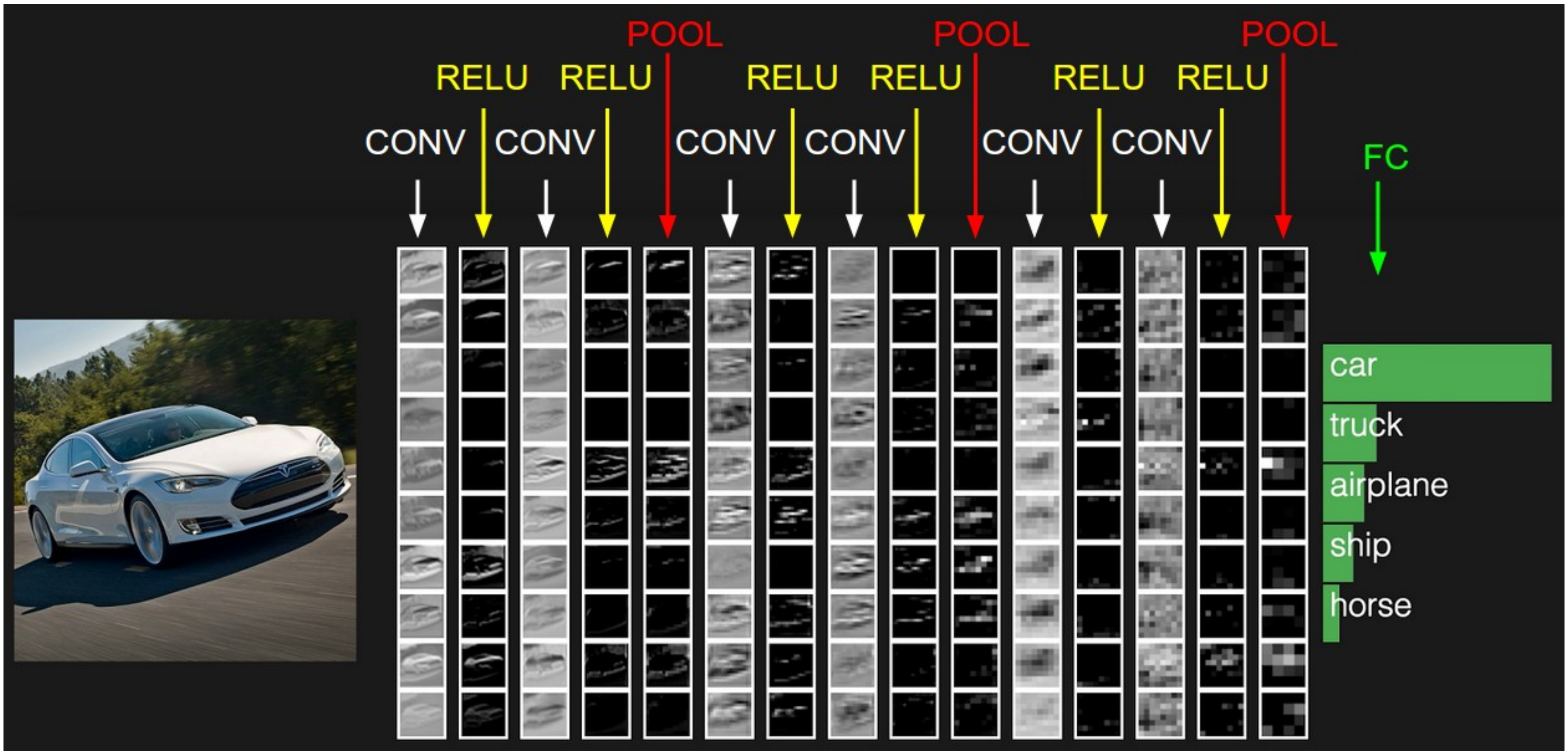
`INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC`

where the `*` indicates repetition, and the `POOL?` indicates an optional pooling layer. Moreover, `N >= 0` (and usually `N <= 3`), `M >= 0`, `K >= 0` (and usually `K < 3`). For example, here are some common ConvNet architectures you may see that follow this pattern:

- `INPUT -> FC`, implements a linear classifier. Here `N = M = K = 0`.
- `INPUT -> CONV -> RELU -> FC`
- `INPUT -> [CONV -> RELU -> POOL]*2 -> FC -> RELU -> FC`. Here we see that there is a single CONV layer between every POOL layer.
- `INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*3 -> [FC -> RELU]*2 -> FC` Here we see two CONV layers stacked before every POOL layer. This is generally a good idea for larger and deeper networks, because multiple stacked CONV layers can develop more complex features of the input volume before the destructive pooling operation.

Convolutional Neural Networks (CNNs)

Example: Input >> [[Conv >> ReLU] * 2 >> Pool] * 3 >> FC



Convolutional Neural Networks (CNNs)

In summary:

•A CNN is in the simplest case a **list of Layers that transform** the image volume into an output volume (e.g. class scores)

•There are a few distinct **types** of Layers

(e.g. CONV/RELU/POOL/FC are by far the most popular)

•Each Layer **may or may not** have parameters

(e.g. CONV/FC do, RELU/POOL don’t)

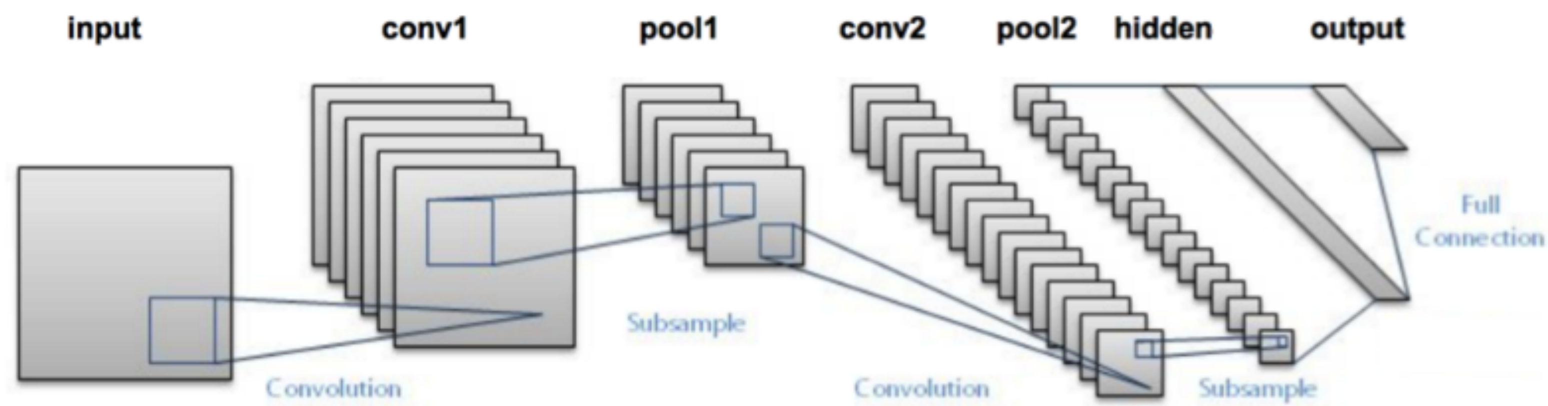
•Each Layer **may or may not** have additional hyperparameters

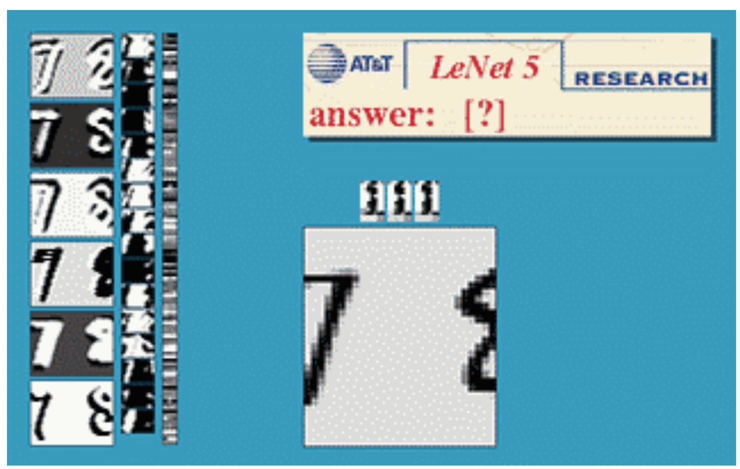
(e.g. CONV/FC/POOL do, RELU doesn’t)

Successful CNN architectures

LeNet-5

•This architecture is an excellent "first architecture" for a CNN

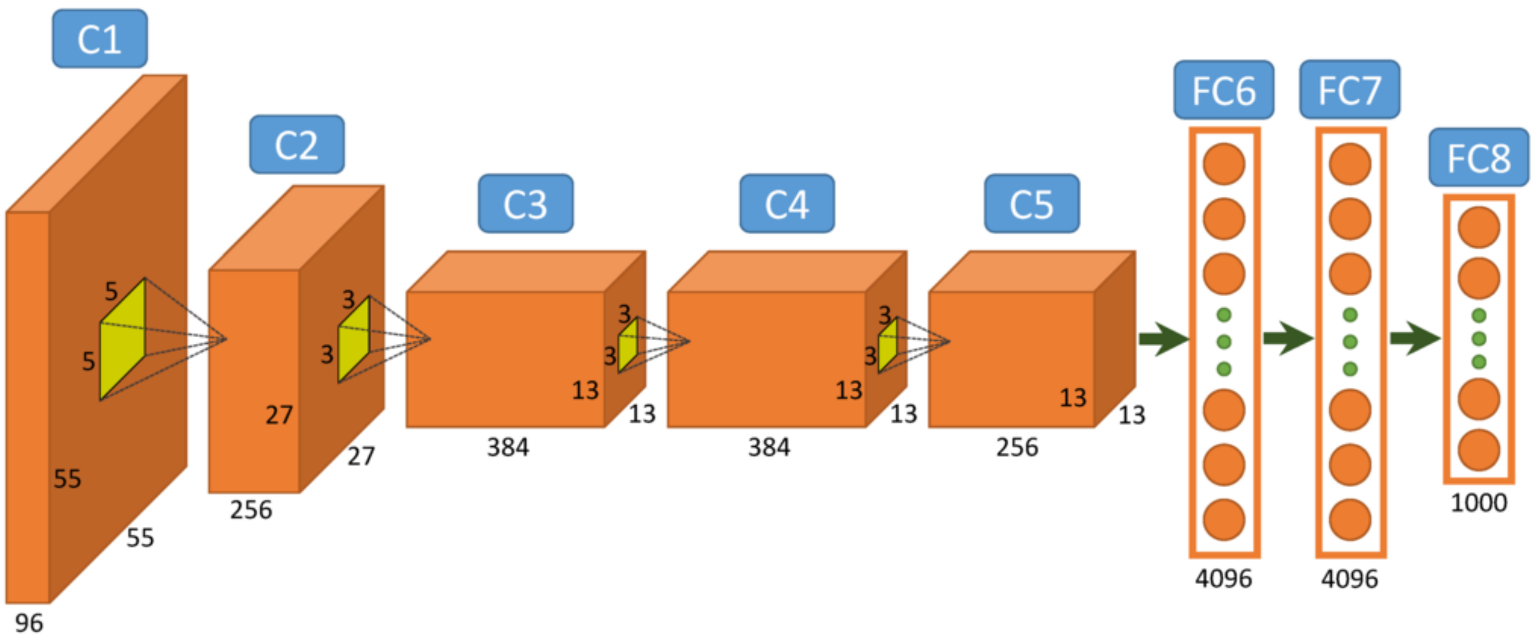




Successful CNN architectures

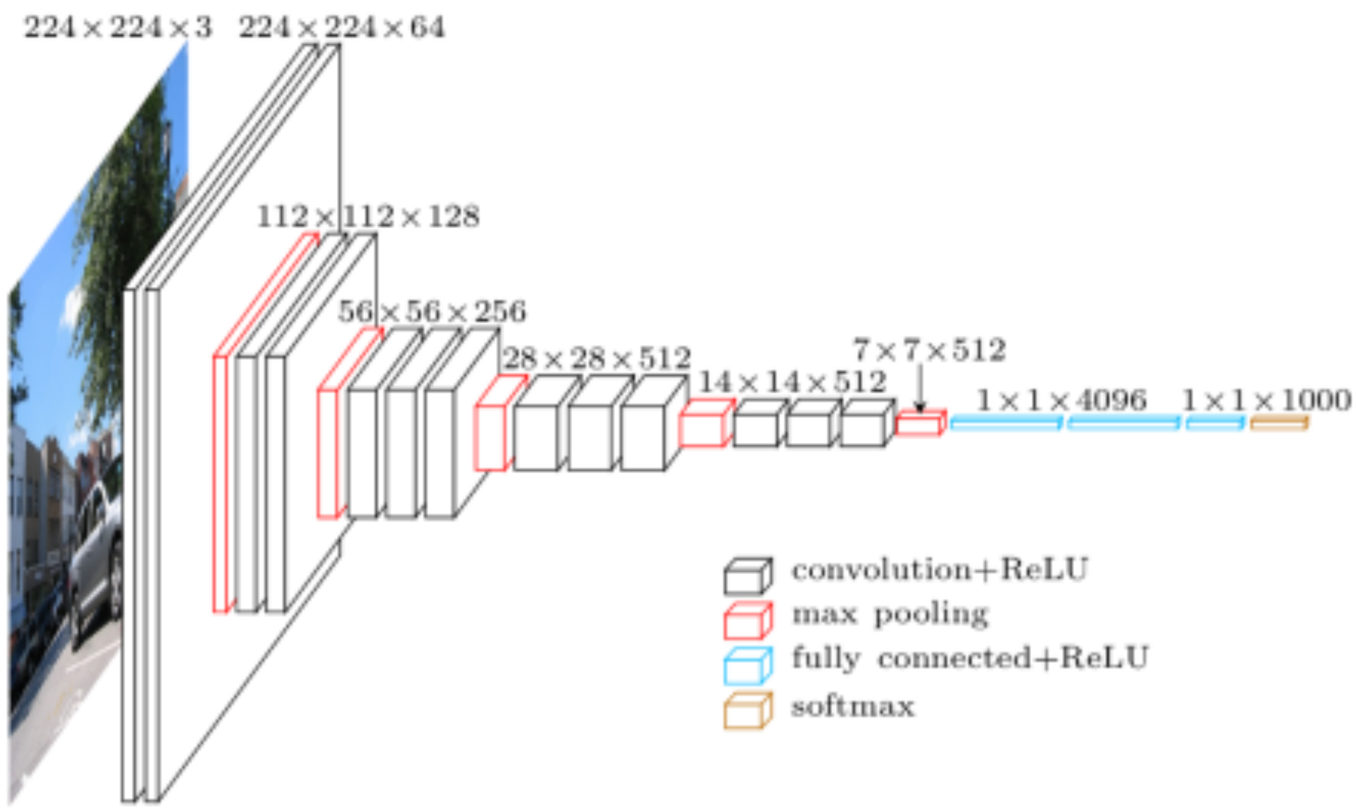
AlexNet

•Famous for winning the **ImageNet** Large Scale Visual Recognition Challenge (**ILSVRC**) in 2012



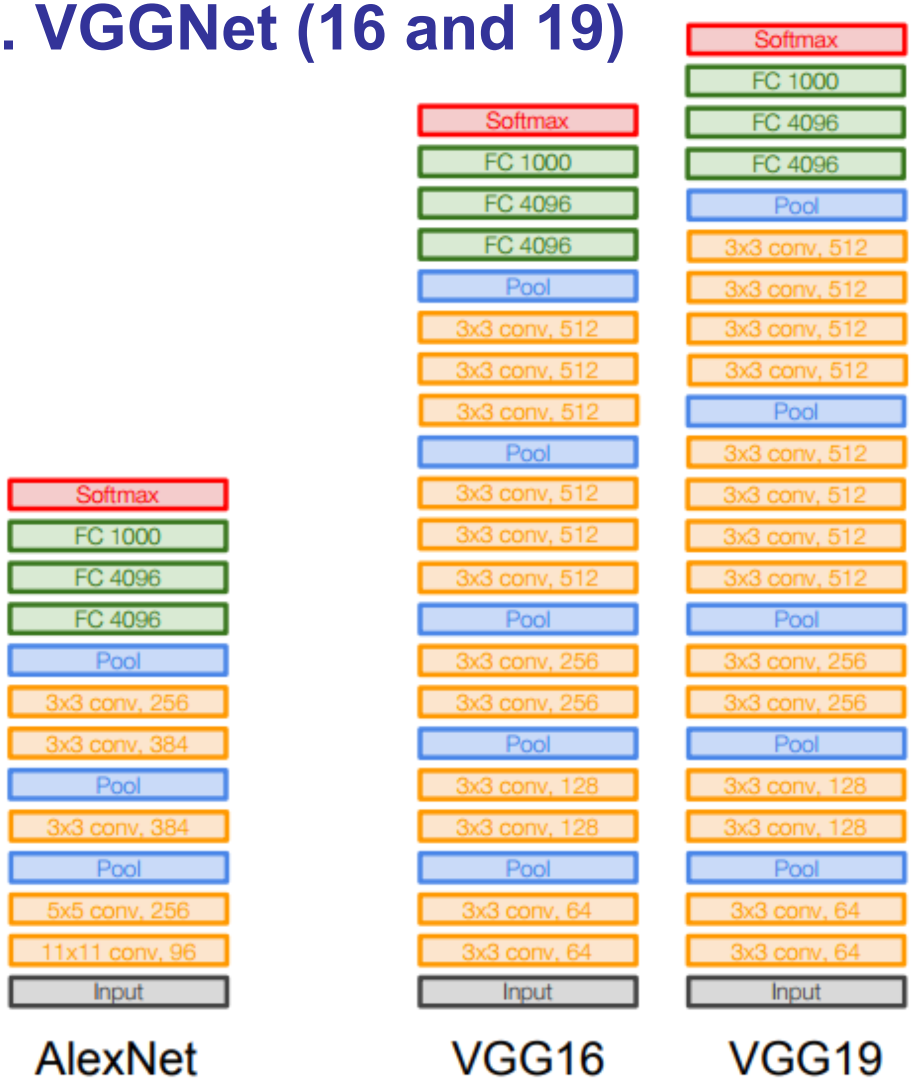
Successful CNN architectures

VGGNet



Successful CNN architectures

AlexNet vs. VGGNet (16 and 19)



Agenda

Introduction

•What we see vs. What computers see (MNIST and CIFAR Datasets)

•Hand-Crafted Features for Image Classification

Deep Learning

•Convolutional Neural Networks (CNNs)

- Architecture (Convolutional, Pooling, and Fully Connected Layers)
- Successful CNN Architectures

Training

•Backpropagation

•Overfitting, Regularization and Dropout

Experiments

Transfer Learning

Complex Networks

Training

Backpropagation

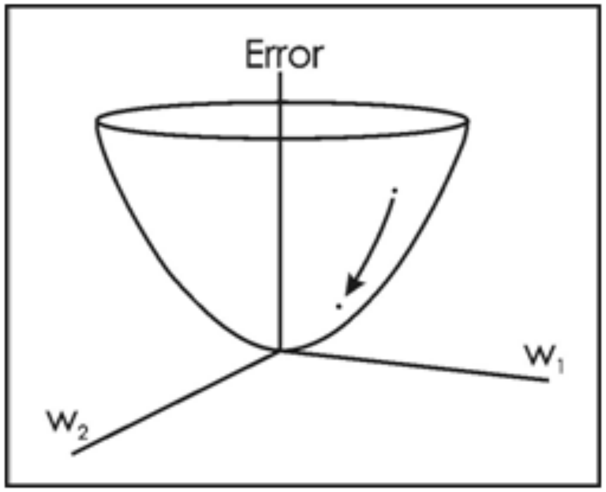
•Algorithm to calculate all weights and biases

•Cost Function

$$L_{total} = \sum \frac{1}{2}(target - output)^2$$

•Minimize gradient of the cost function

- This is the mathematical equivalent of a **dL/dW** where W are the weights at a particular layer



Training

Backpropagation

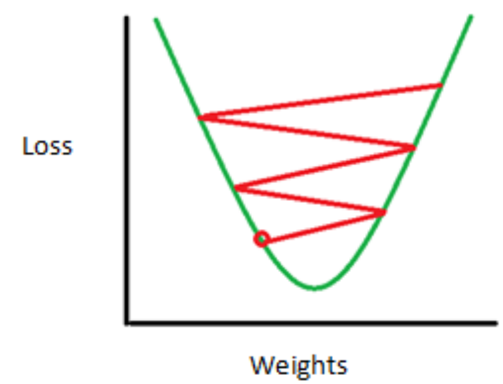
•Weight Updates

$$w = w_i - \eta \frac{dL}{dW}$$

w = Weight
w_i = Initial Weight
η = Learning Rate

•Learning Rate

- Parameter chosen by the programmer
- A high learning rate means that bigger steps are taken in the weight updates
- However, a learning rate that is too high could result in jumps that are too large and not precise enough to reach the optimal point

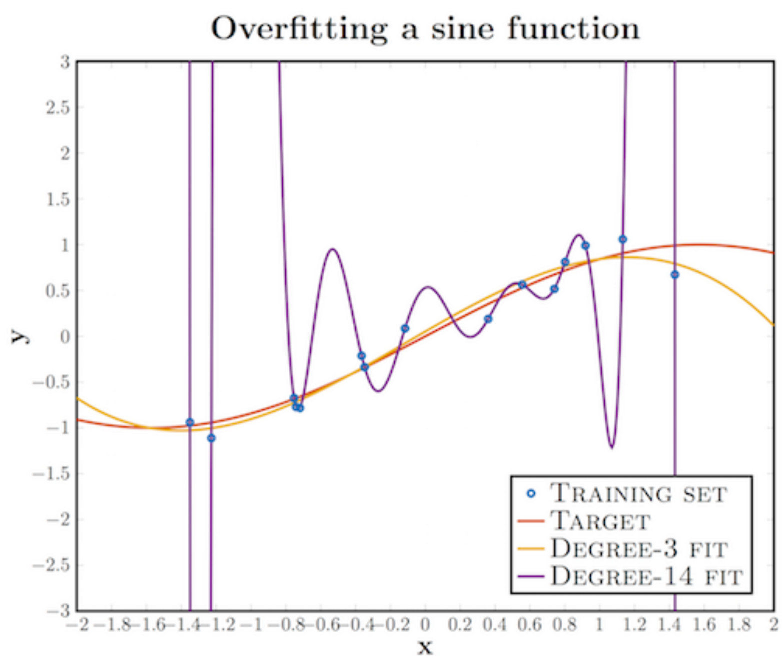


Training

Overfitting

•Our model might have learned the training set (along with any noise present within it) **perfectly**, but it has failed to capture the underlying process that generated it

- On CNNs, overfitting may occur if we **don't have sufficiently** training examples, then a small group of neurons might become responsible for doing most of the processing and other neurons becoming redundant

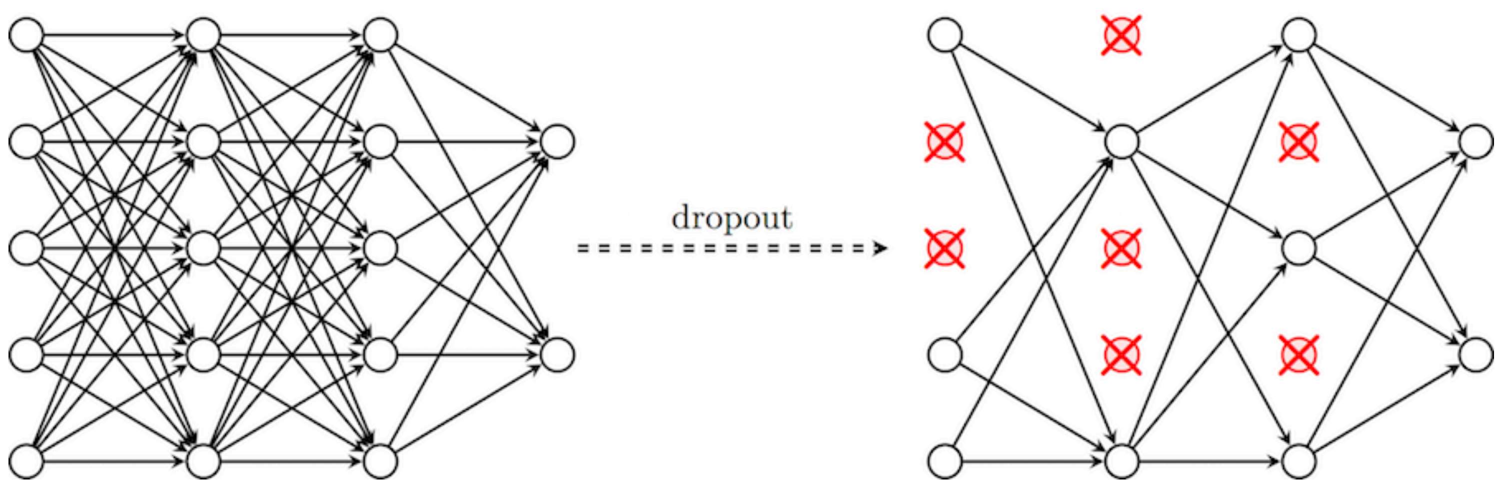


Training

Regularization

•Rather than reducing the number of parameters, for CNNs we impose constraints on the model parameters during training to keep them from learning the noise in the training data

•**Dropout**: This has the effect of forcing the neural network to cope with *failures*, and not to rely on existence of a particular neuron (or set of neurons) – relying more on a ***consensus*** of several neurons within a layer



Agenda

Introduction

•What we see vs. What computers see (MNIST and CIFAR Datasets)

•Hand-Crafted Features for Image Classification

Deep Learning

•Convolutional Neural Networks (CNNs)

- Architecture (Convolutional, Pooling, and Fully Connected Layers)
- Successful CNN Architectures

Training and Testing

•Backpropagation

•Overfitting, Regularization and Dropout

Experiments

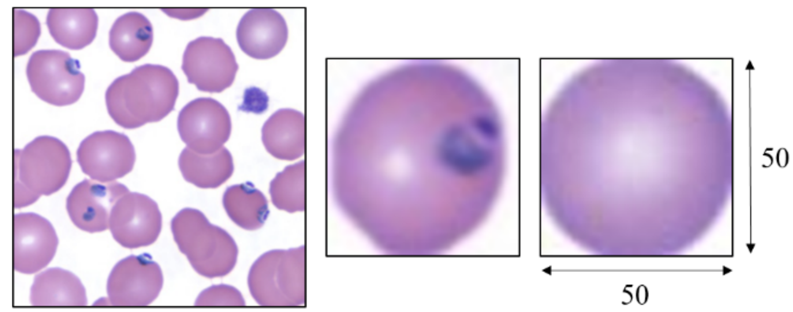
Transfer Learning

Complex Networks

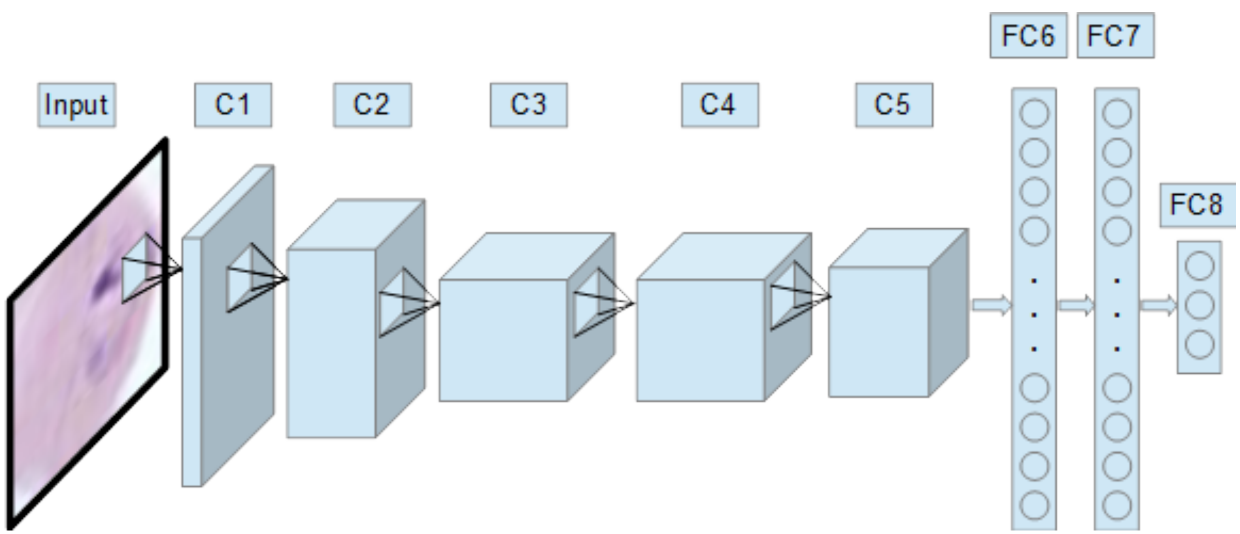
Experiments

Malaria Recognition

•Training Dataset:



•Adapted **AlexNet**:

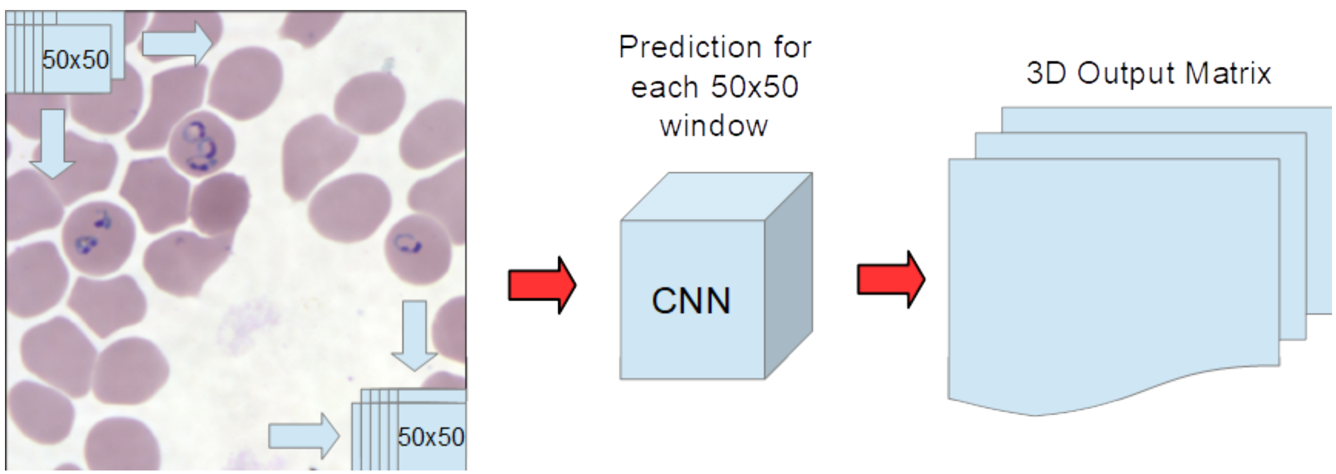


Experiments

Malaria Recognition

•Feature Maps Learned:

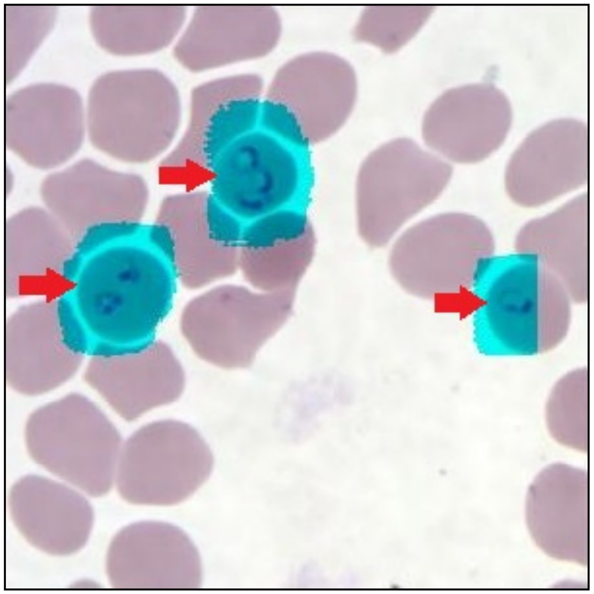
•Thin Blood Smear Analysis Framework:

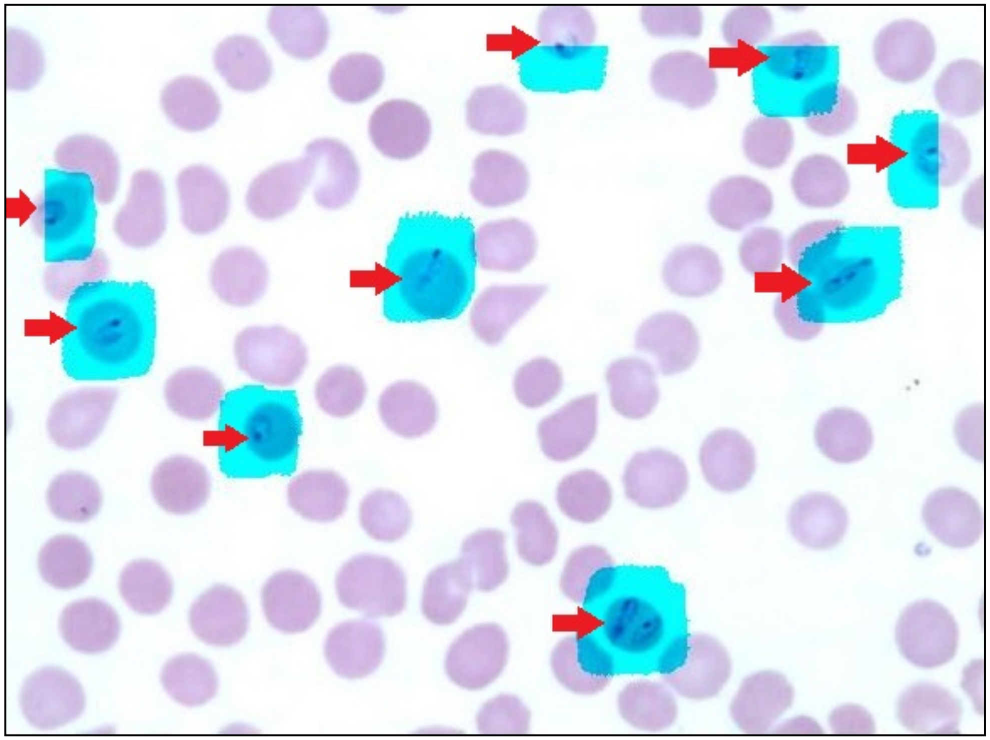


Experiments

Malaria Recognition

•Results:





Experiments

Plant Recognition

•Dataset ➔ Plants in Natural Images (natural background)

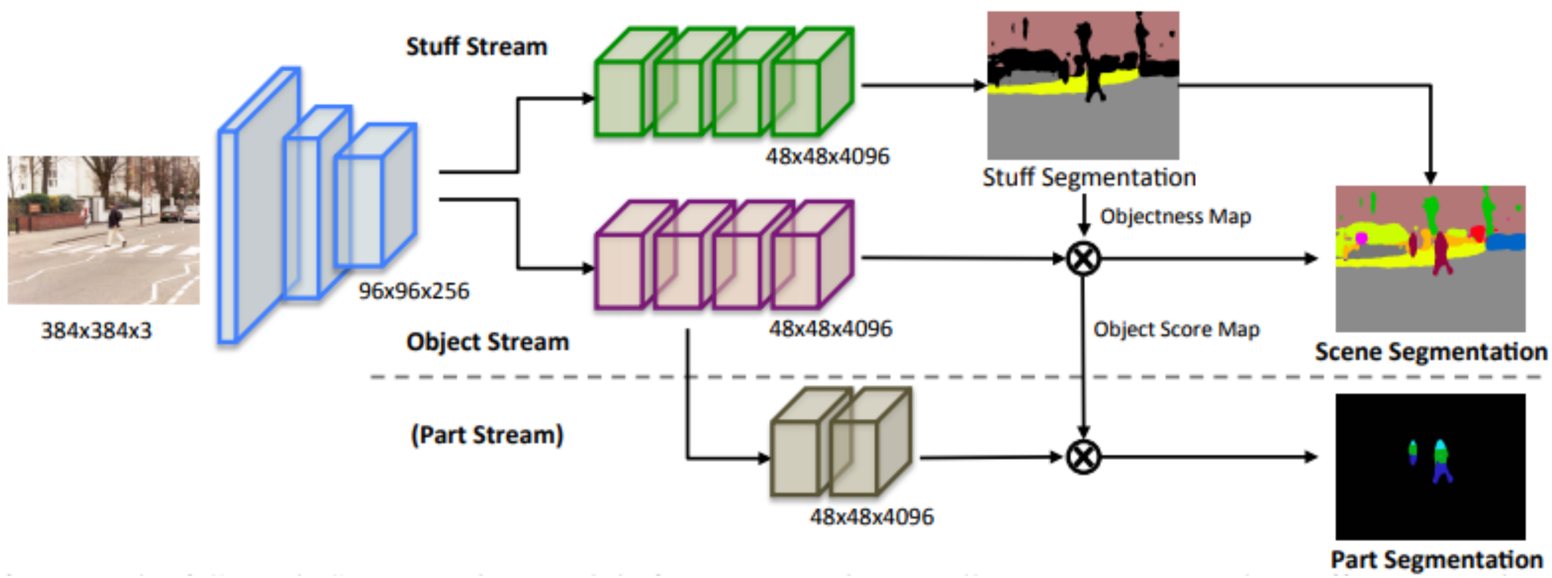
•Step 1: Segmentation

- Using **MIT Scene Parsing**, pre-trained model (ADE20K dataset)

Experiments

Plant Recognition

•MIT Scene Parsing ➔ Stacked CNNs



Experiments

Plant Recognition

•Initial Results:

Experiments

Plant Recognition

•Initial Results:

Agenda

Introduction

- What we see vs. What computers see (MNIST and CIFAR Datasets)
- Hand-Crafted Features for Image Classification

Deep Learning

- Convolutional Neural Networks (CNNs)
  - Architecture (Convolutional, Pooling, and Fully Connected Layers)
  - Successful CNN Architectures

Training

- Backpropagation
- Overfitting, Regularization and Dropout

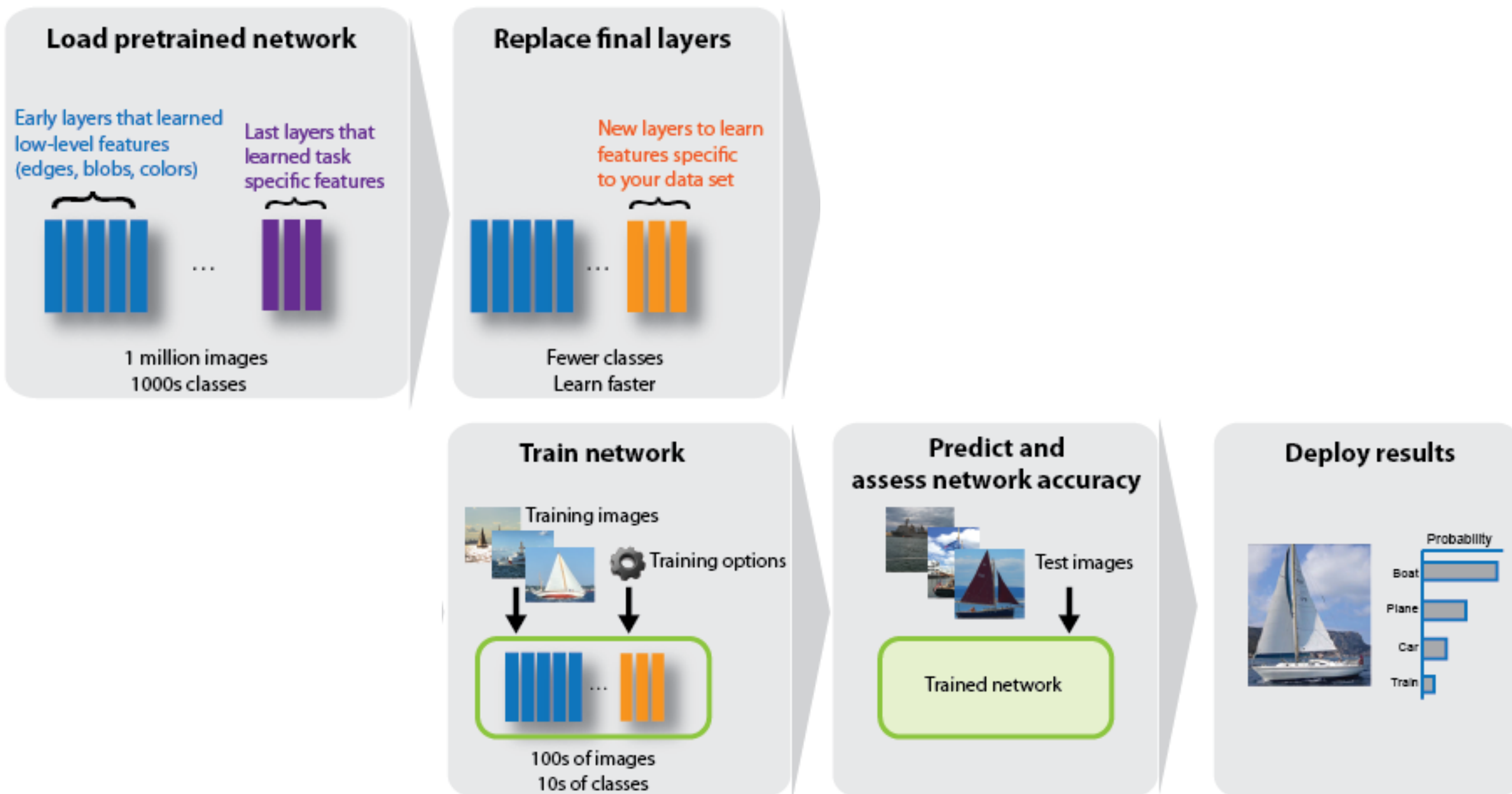
Experiments

$\boxed{\text{Transfer Learning}}$

Complex Networks

Transfer Learning

- Most pre-trained models used the ImageNet Dataset (1 Million of images and 1,000 Classes)



Agenda

Introduction

- What we see vs. What computers see (MNIST and CIFAR Datasets)
- Hand-Crafted Features for Image Classification

Deep Learning

- Convolutional Neural Networks (CNNs)
  - Architecture (Convolutional, Pooling, and Fully Connected Layers)
  - Successful CNN Architectures

Training

- Backpropagation
- Overfitting, Regularization and Dropout

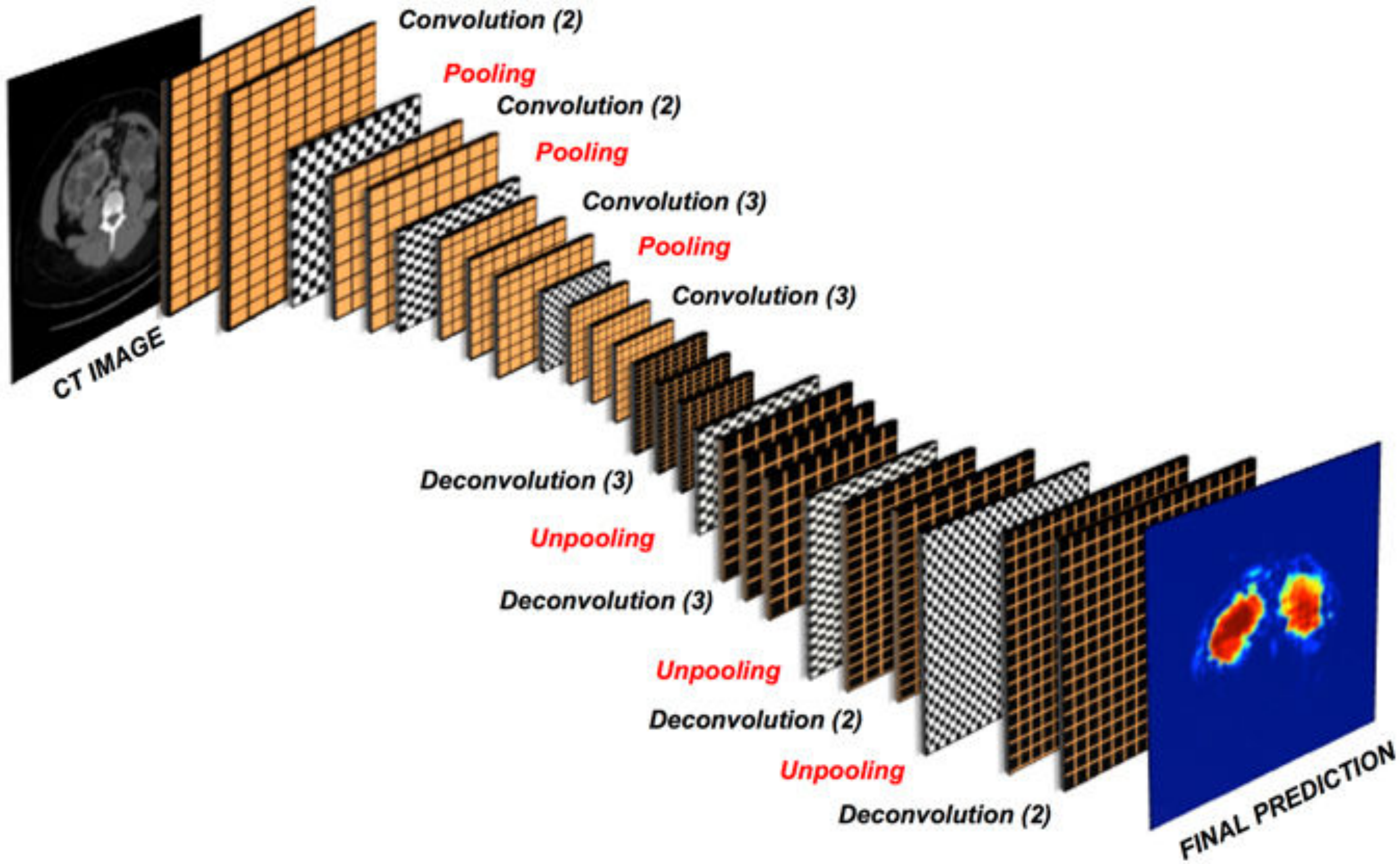
Experiments

Transfer Learning

Complex Networks

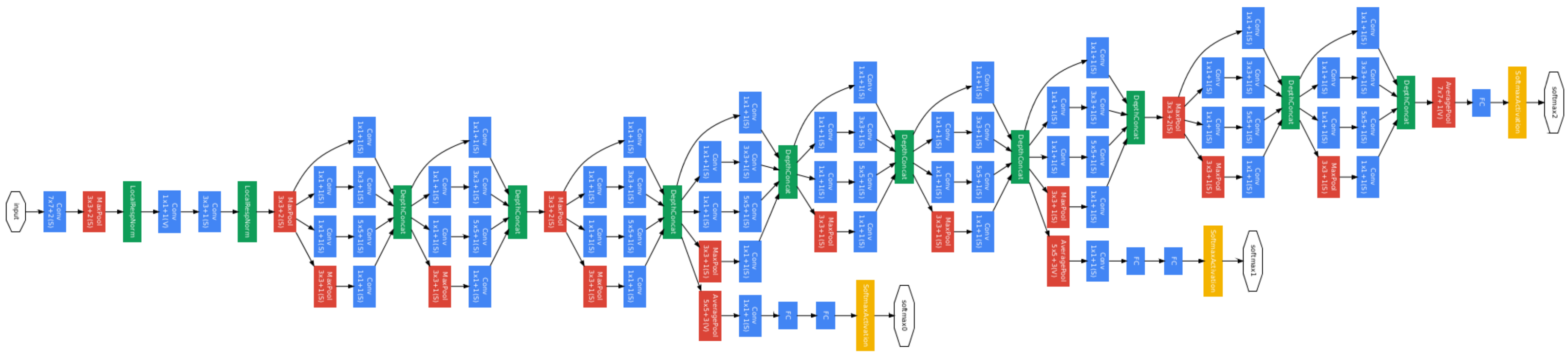
Complex Networks

Deconvolutional Neural Networks (DCNN)



Complex Networks

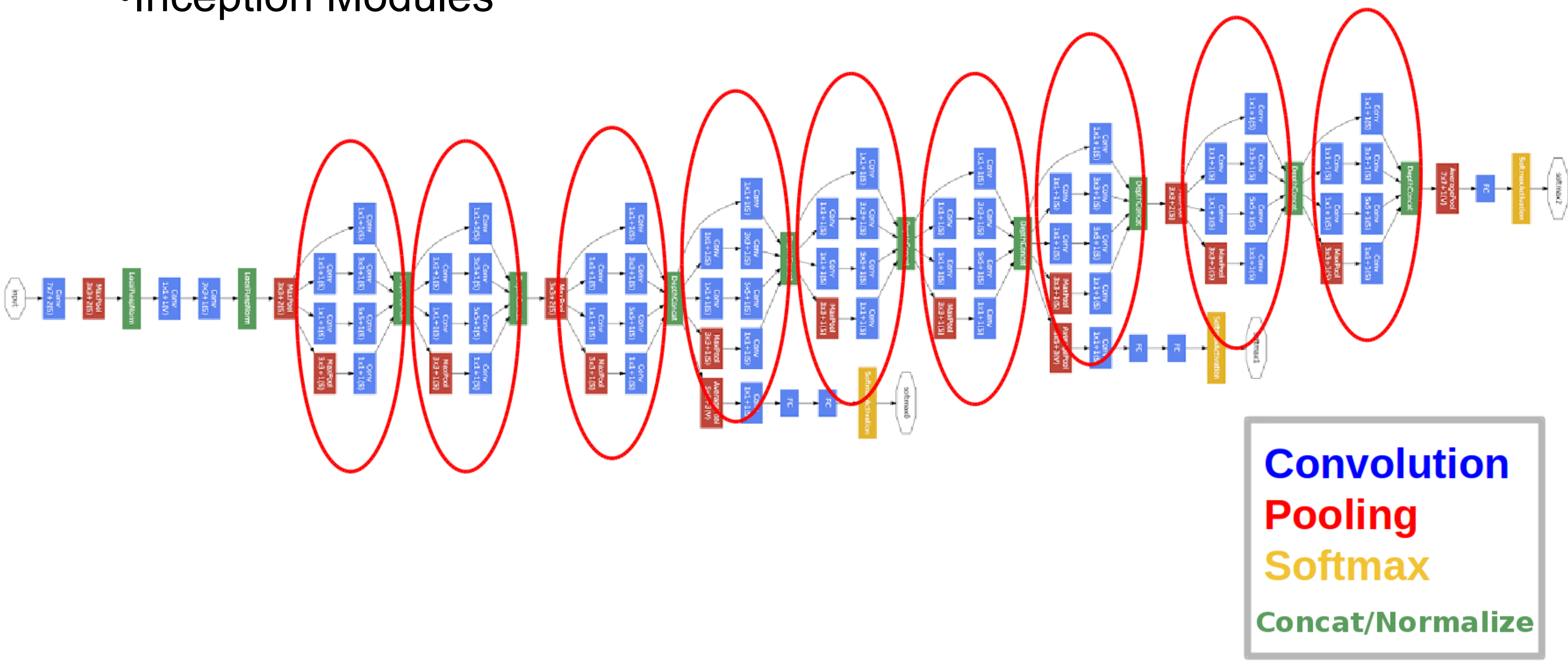
GoogLeNet (ILSVRC 2014 winner)



Complex Networks

GoogLeNet

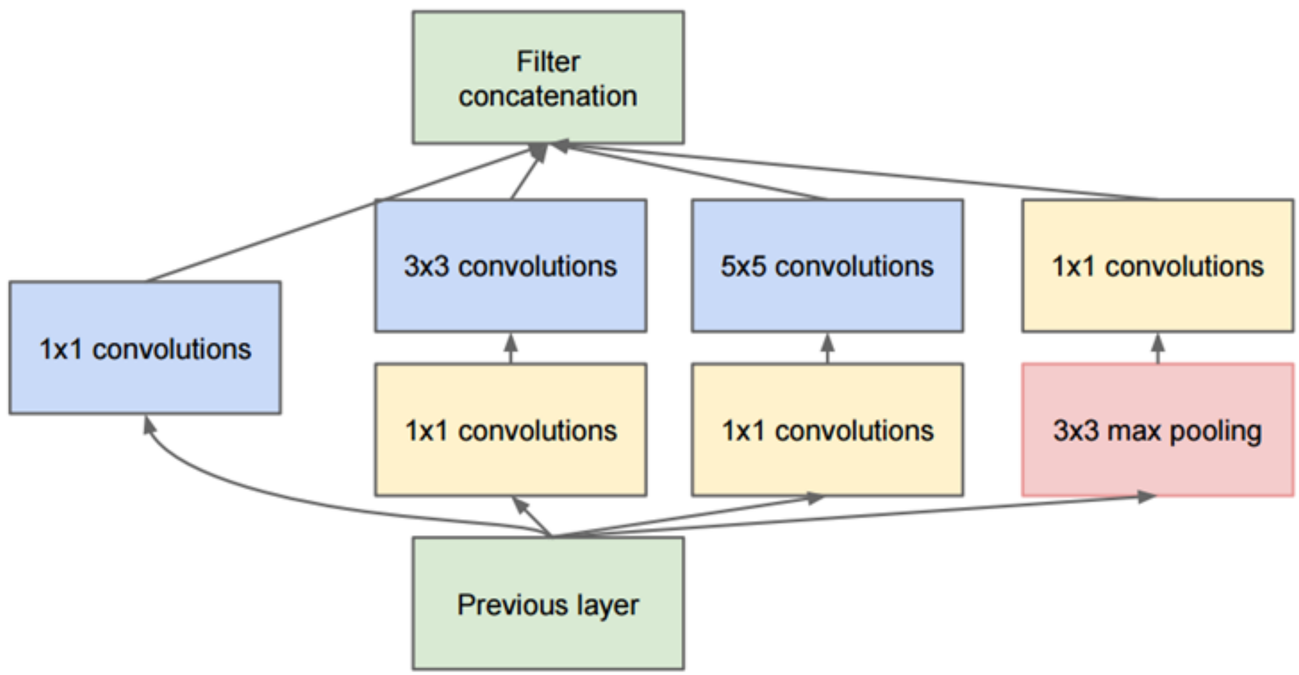
- Inception Modules



Complex Networks

GoogLeNet

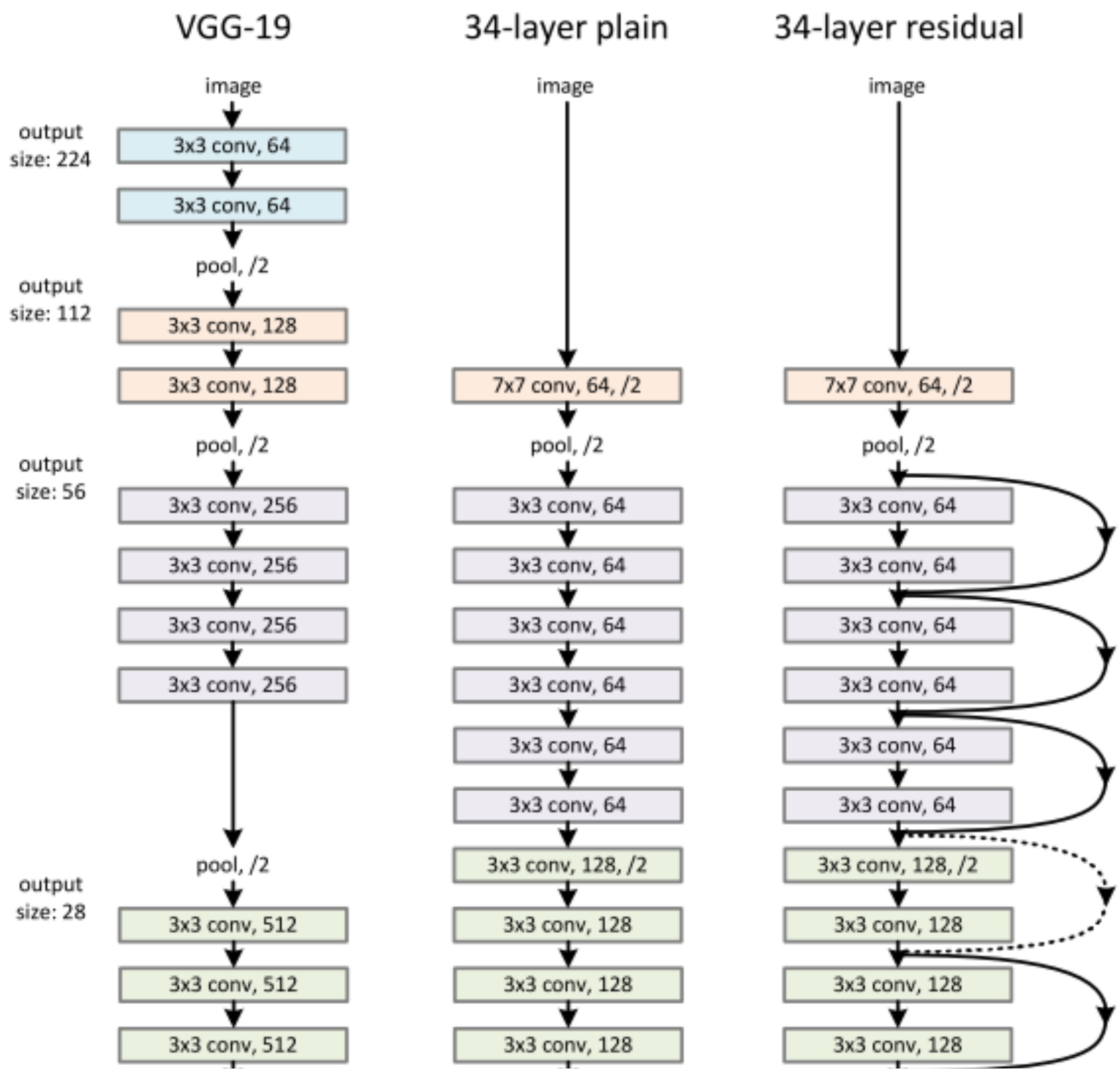
•Inception Modules (Network inside a network)

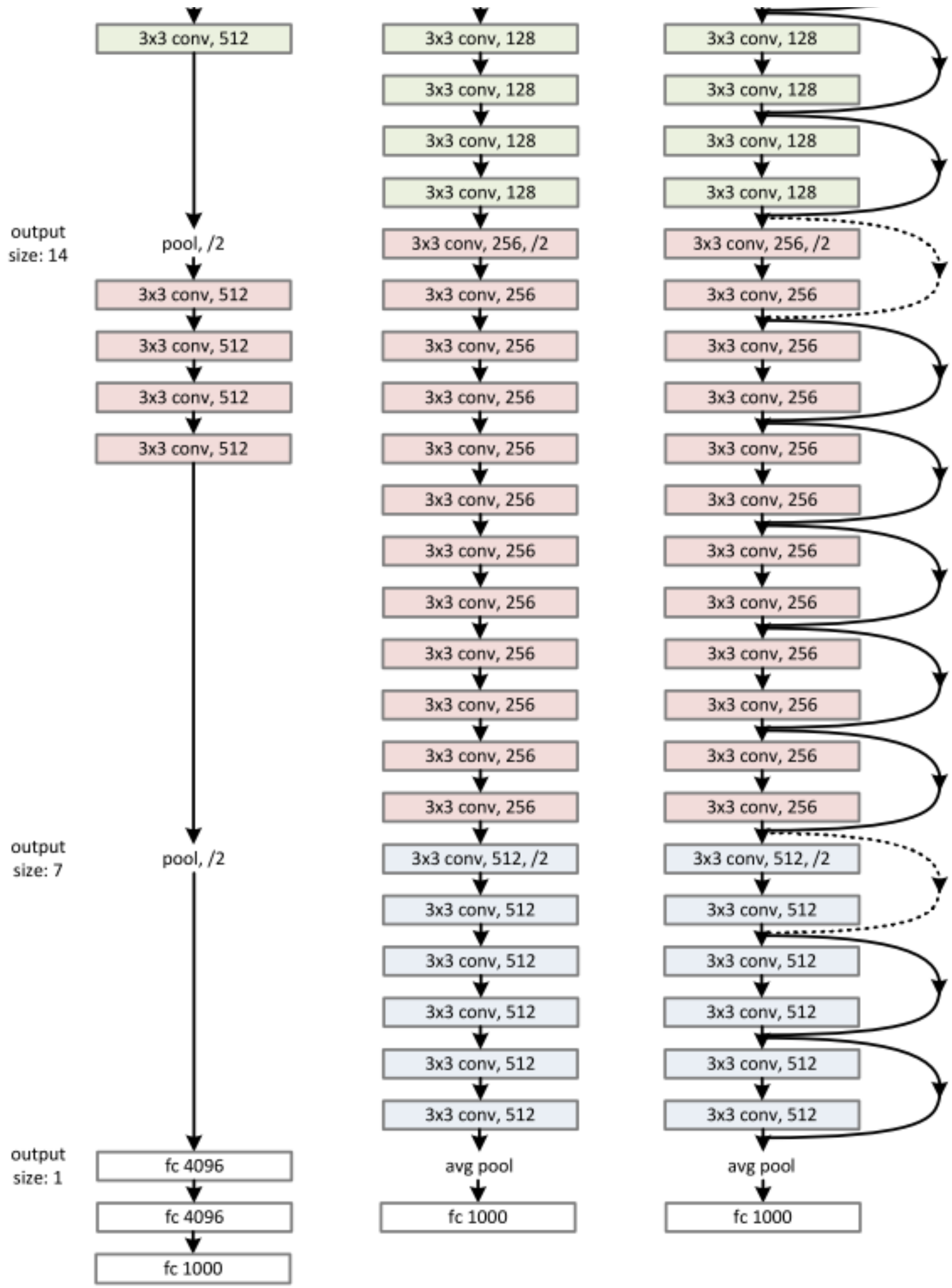


Full Inception module

Complex Networks

ResNet (ILSVRC 2015 winner)



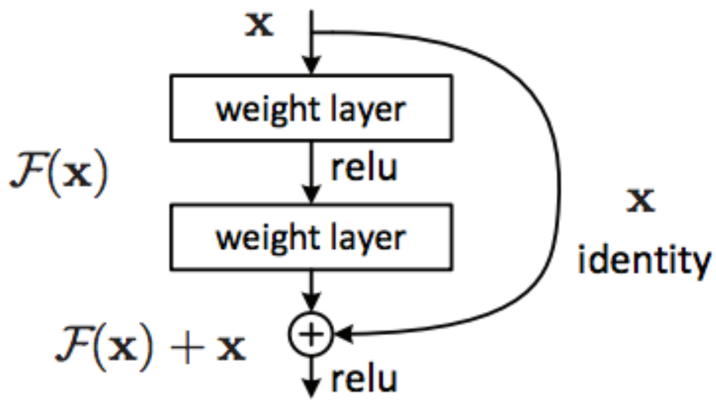


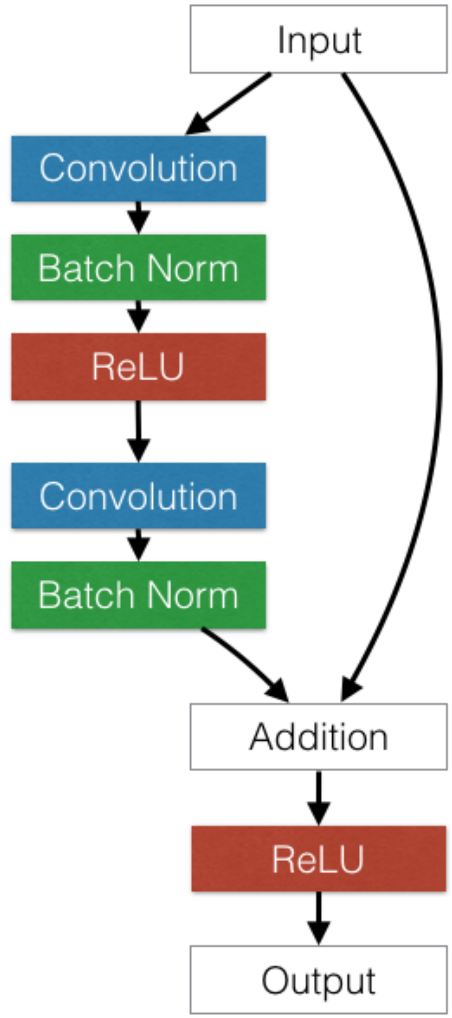
3x3 conv, 512
output size: 14
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
output size: 7
pool, /2
output size: 1
fc 4096
fc 4096
fc 1000
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Complex Networks

ResNet

•Residual Block

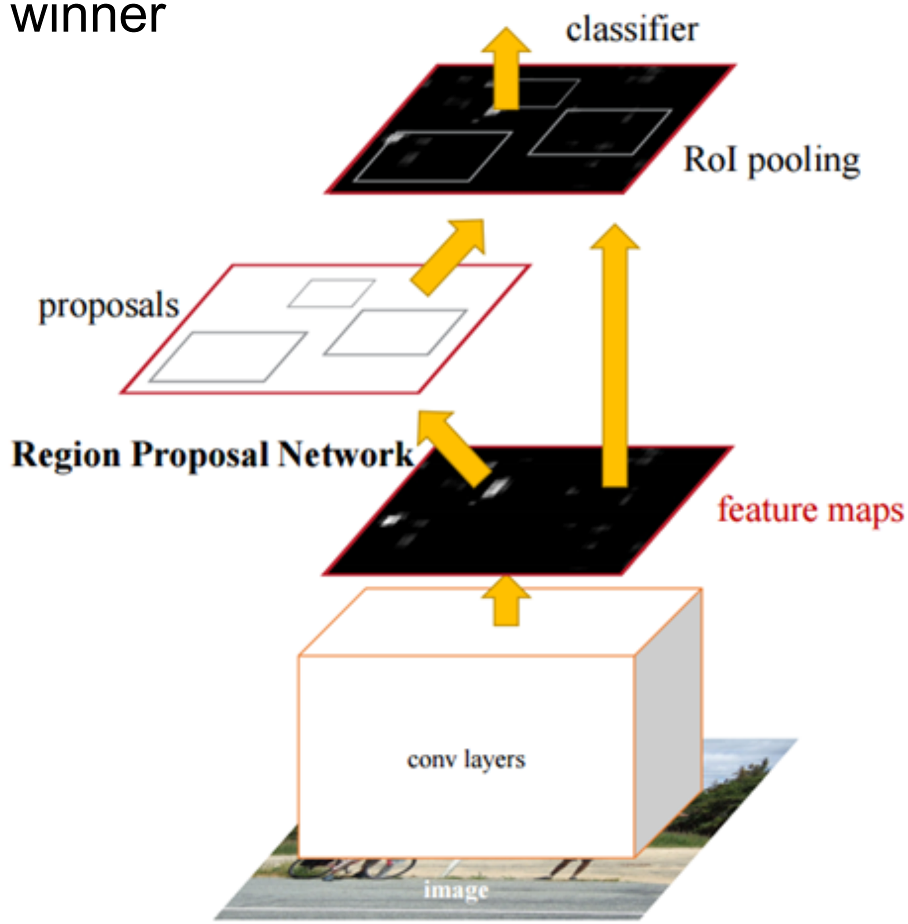




Complex Networks

CUImage (Fast Region-based CNN)

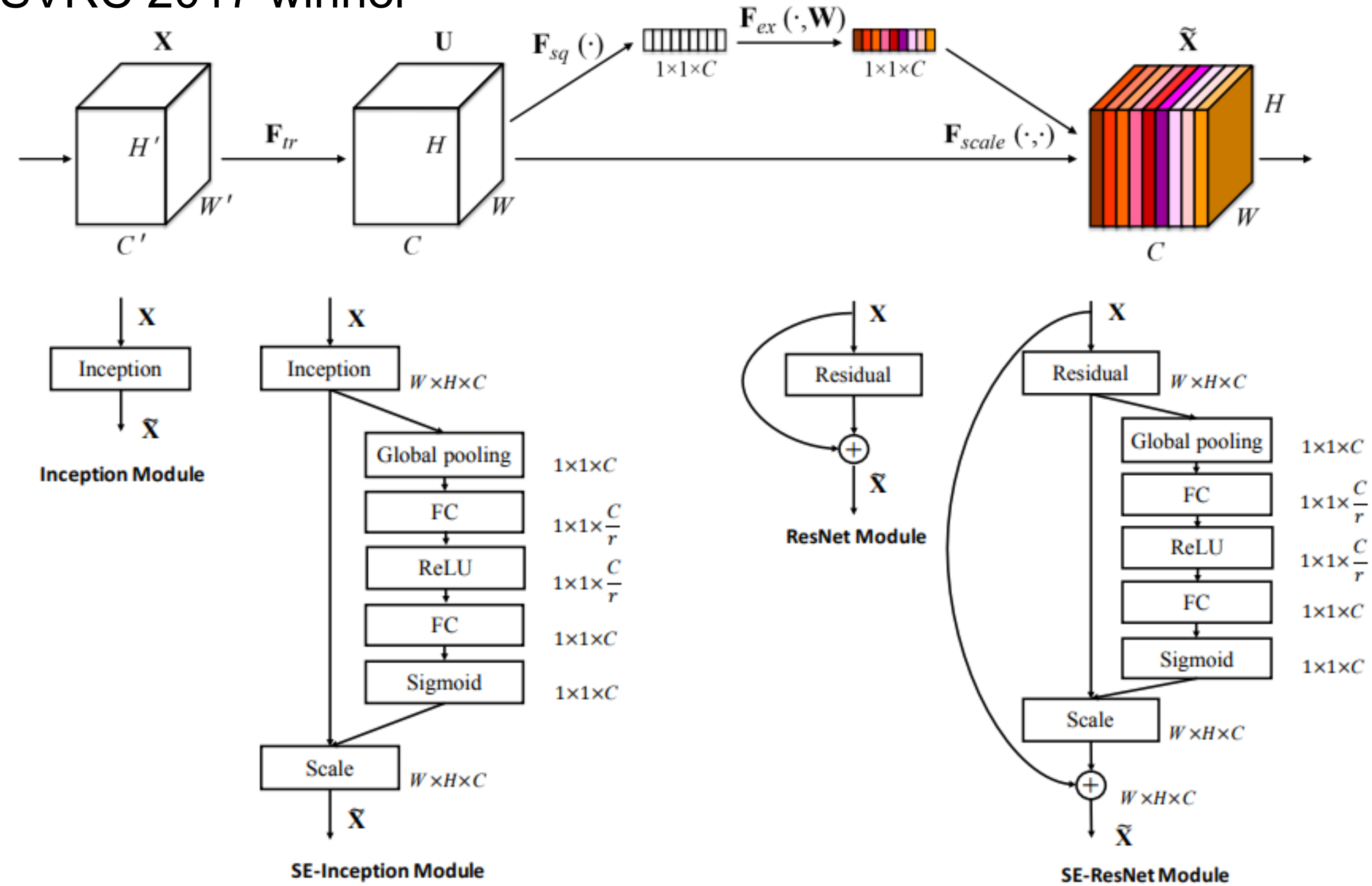
- ILSVRC 2016 winner



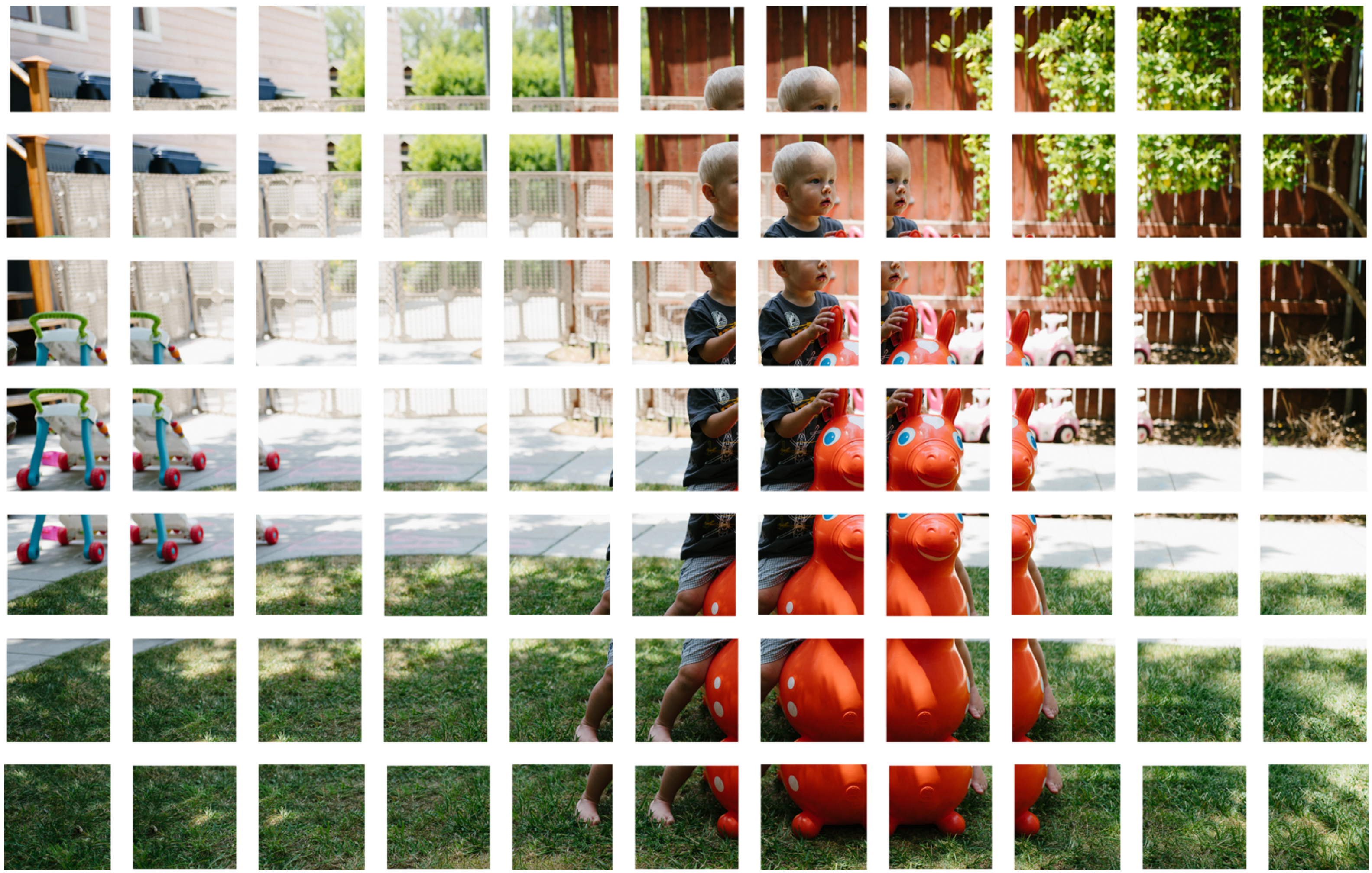
Complex Networks

SENet & SE-ResNet (Squeeze-and-Excitation)

- ILSVRC 2017 winner



That’s all folks!!! Thank you!



Annex I

CNN useful links:

•http://cs231n.github.io/convolutional-networks/

•https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

•https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html#fnref1

•https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

•https://docs.gimp.org/en/plug-in-convmatrix.html