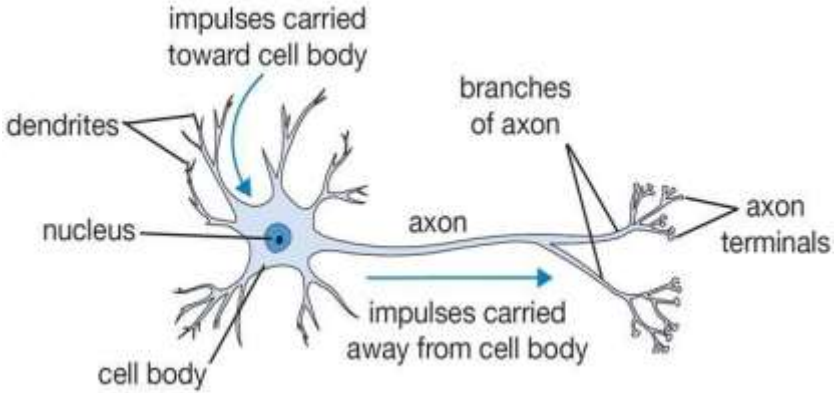
Neural Networks// Deep Learning Lab

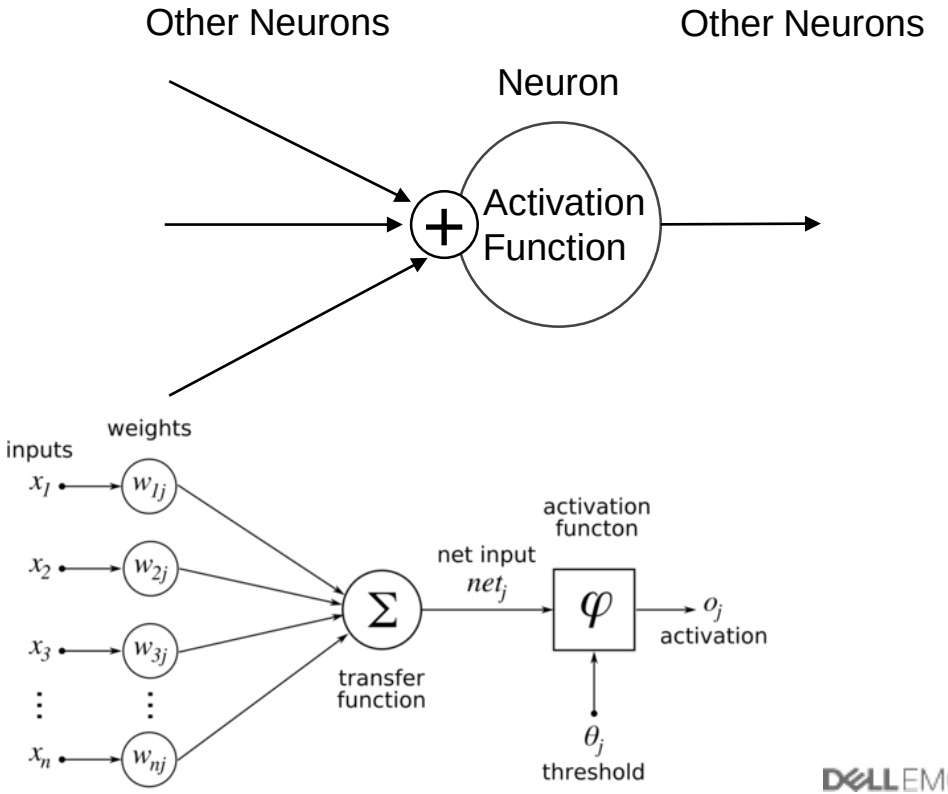
DELL EMC

Agenda

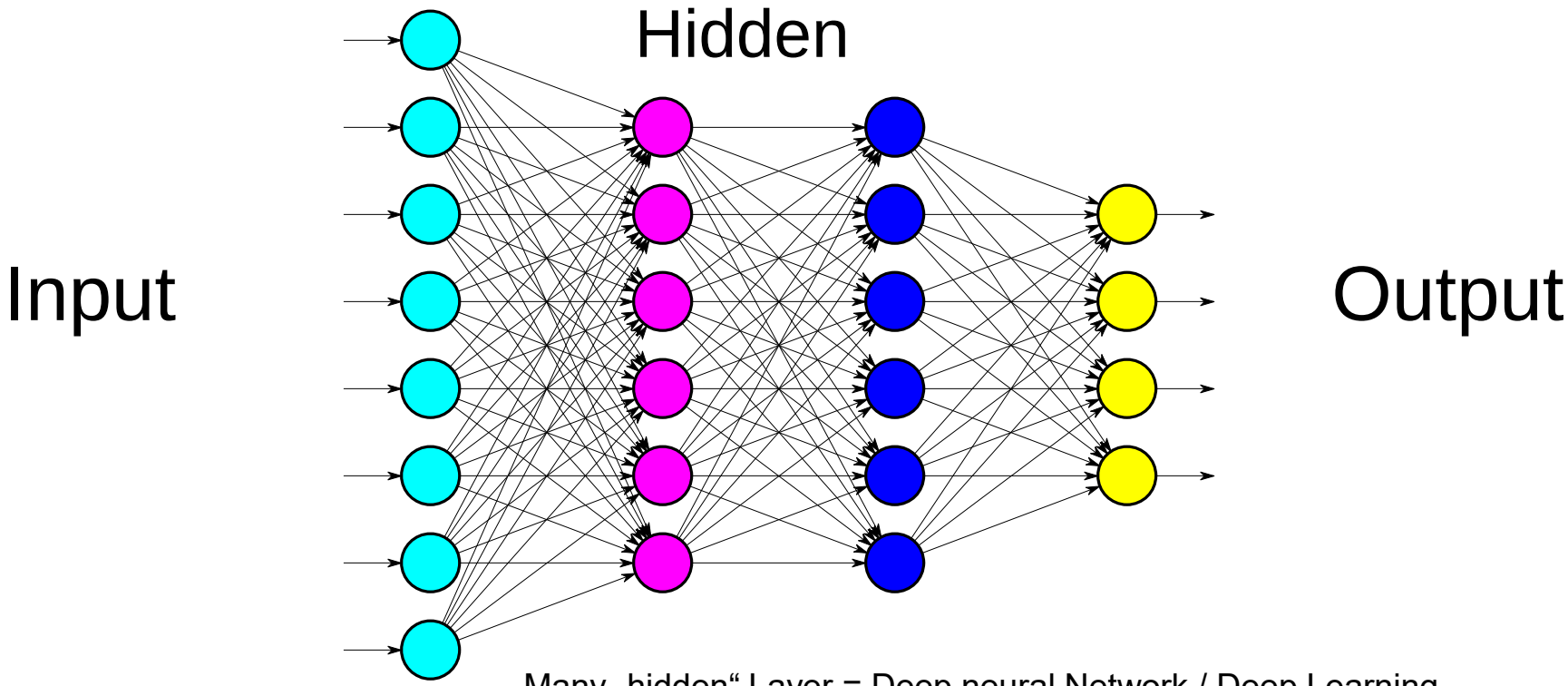
- Introduction
- Deep Learning 201
- CPU vs. GPU
- Deep Learning Frameworks
- CUDA / pycuda Lab with Intro Lecture
- Deep Learning Lab with Intro Lecture
- Fare Well ☺

Neural Network, how they work, Neurons





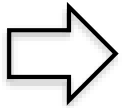
Neural Network

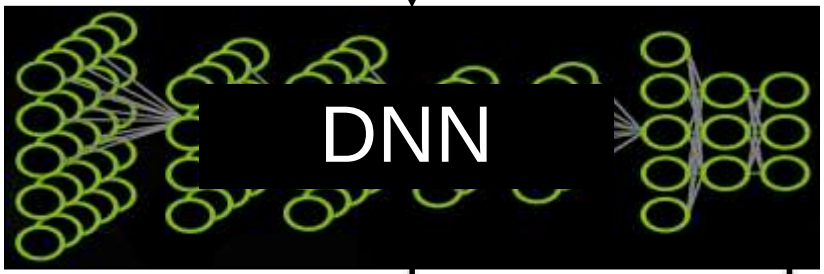


Many „hidden“ Layer = Deep neural Network / Deep Learning

Training and Inference

Train:

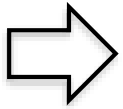


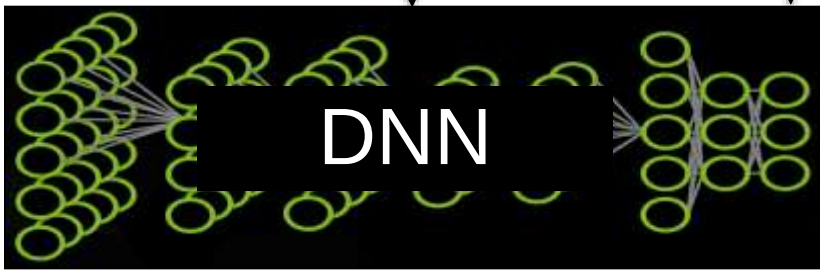


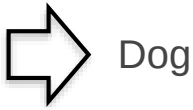
Backpropagation

Errors

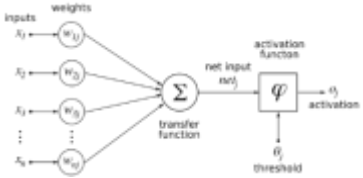
Deploy, Inference:







Neural Nets / Deep Learning is Tensor Math



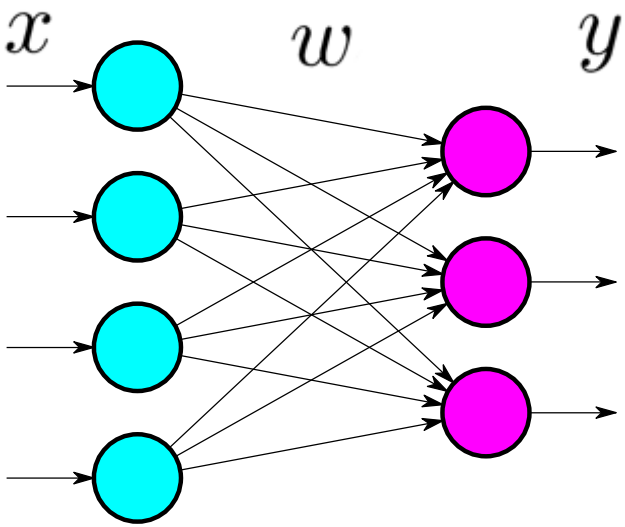
A very simple universal Approximation

$$y_j = f\left(\sum_i w_{ij} x_i\right)$$

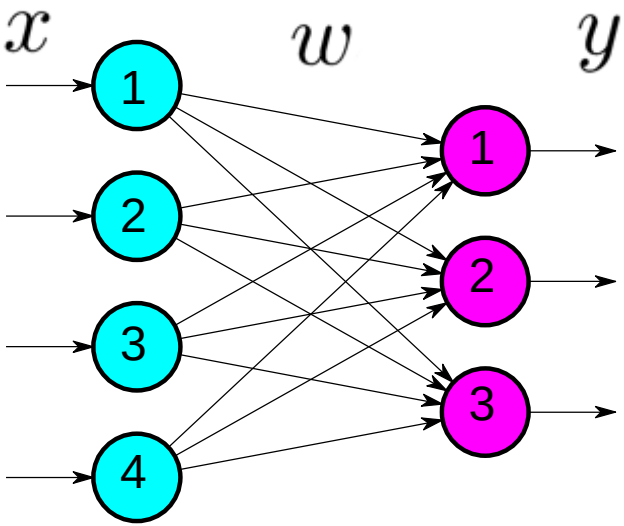
One layer

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

nonlinearity



Neural Net Vector Math



For one Output Y

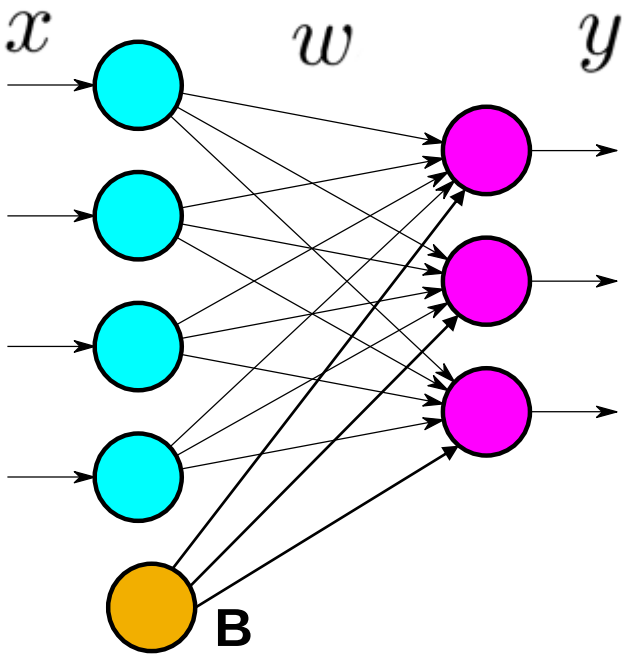
$$Y1 = f(X1*W11 + X2*W12 + X3*W13 + X4*W14)$$

For whole Layer

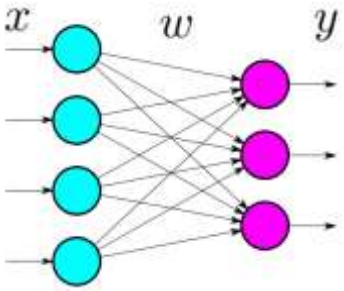
$$f\left(\begin{pmatrix} W11 & W12 & W13 & W14 \\ W21 & W22 & W23 & W24 \\ W31 & W32 & W33 & W34 \end{pmatrix} \bullet \begin{pmatrix} X1 \\ X2 \\ X3 \\ X4 \end{pmatrix}\right) = \begin{pmatrix} Y1 \\ Y2 \\ Y3 \end{pmatrix}$$

f : Activation Function

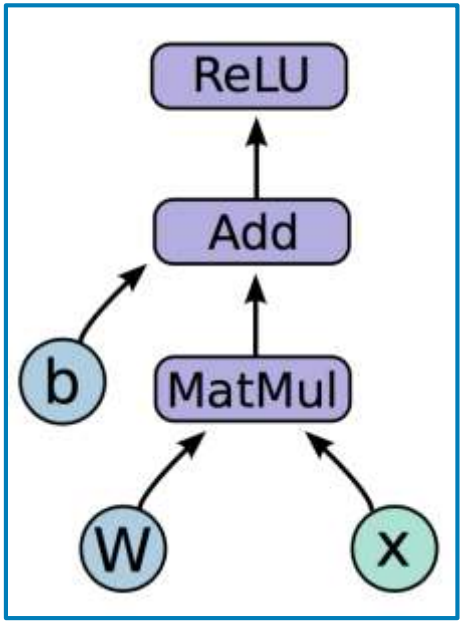
Neural Network and Bias for Normalization



Tensorflow Programing Model

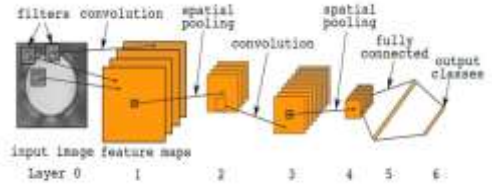



```
import numpy as np
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100),
                -1, 1))

x = tf.placeholder(tf.float32, (None, 784))
h_i = tf.nn.relu(tf.matmul(x, W) + b)



sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h_i, {x: np.random.random(64, 784)})
```



See: Tensorflow and deep learning - without a PhD by Martin Görner

LeNet : Tensorflow vs. Keras




```
# The model
stride = 1  # output is 28x28
Y1 = tf.nn.relu(tf.nn.conv2d(X, W1, strides=[1, stride, stride, 1], padding='SAME') + B1)
stride = 2  # output is 14x14
Y2 = tf.nn.relu(tf.nn.conv2d(Y1, W2, strides=[1, stride, stride, 1], padding='SAME') + B2)
stride = 2  # output is 7x7
Y3 = tf.nn.relu(tf.nn.conv2d(Y2, W3, strides=[1, stride, stride, 1], padding='SAME') + B3)

# reshape the output from the third convolution for the fully connected layer
YY = tf.reshape(Y3, shape=[-1, 7 * 7 * M])

Y4 = tf.nn.relu(tf.matmul(YY, W4) + B4)
YY4 = tf.nn.dropout(Y4, pkeep)
Ylogits = tf.matmul(YY4, W5) + B5
Y = tf.nn.softmax(Ylogits)

# cross-entropy loss function (= -sum(Y_i * log(Yi)) ), normalised for batches of 100  images
# TensorFlow provides the softmax_cross_entropy_with_logits function to avoid numerical stability
# problems with log(0) which is NaN
cross_entropy = tf.nn.softmax_cross_entropy_with_logits(logits=Ylogits, labels=Y_)
cross_entropy = tf.reduce_mean(cross_entropy)*100

# accuracy of the trained model, between 0 (worst) and 1 (best)
correct_prediction = tf.equal(tf.argmax(Y, 1), tf.argmax(Y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# training step, the learning rate is a placeholder
train_step = tf.train.AdamOptimizer(lr).minimize(cross_entropy)

# init
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

Tensorflow

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Keras

Neural Network in 10 Lines of Python

```
X = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
y = np.array([[0,1,1,0]]).T
syn0 = 2*np.random.random((3,4)) - 1
syn1 = 2*np.random.random((4,1)) - 1
for j in xrange(60000):
    l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
    l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
    l2_delta = (y - l2)*(l2*(1-l2))
    l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
    syn1 += l1.T.dot(l2_delta)
    syn0 += X.T.dot(l1_delta)
```

https://iamtrask.github.io/2015/07/12/basic-python-network/#viewSource

Image Processing Convolutional Networks

COMPUTER VISION TASKS

Image Classification

Image Classification + Localization

Object Detection

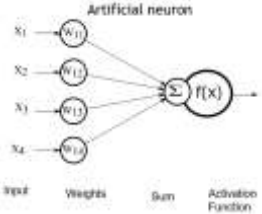
Image Segmentation

Our LAB

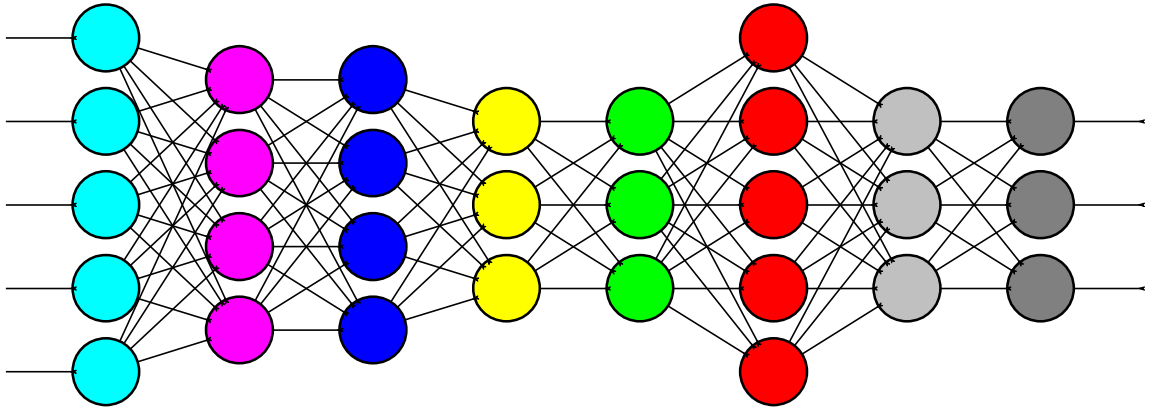
(inspired by a slide found in cs231n lecture from Stanford University)

ARTIFICIAL NEURAL NETWORK

A collection of simple, trainable mathematical units that collectively learn complex functions



Hidden layers / Deep / Fully Connect



Input layer

Output layer

Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

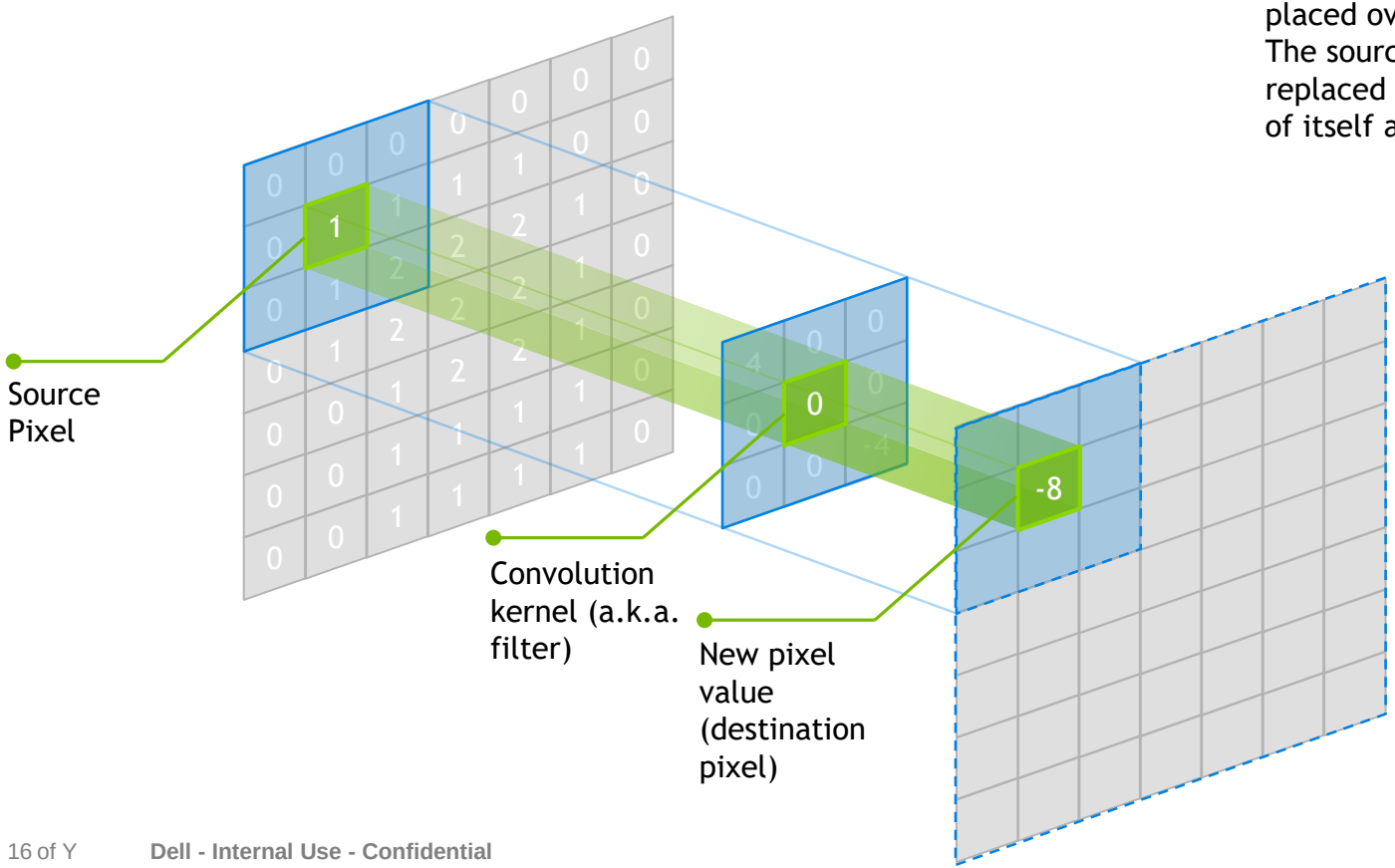
Connvolutional Neural Networks

For Image Classification

- Fully connected NN are good for „flat“ Classification Problems
- They are not good for Images as they do not respect the 2 Dimension of an Image
- Convolutional Networks were developed to respect the 2 Dimension of Images

CONVOLUTION

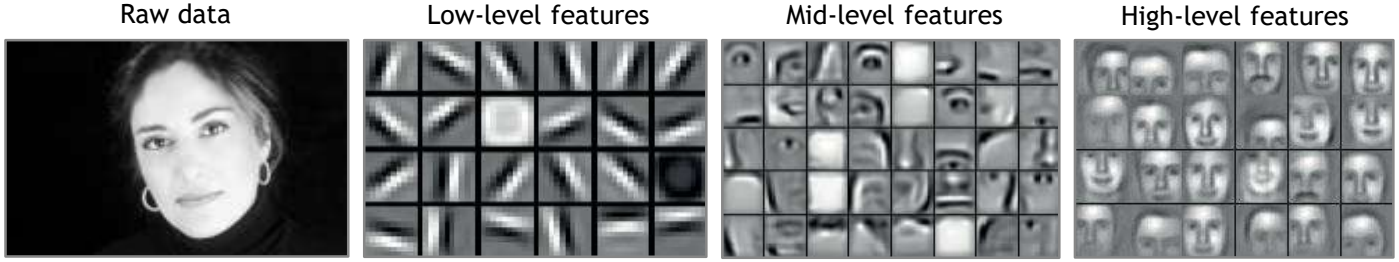
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

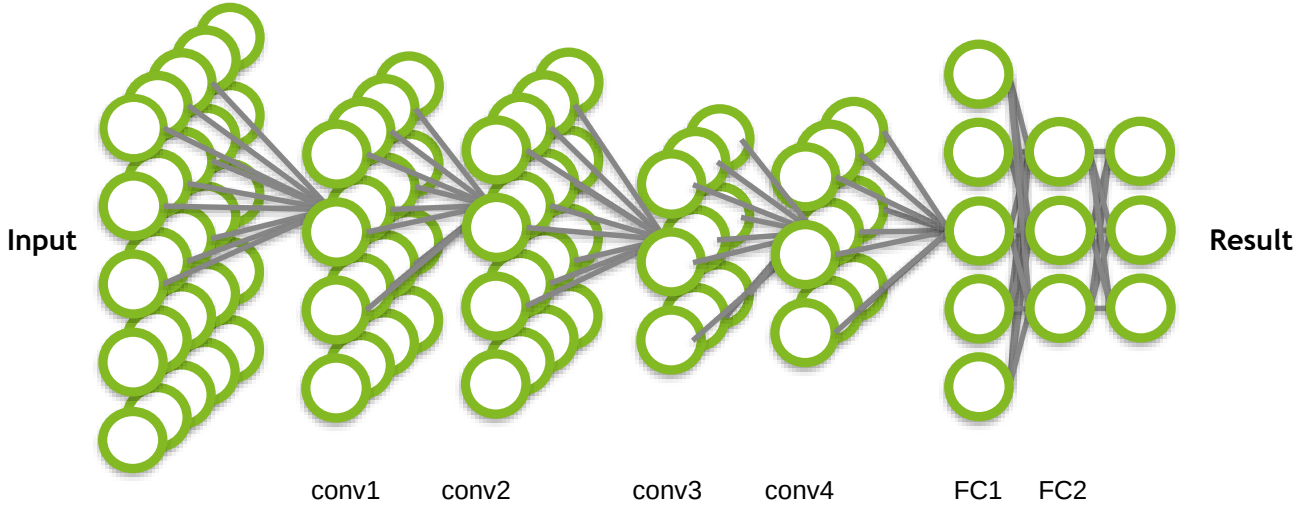


Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Typical Convolution based Filter in Computer Vision / Image Editing

CONVOLUTION DEEP NEURAL NETWORK (CNN)





Application components:

Task objective
e.g. Identify face

Training data
10-100M images

Network architecture
~10s-100s of layers
1B parameters

Learning algorithm
~30 Exaflops
1-30 GPU days

LeNet (Yann Lecunn)

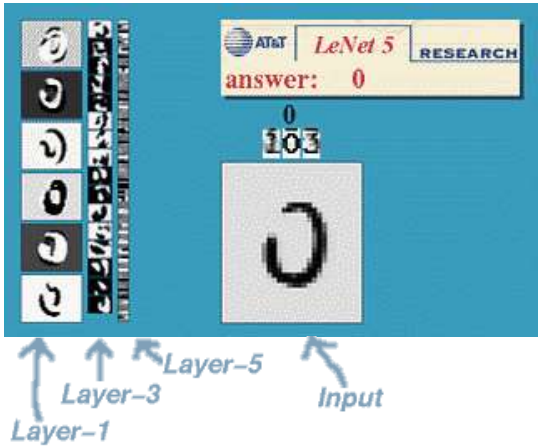
C1,C3,C5 : Convolutional layer.

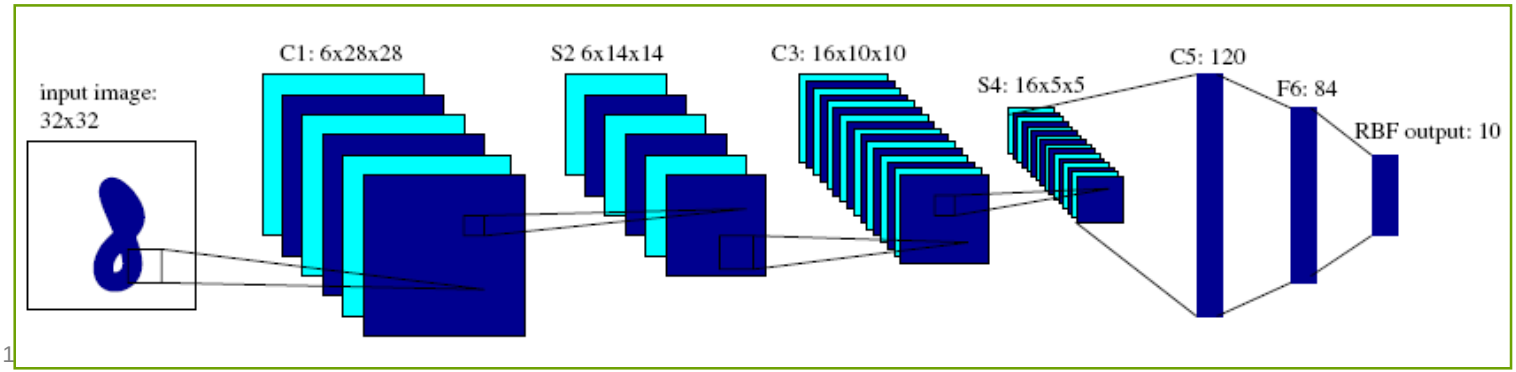
5 × 5 Convolution matrix.

S2 , S4 : Subsampling layer.

Subsampling by factor 2.

F6 : Fully connected layer.

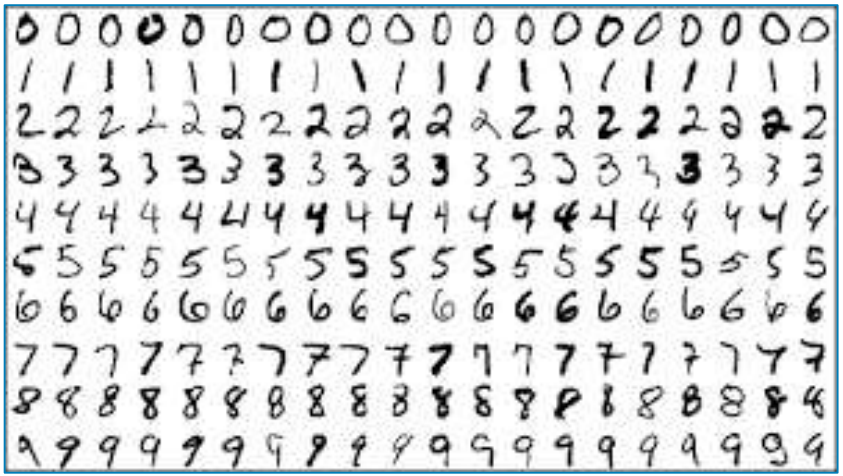




HANDWRITTEN DIGIT RECOGNITION

MNIST data set of handwritten digits from Yann Lecun's website

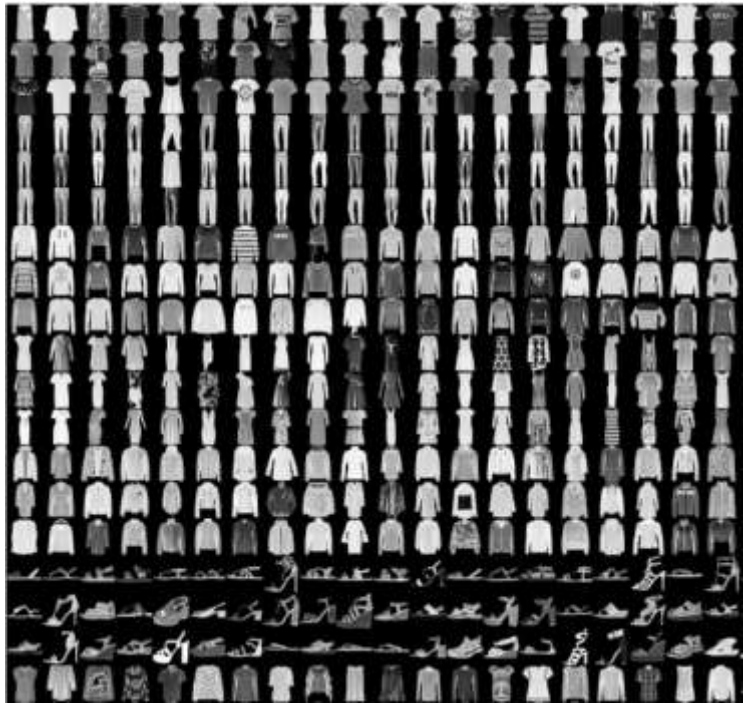
- All images are 28x28 grayscale
  - Pixel values from 0 to 255
- 60K training examples / 10K test examples
- Input vector of size 784
  - 28 * 28 = 784
- Output value is integer from 0-9



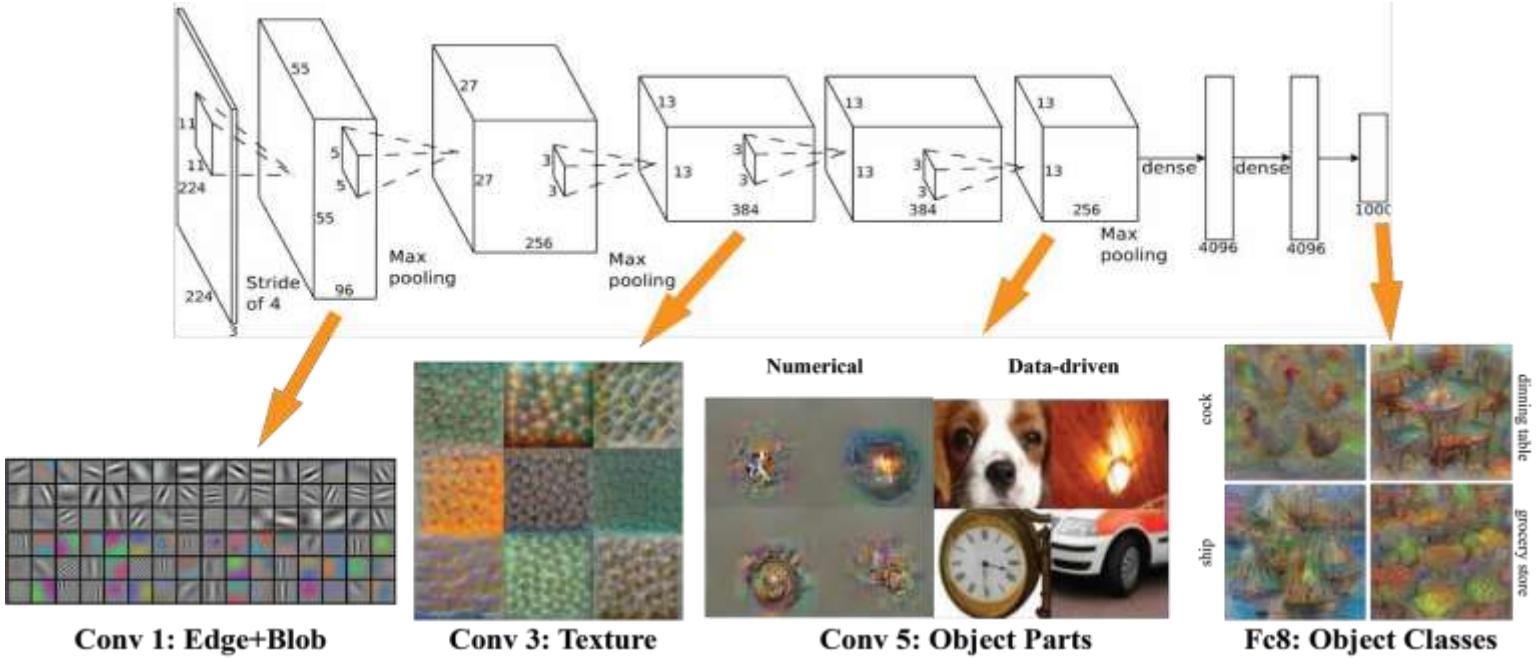
Fashion MNIST

Zalando data set as a replacement of MNIST from Yann Lecun

- All images are 28x28 grayscale
  - Pixel values from 0 to 255
- 60K training examples / 10K test examples
- Input vector of size 784
  - 28 * 28 = 784
- Output value is integer from 0-9



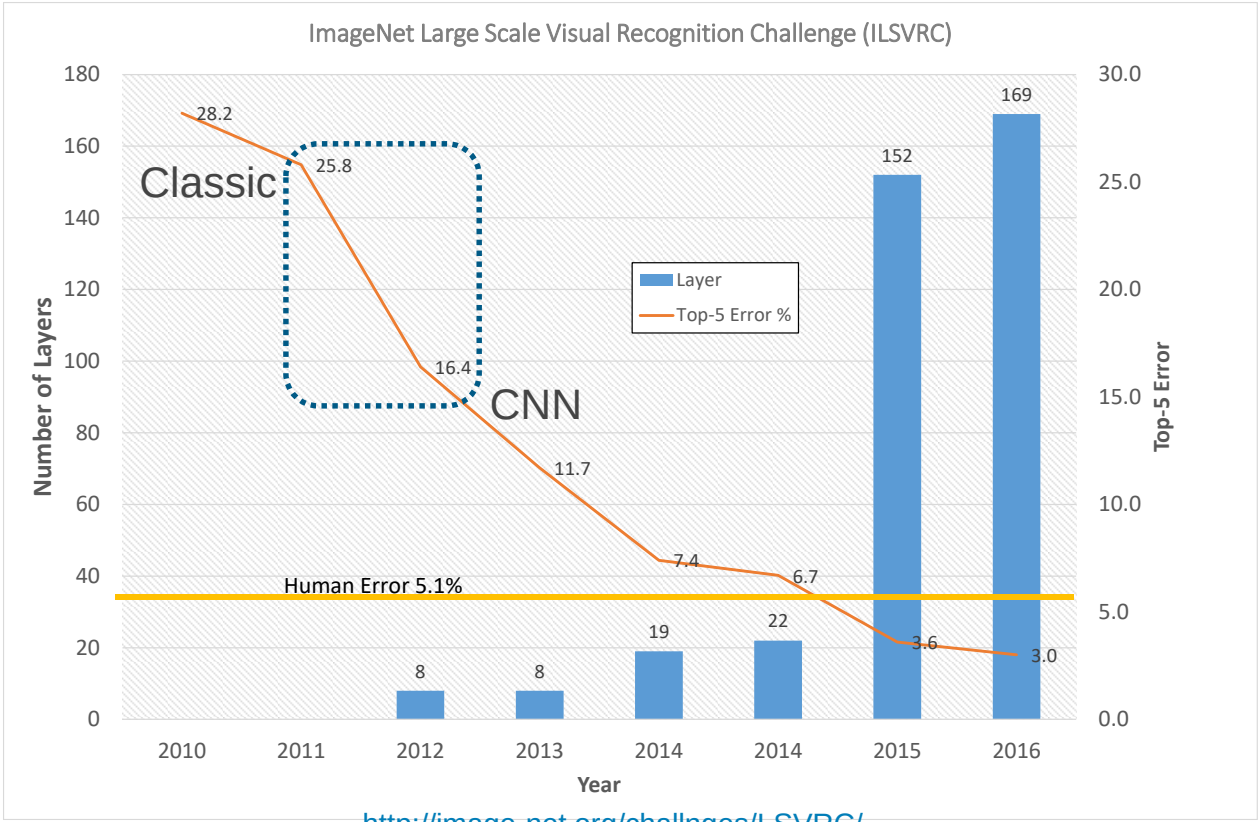
Deep Learning, Convolutional Neural Nets (CNN)



Deep, because: multiple hidden Layers

CNN: automatic Feature extraction

Why Neural Net : Surpasses Human



http://image-net.org/challnges/LSVRC/

Hyper Parameter

- Can influence the speed in which learning takes place
- Can impact the accuracy of the model
- Examples: Learning rate, decay rate, batch size

- Epoch – complete pass through the training dataset

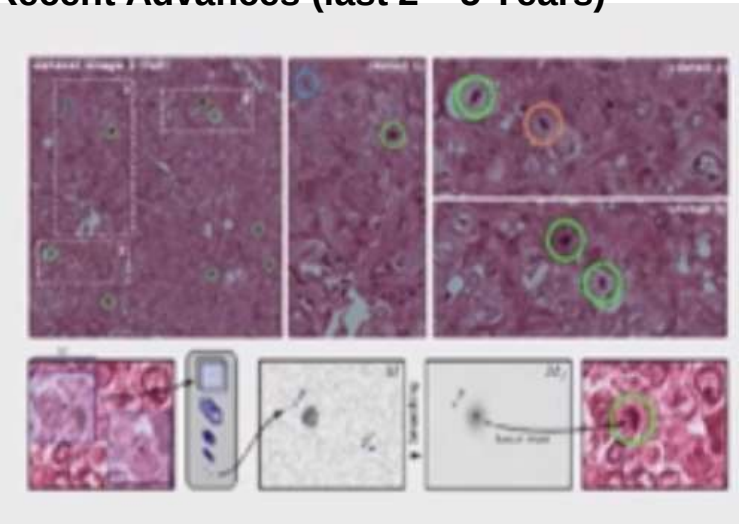
Status of Deep Learning

Five broad Categories of AI (Mc Kinsey)

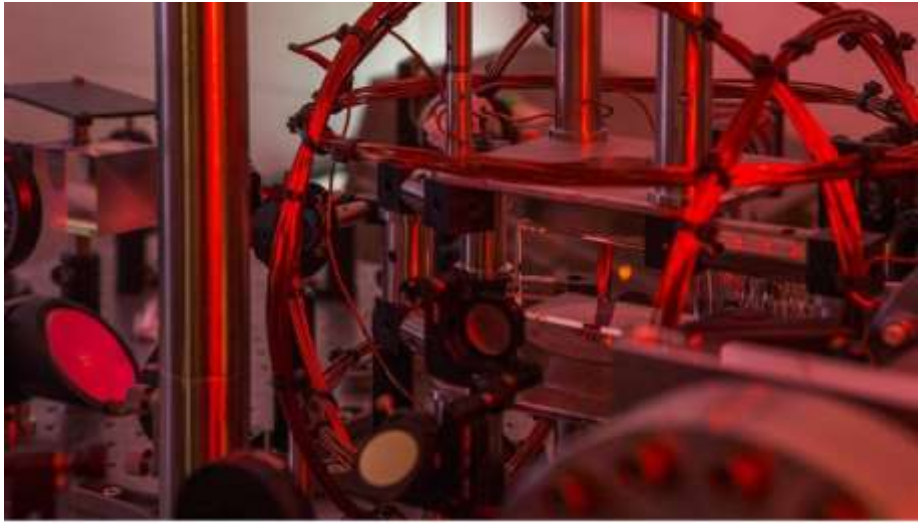
- Computer Vision
- Natural Language
- Virtual Assistants
- Robotic Process Automation
- Advanced Machine Learning

Advanced Machine Learning

Recent Advances (last 2 – 5 Years)



A Neural Net discovers Breast Cancer before it appears.
Discovers Things not known before

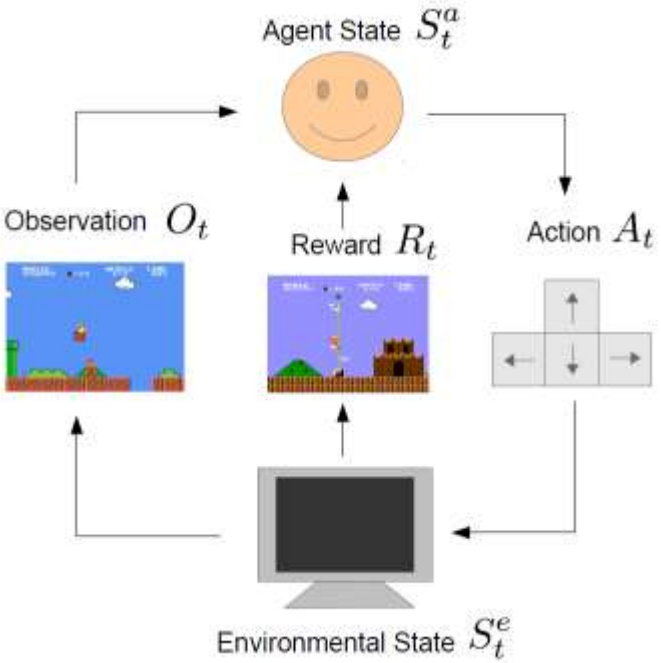


Artificial Intelligence Recreates Nobel Prize-Winning Physics Experiment - In One Hour - Using Methods a Human would not think of

Source: Forbes

First Time in History artificial Systems finding Ways and Knowledge not known to Human

Deep Reinforcement Learning



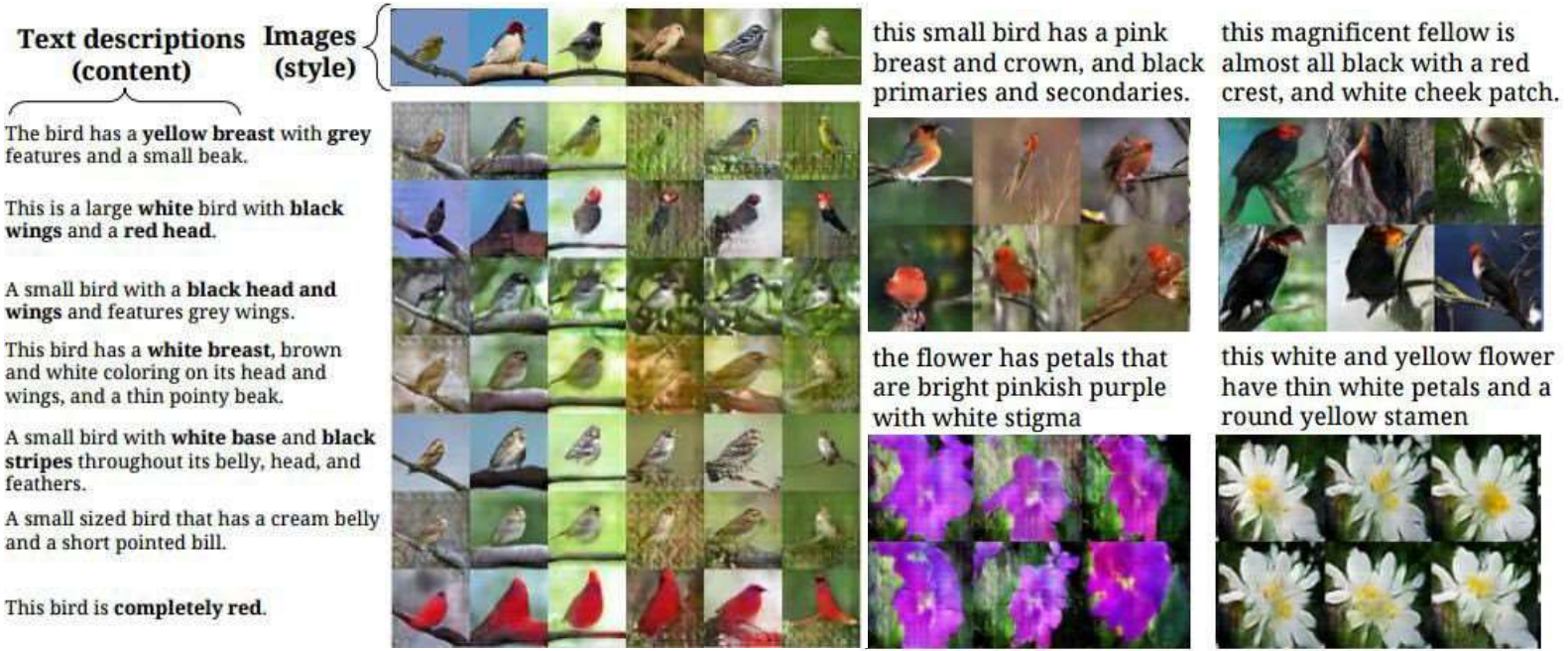
- No Supervisor / Labels
- Only a Reward as Feedback
- Playground : Elon Musk Openai.org

AlphaGo

See also:

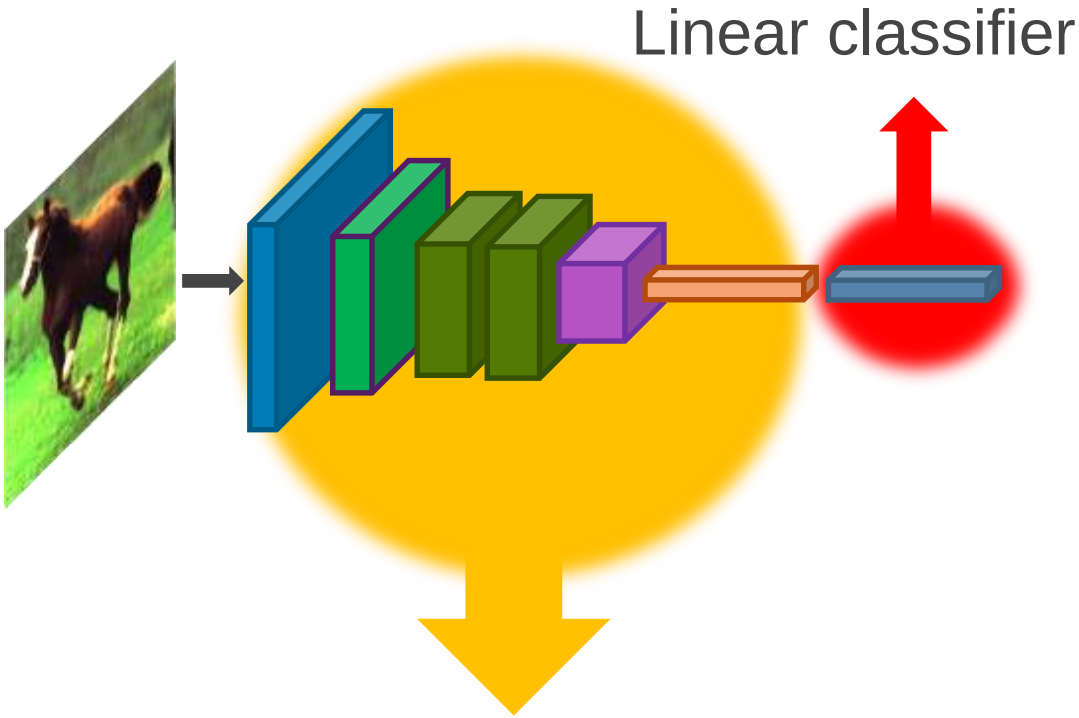
MIT Techreview 10 Breakthrough Technologies 2017 – Reinforcement Learning

Generative adversal Networks

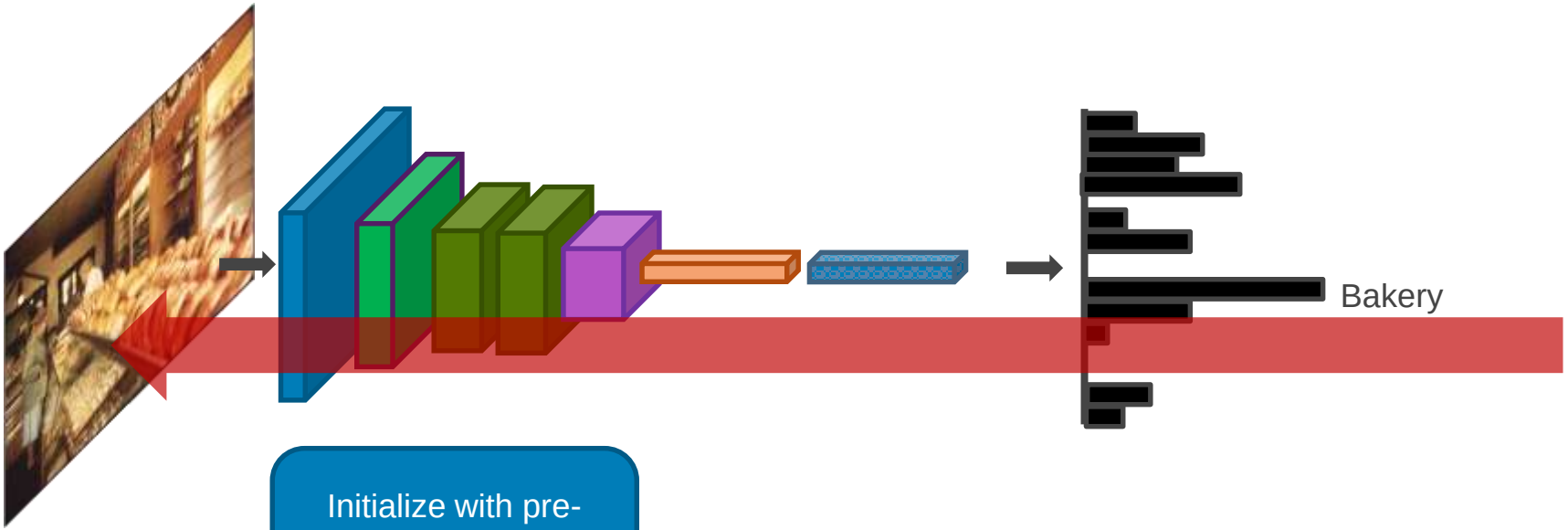


Transfer Learning

- What do we do for a new image classification problem?
- Key idea:
  - *Freeze* parameters in feature extractor
  - *Retrain* classifier



Fine Tuning



Initialize with pre-trained, then train with low learning rate


DELL EMC