Tensorflow & Keras:

Open source
Deep Learning

Deep Learning successes:

Board games (Go, Chess)

Super-human performance

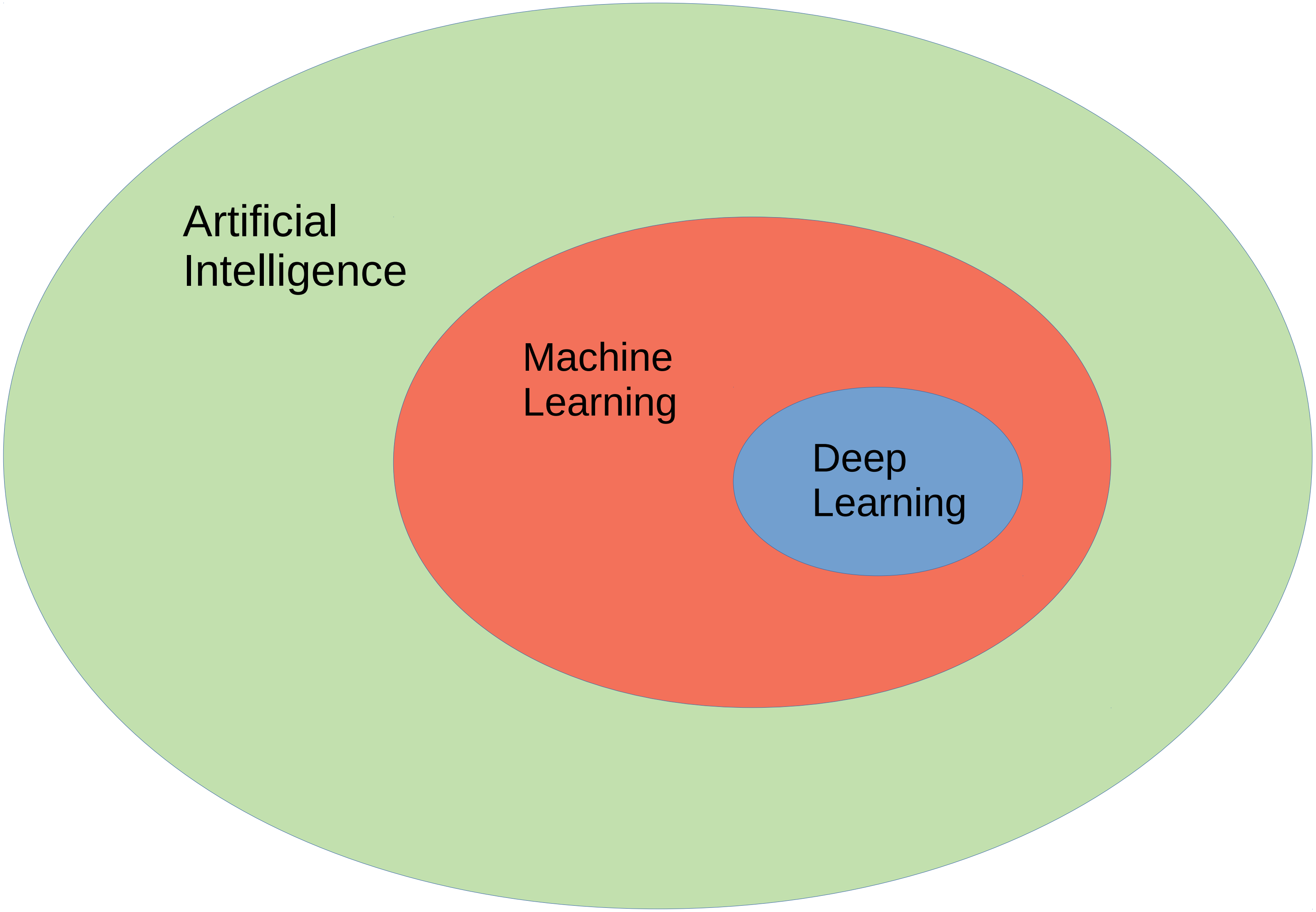
Object recognition / autonomous driving

Roughly human performance

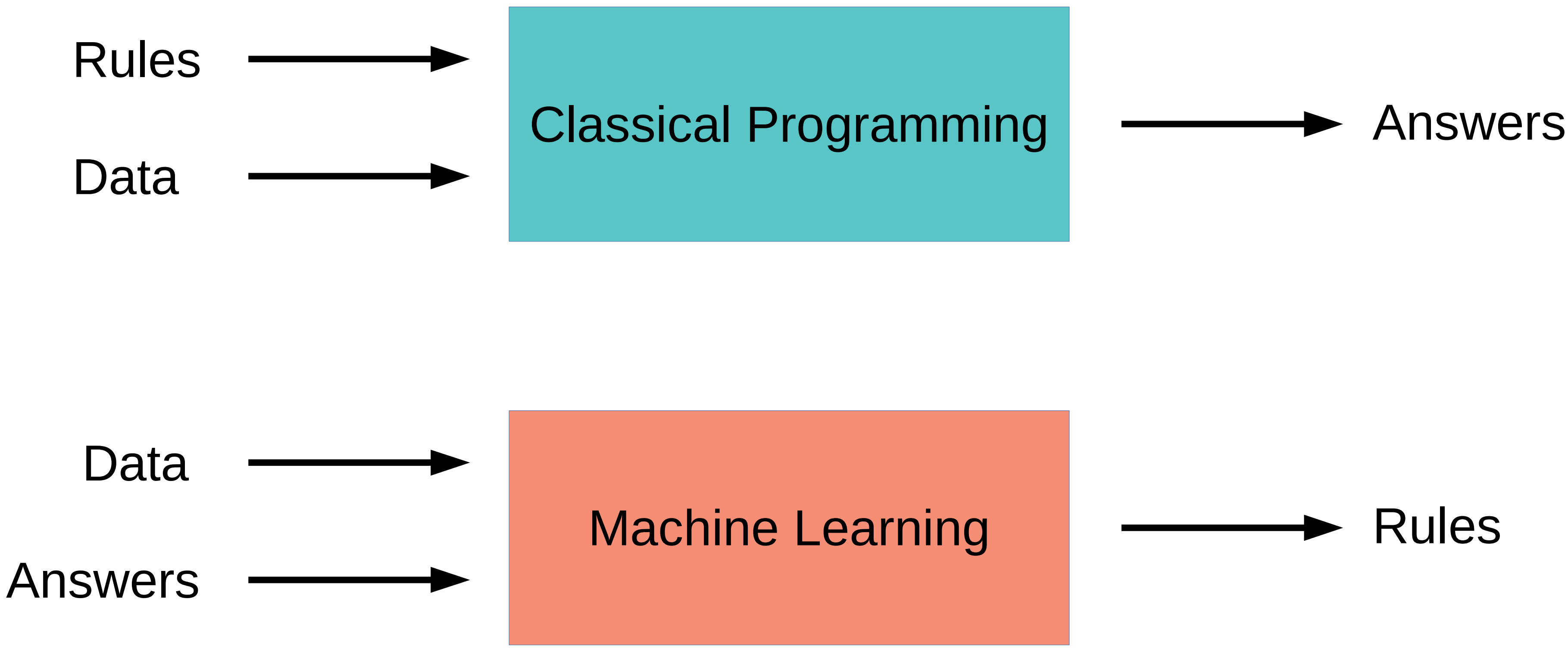
Speech recognition & machine translation

Remarkable progress, but still below human performance

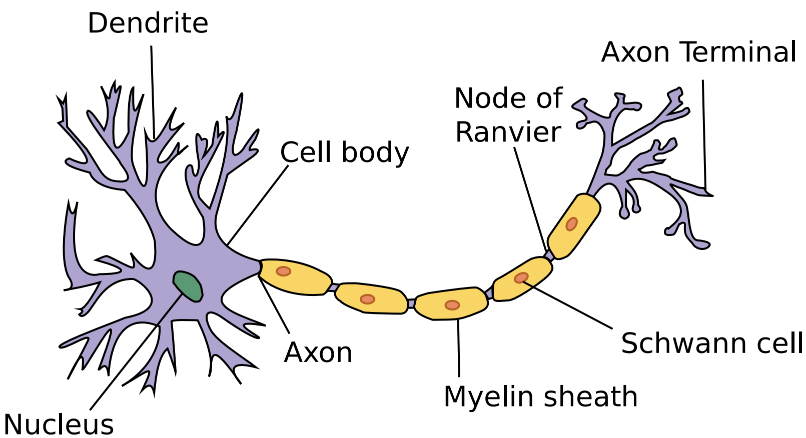
Artificial Intelligence, Machine Learning & Deep Learning

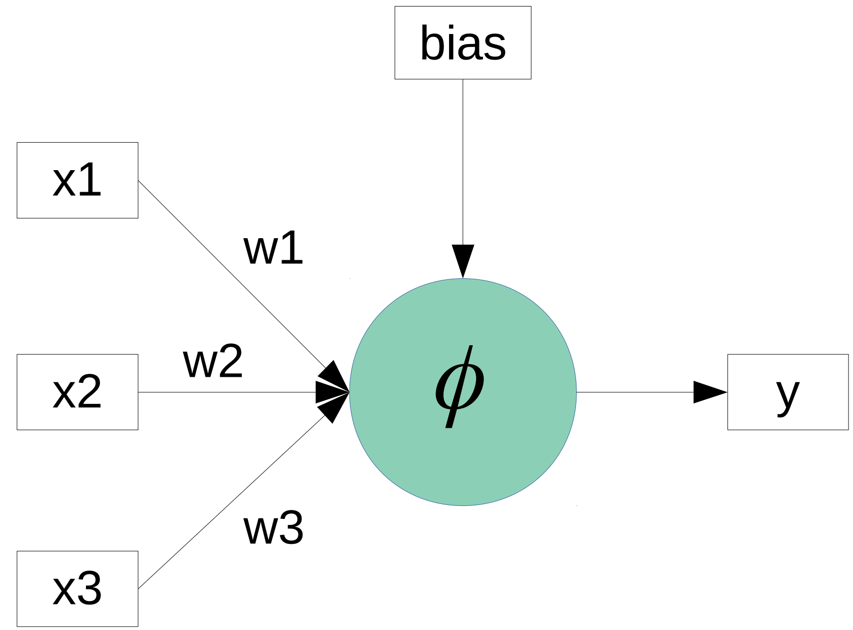


Classical Programming versus Machine Learning



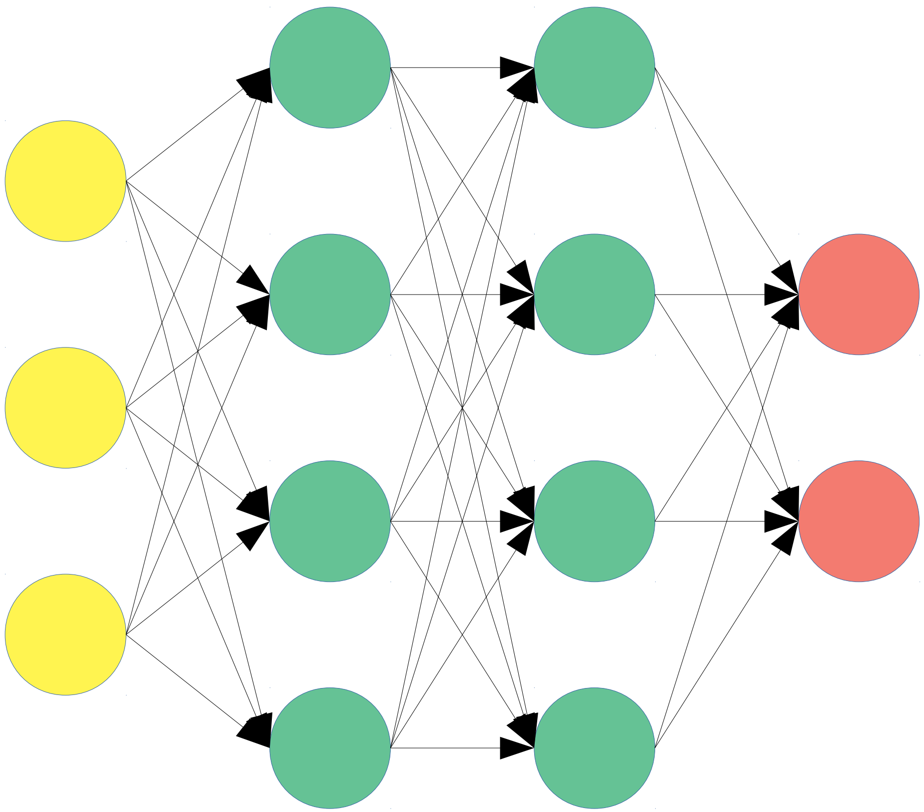
Artificial neurons: Building blocks for Deep Learning





$$y=\phi\left(\left(\sum^{n} x_n \cdot w_n\right)+b\right)$$

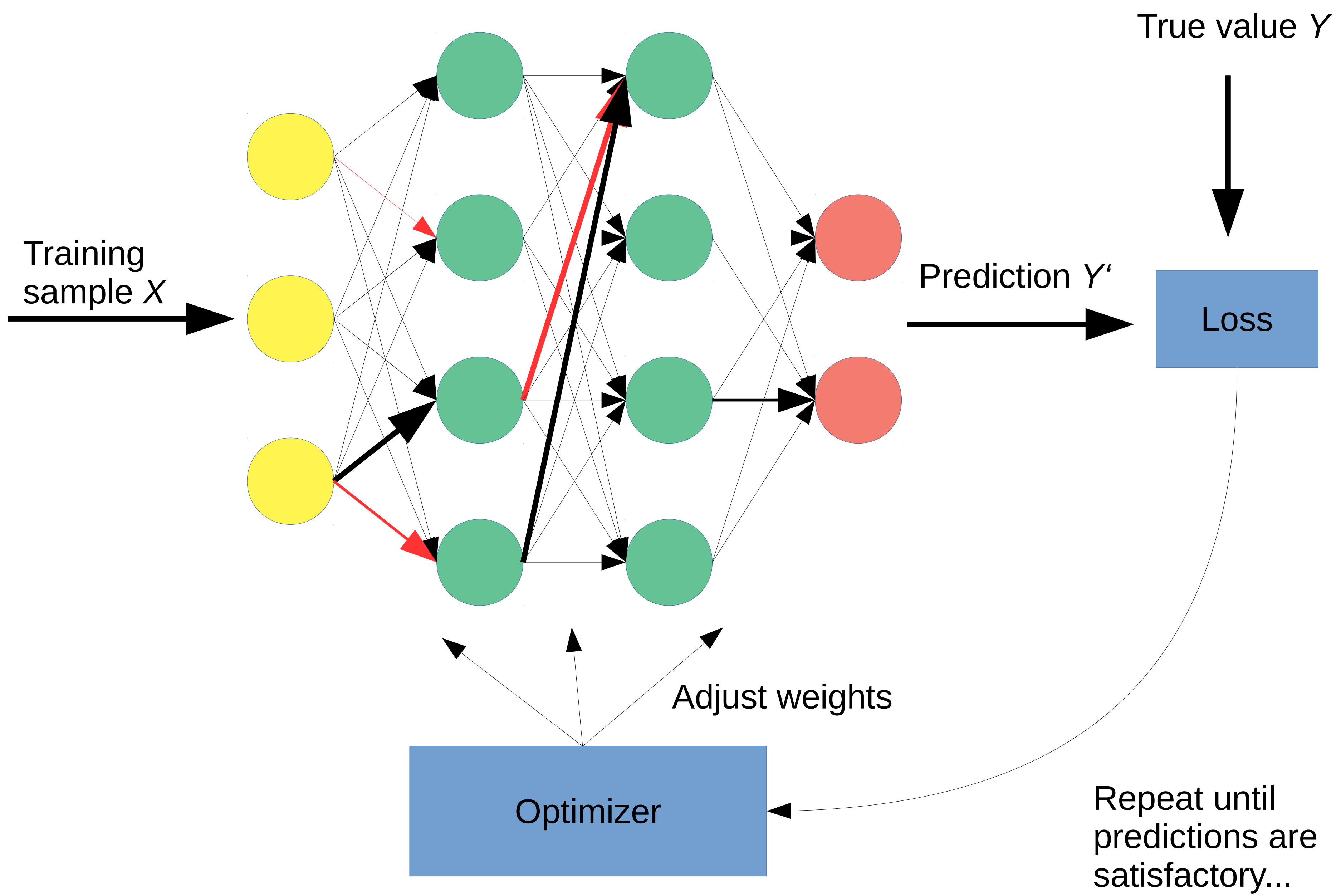
Dense feed forward networks



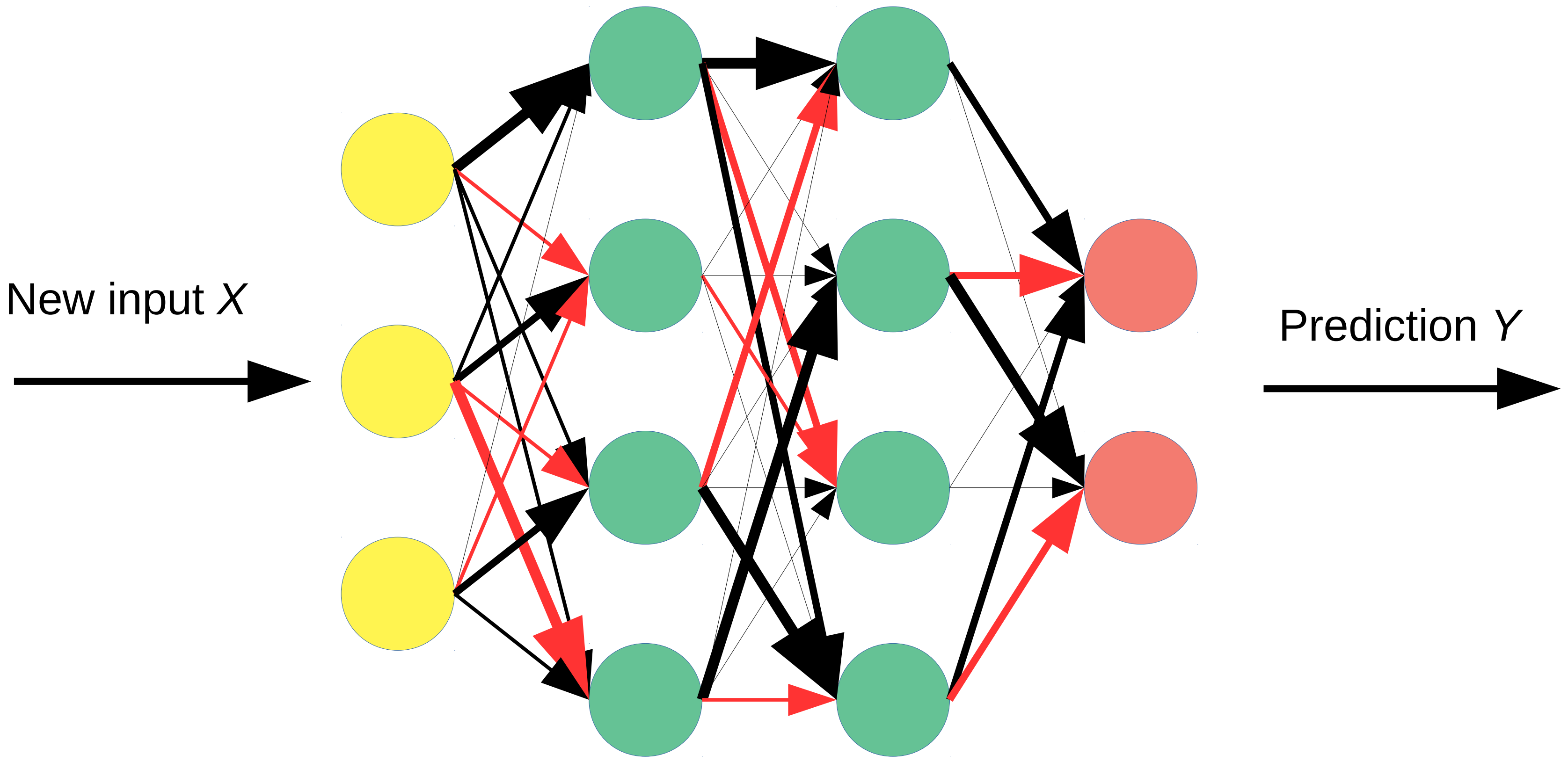
Feedforward networks: No cycles, data 'flows' from input layer to output layer

Dense networks: Each neuron is connected to each neuron of the next layer

Training a feed forward network - Supervised learning



Predicting from a trained feed forward network (‚Inference‘)



Production network with trained weights & biases

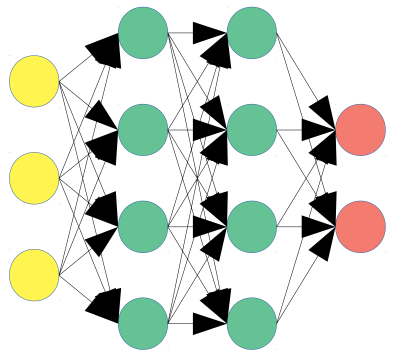
A well-trained network can generalize new input!

Learned knowledge is contained in the **weights**

Neural Networks: Common architectures

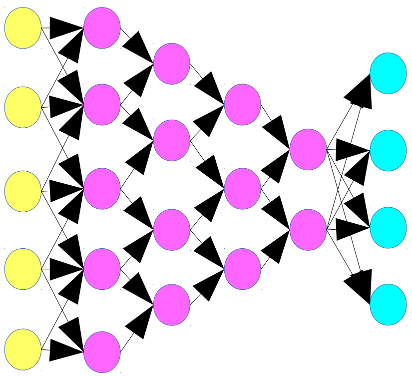
Dense Feedforward networks

- General purpose for classification / regression
- All neurons connected between layers
- Data as tensors flows from left to right



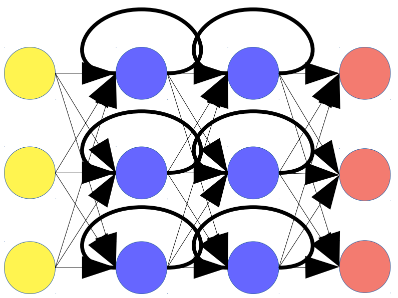
Convolutional networks

- Convolution operation on local features
- Funnel architecture
  -> increasing abstract concepts in deeper layers
- Object recognition; autonomous driving

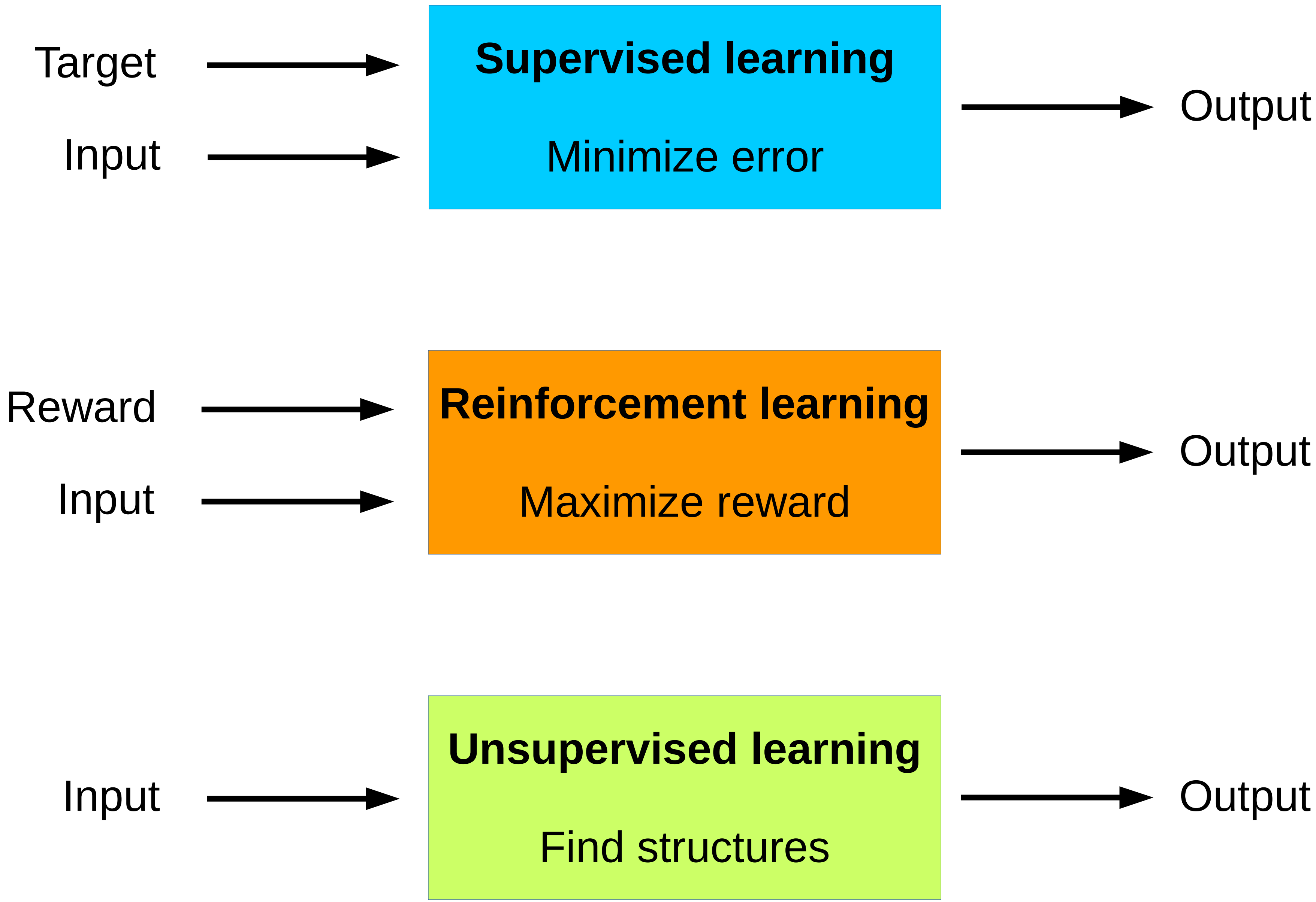


Recurrent networks

- Feedback loops -> Remembers past data
- Time series analysis; translation; speech recognition
- LSTM Long-Short Term Memory
  GRU Gated Recurrent Unit



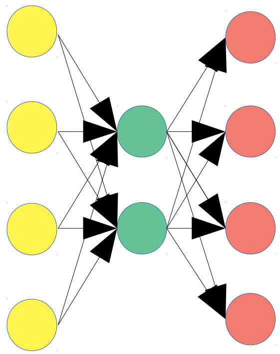
Learning methods



Neural Networks: Unsupervised / Generative architectures

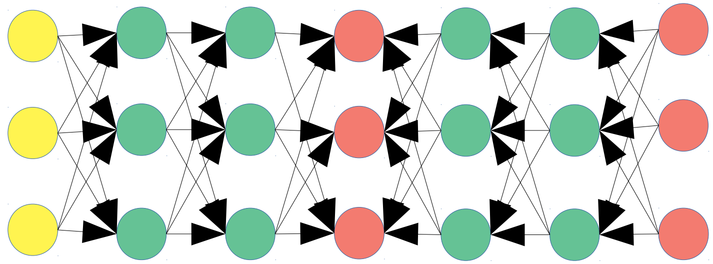
Autoencoder

- Trained with Input = Output
- Hourglass architecture
  -> compressed 'essence' at bottleneck layer
- DAE Denoising Autoencoder
  VAE Variational Autoencoder



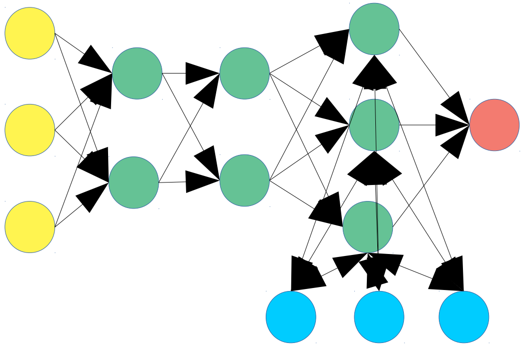
Generative adversarial networks

- 2 combined networks
  (Generator / Discriminator)
- Both work against each other & learn simultaneously
- Generates photorealistic pictures

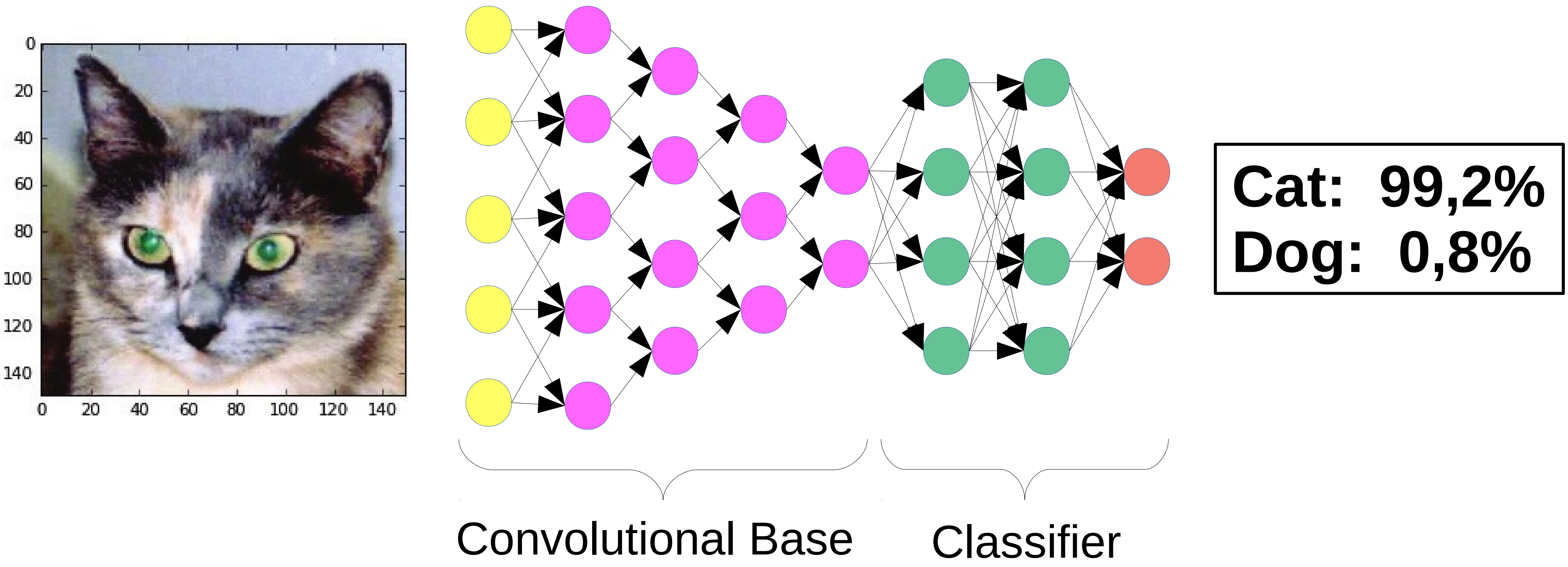


Neural Turing Machines

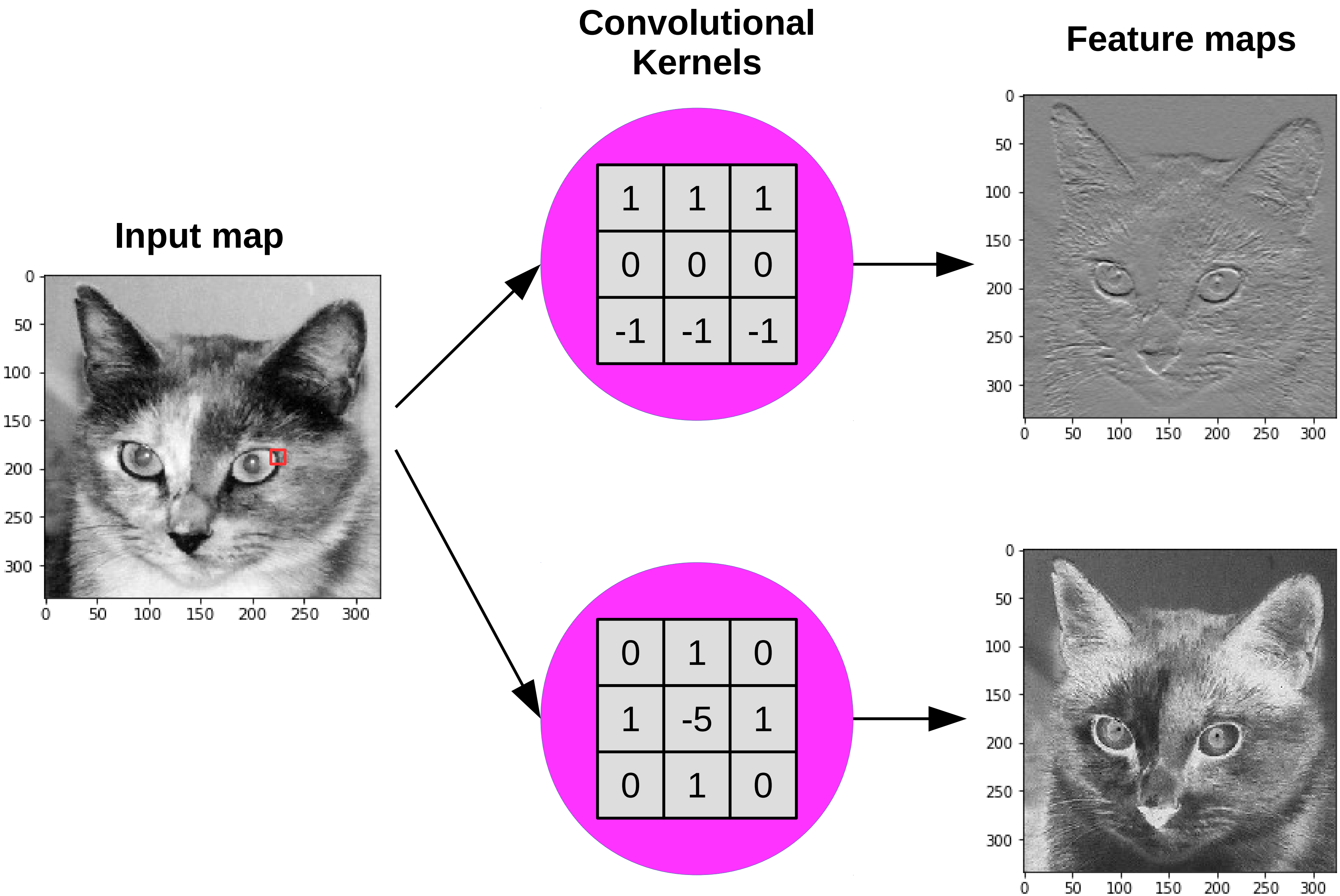
- Network connected to Turing memory bank
- Can generate algorithms by itself
- Highly experimental / academic
  (so far simple copy / sort algorithms)



Convolutional Networks



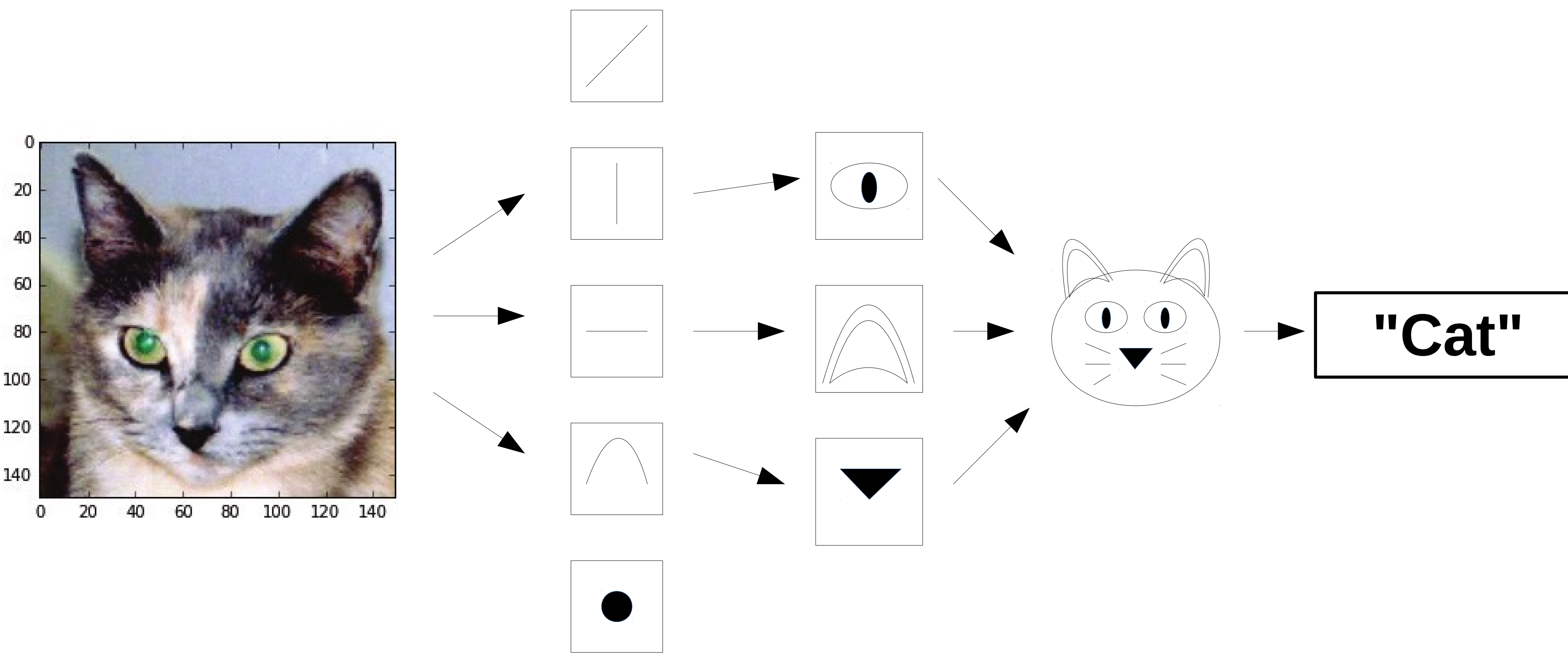
Convolution as image filters



Basic idea of a convnet

A convnet learns a *spatial hierarchy* of *translation-invariant* ***features***:

- Hyperlocal simple geometrical patterns in entry layers
- Local objects in middle layers
- Global high-level abstract concepts in deeper layers



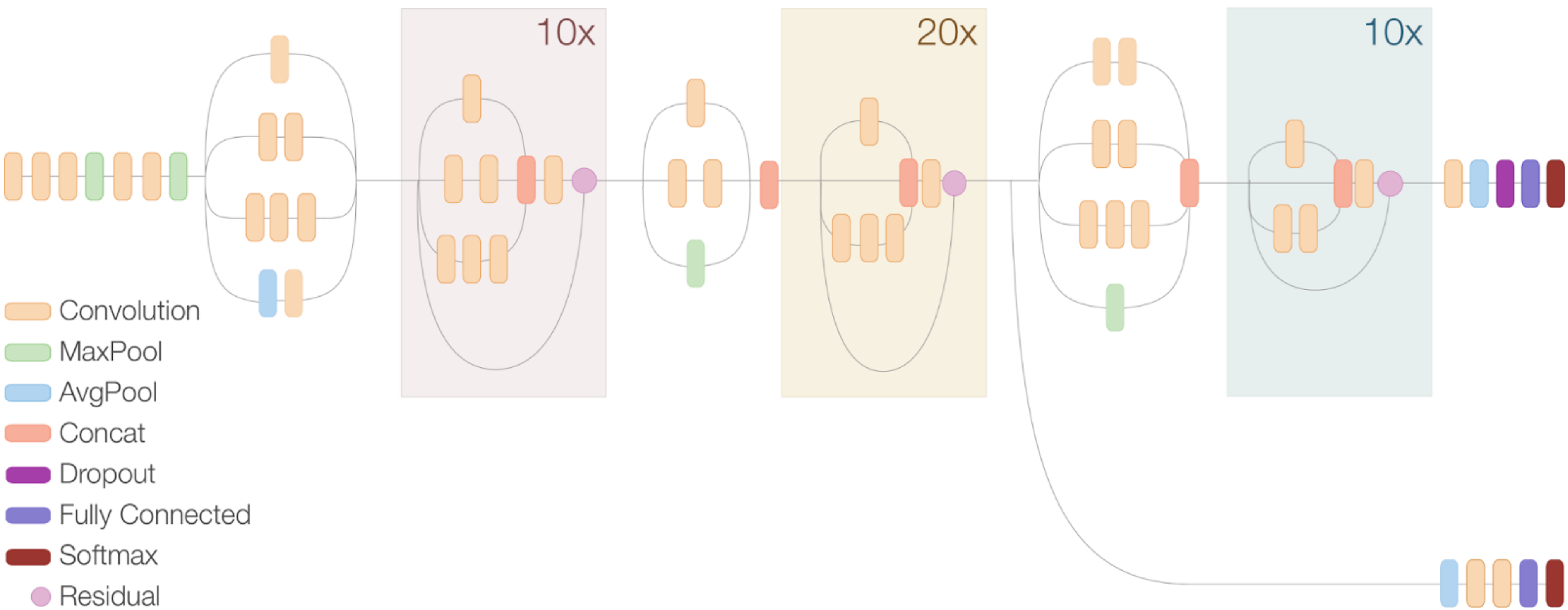
Current convnet performance (ILSVRC competition)

Team Name	Year	Top-5 error
SuperVision (University of Toronto)	2012	15.3%
Clarifai Corp. (USA)	2013	11.2%
GoogLeNet	2014	6.7%
MRSA (China)	2015	3.6%
Trimps-Soushen (Ministry of Public Security China)	2016	3.0%
WMW (Momenta Beijing & University Oxford)	2017	2.3%
Reference: Human expert(*)		5.1%
Pre-2012 Non-convolutional algorithms		> 25%

(*): Methodology at:
http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/

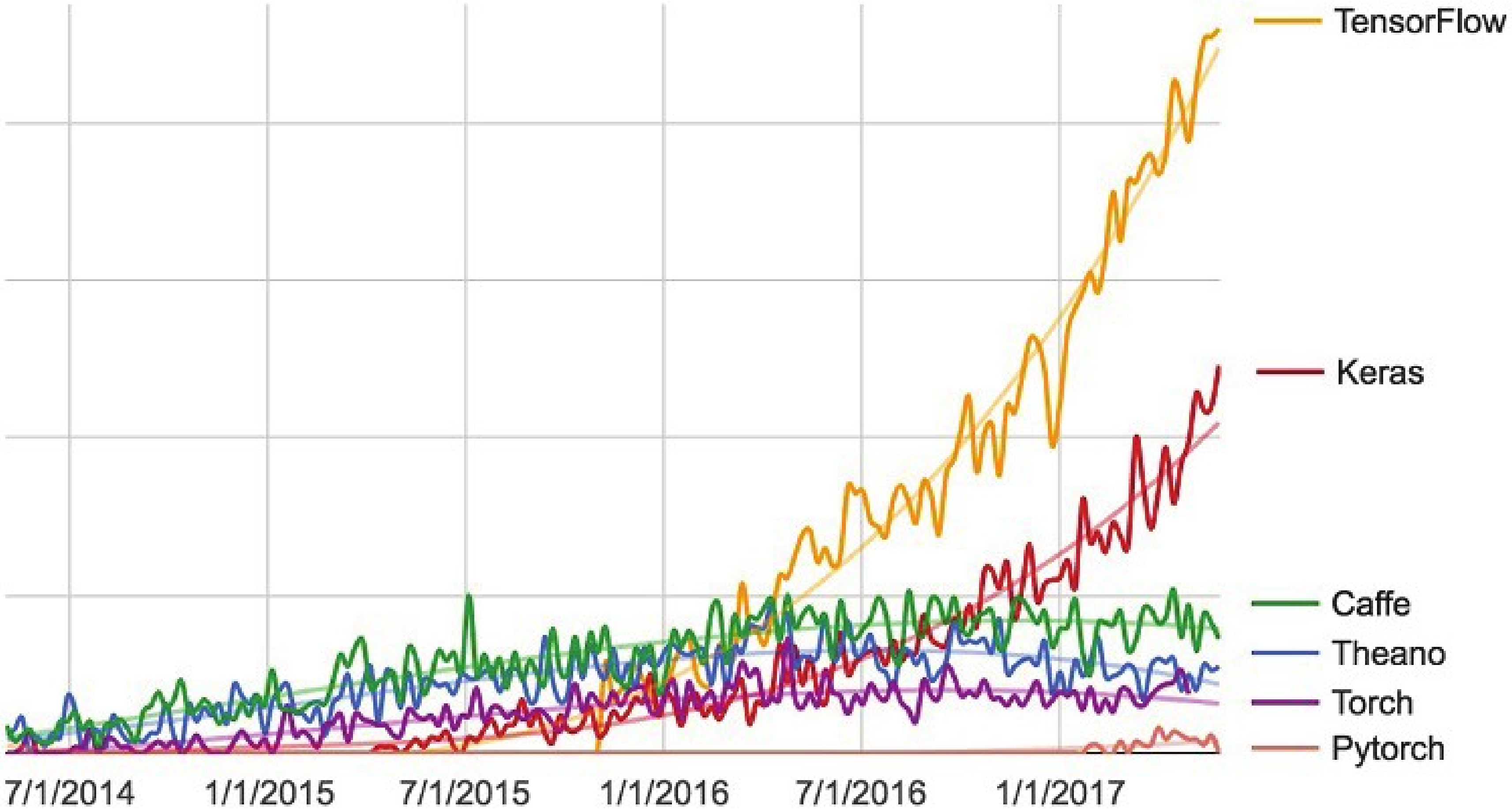
Inception Resnet V2 Network

Compressed View



Source: https://research.googleblog.com/2016/08/improving-inception-and-image.html

Google web search for Deep Learning frameworks



Source: "Deep Learning with Python", Francois Chollet, P. 61

The Tensorflow library

- A numerical computation library for dataflow graphs (‚tensor-flow‘)
- Can run on CPU and GPU
- Main application: Neural network machine learning
- Open Source since 5.11.2015 (Apache 2.0 license)

TensorFlow

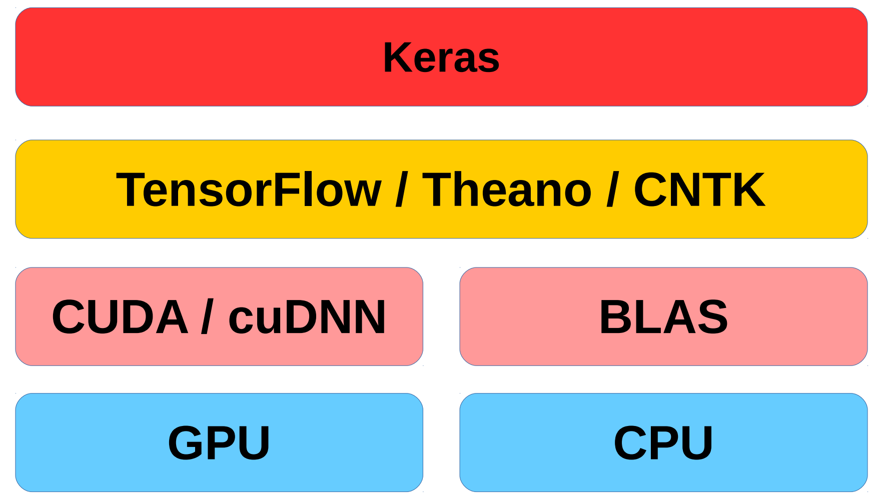
Official documentation: `https://www.tensorflow.org`

Google Research Blog: `https://research.googleblog.com/`

Sources & Resources: `https://github.com/tensorflow`

The Keras Neural Network Library

- A high-level neural networks API
- Written in Python
- Capable of running on top of *TensorFlow*, *CNTK*, or *Theano*.



Official documentation: `https://keras.io/`

Sources & example code: `https://github.com/keras-team/keras`

Keras Blog: `https://blog.keras.io/`

Keras Resources: `https://github.com/fchollet/keras-resources`

Installation

General requirements:

- **Python** 2.7 or 3.4+
- Python ***pip*** packet manager
- Recommended: ***h5py*** (save/load networks), ***matplotlib*** (image visualization)

Requirements for **GPU** version:

- Nvidia graphics card, compute capability >= 3.0
- Nvidia CUDA Toolkit & drivers 8.0 (Closed Source!)
- Nvidia CuDNN 6.0 library (requires registration at Nvidia)

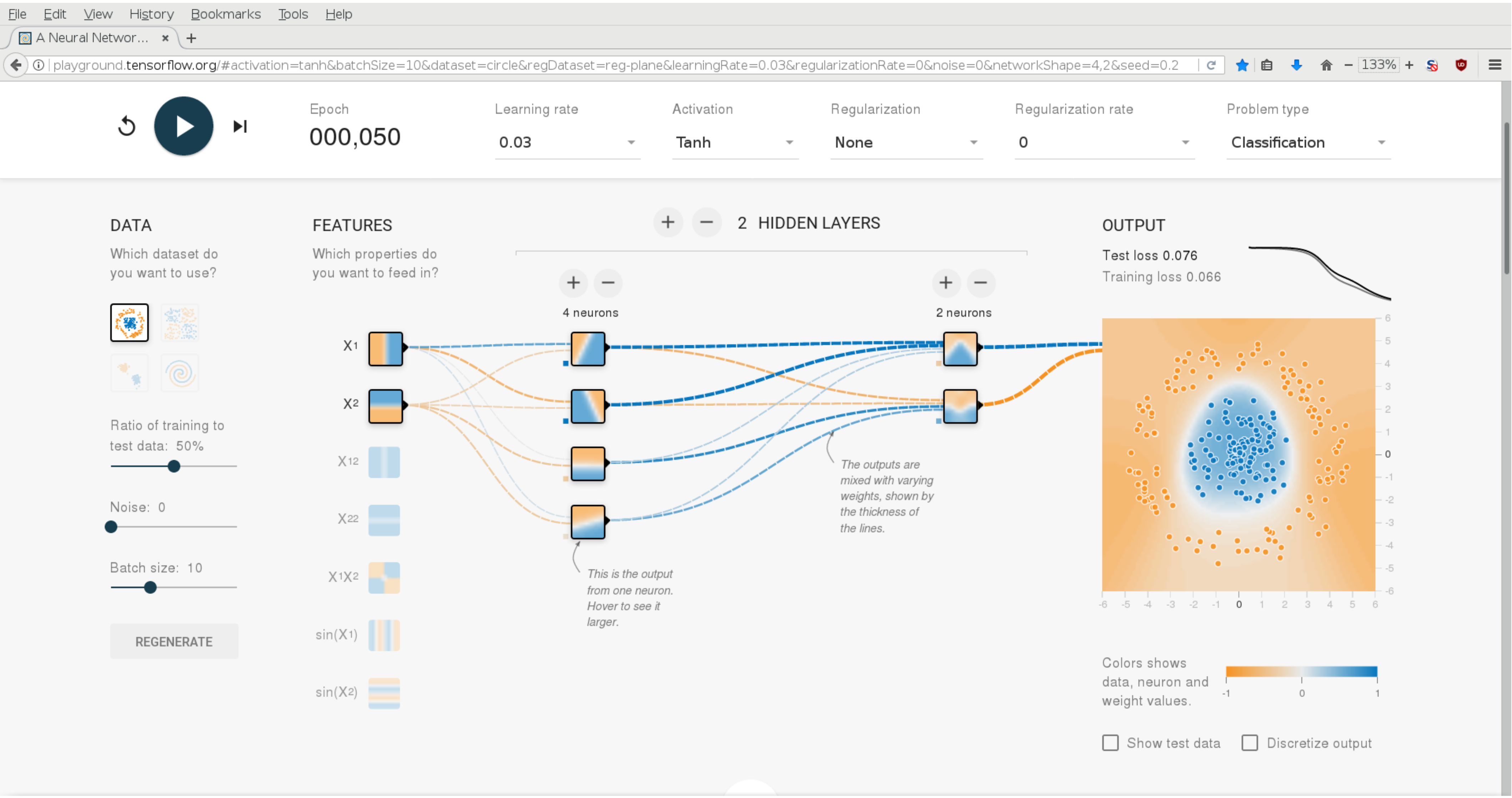
Tensorflow **openCL** support ‚work in progress‘ since 2016

```
> pip install tensorflow keras      # CPU-only version
> pip install tensorflow-gpu keras  # GPU version
```

Strongly recommended: System-specific compilation from tensorflow sources
Tedious, but worth it (CPU: SSE/AVX extensions; GPU: Nvidia compute capability)

`https://www.tensorflow.org/install/install_sources`

playground.tensorflow.org



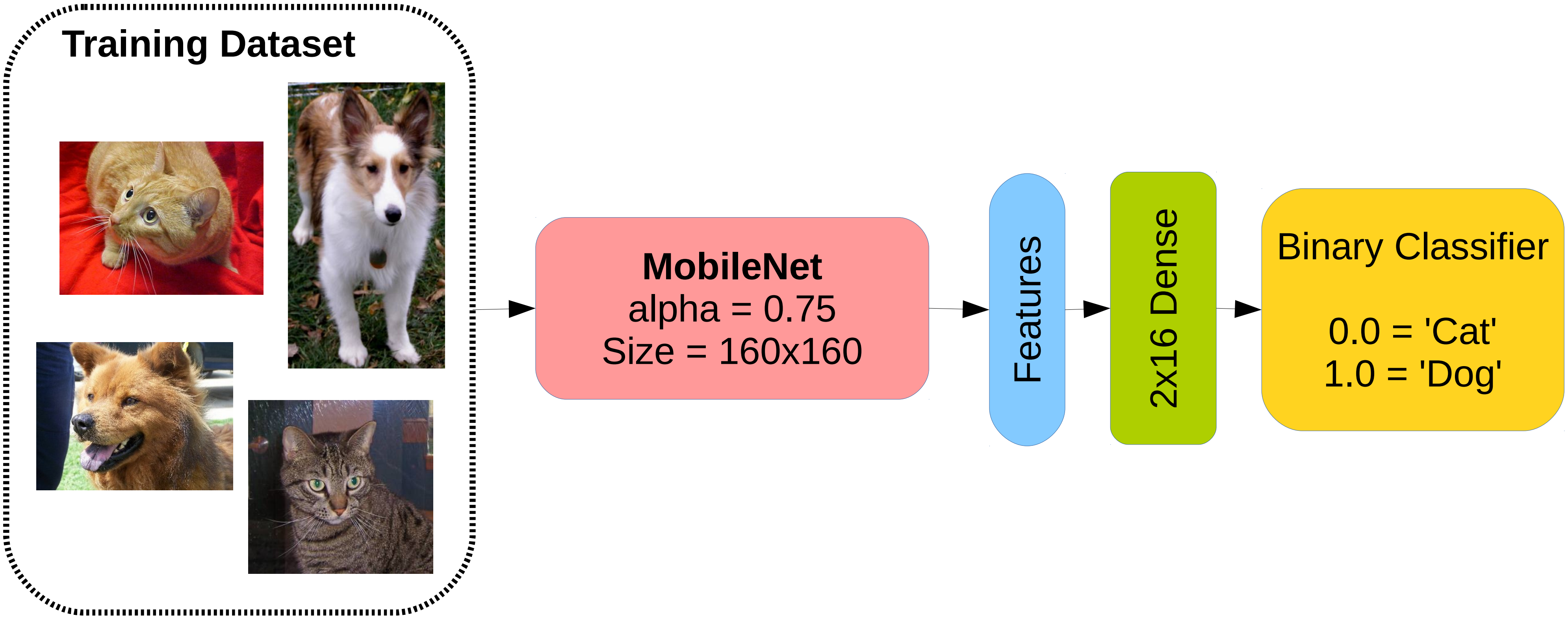
Keras: Provided datasets for learning / experimentation

Dataset	Number of entries (Train / Test)	Result type
CIFAR 10 (Color images of various items, 32x32)	50000 / 10000	10 classes
CIFAR 100 (Color images of various items, 32x32)	50000 / 10000	100 classes
IMDB Movie Reviews (Preprocessed texts)	25000	Binary (good / bad)
Reuters Newswire (Preprocessed texts)	11228	46 topics
MNIST Handwritten digits (Grayscale images 28x28)	60000 / 10000	10 classes (number 0-9)
MNIST Fashion icons (Grayscale images 28x28)	60000 / 10000	10 classes (clothing type)
Boston House Prices (13 location attributes)	506	Regression (House prices)

Keras: Provided pre-trained networks

Network (trained on ILSVRC 1000 image classes)	Size / Parameters	Top-5 classification error
VGG16	528 MB / 138 M	9.9 %
VGG19	549 MB / 143 M	9.0 %
ResNet50	99 MB / 25 M	7.1 %
InceptionV3	92 MB / 23 M	5.6 %
Xception	88 MB / 22 M	5.5 %
InceptionResNetV2	215 MB / 55 M	4.7 %
MobileNet	17 MB / 4 M	12.9 %
DenseNet (Keras 2.1.3)	81 MB / 20 M	6.7 %
NASNet (Keras 2.1.3)	24 MB / 5 M 344 MB / 88 M	8.4 % 3.8 %

Live-Demo "Dogs versus Cats". Workflow "Feature Extraction"



1) Extract *image features* with pre-trained convbase
2) Use features to train the *classifier*
3) Build production network with convbase & final classifier

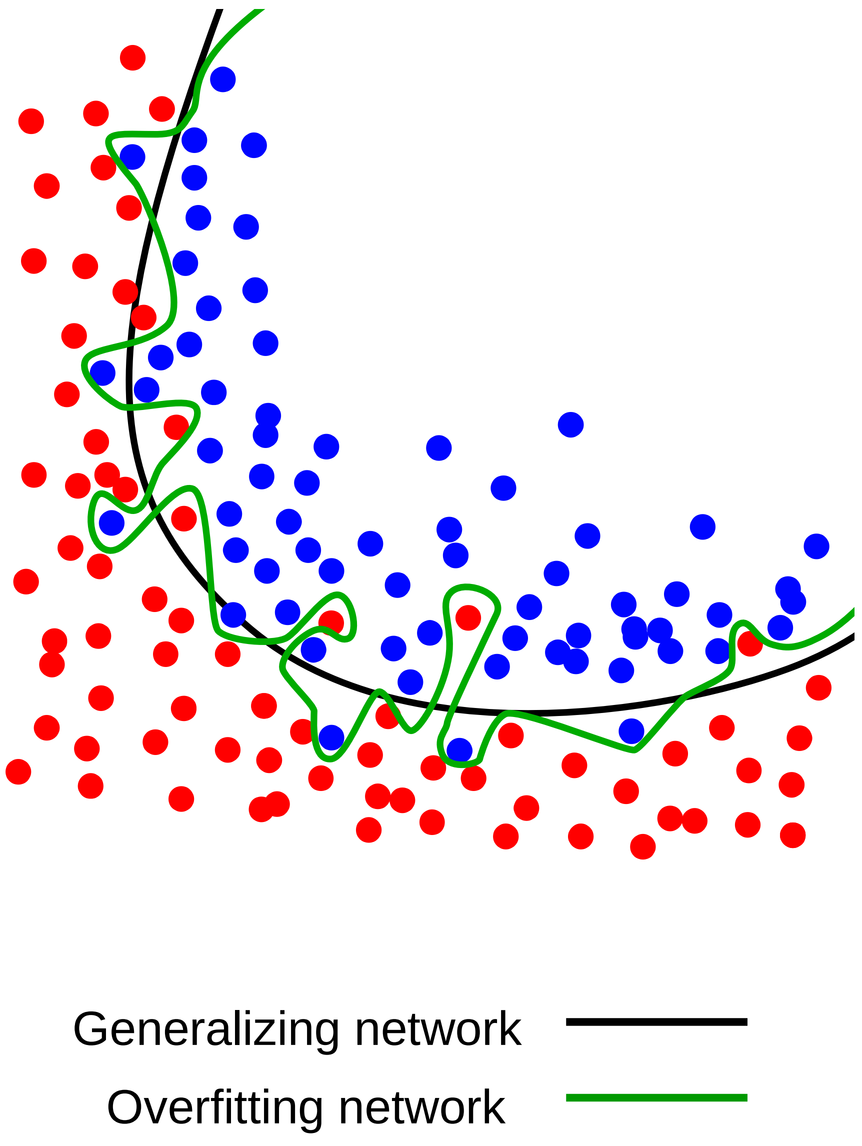
Overfitting – Biggest problem of neural networks

„Bigger is better“ doesn't apply to neural networks!

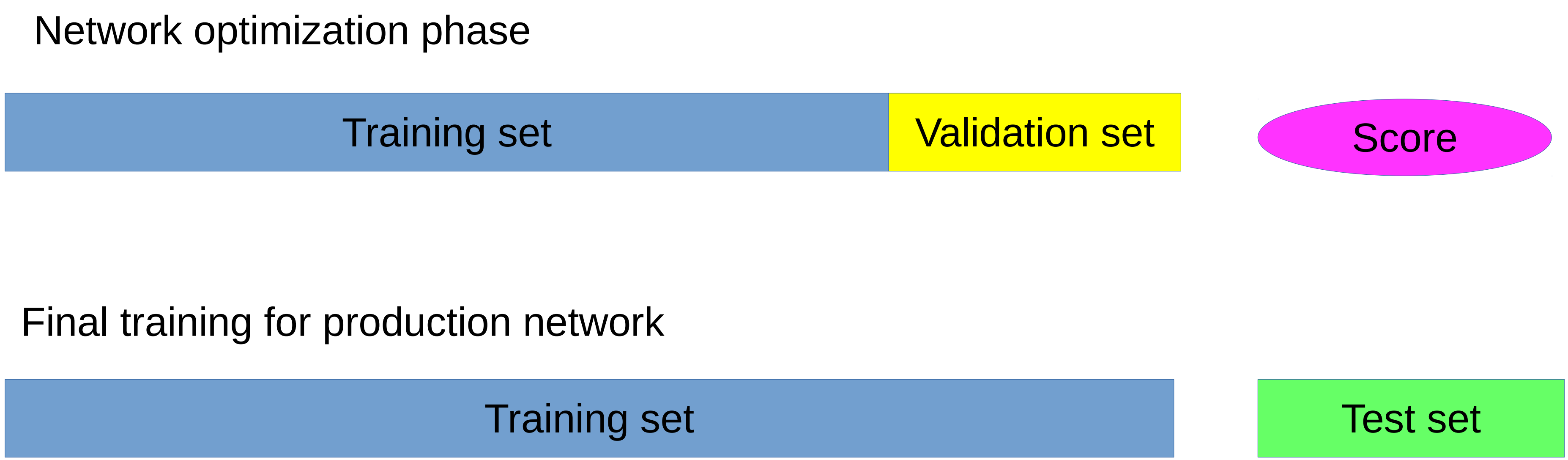
Over-complex networks just memorize and fail to generalize.

Common rookie mistake: Build a huge network and train it for too long.

100% accuracy on training data is irrelevant.



Hold-out validation



Underfitting - Overfitting