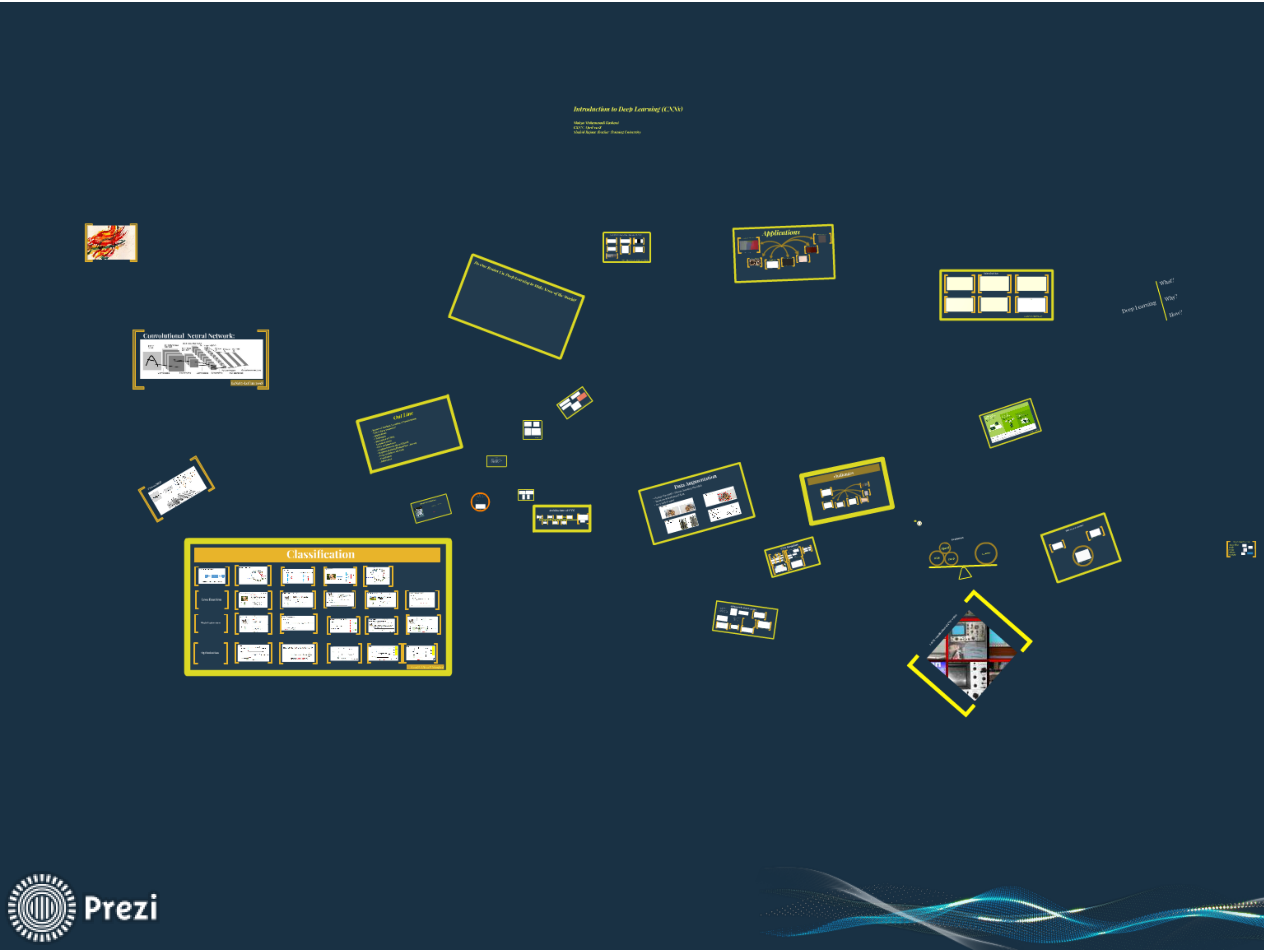
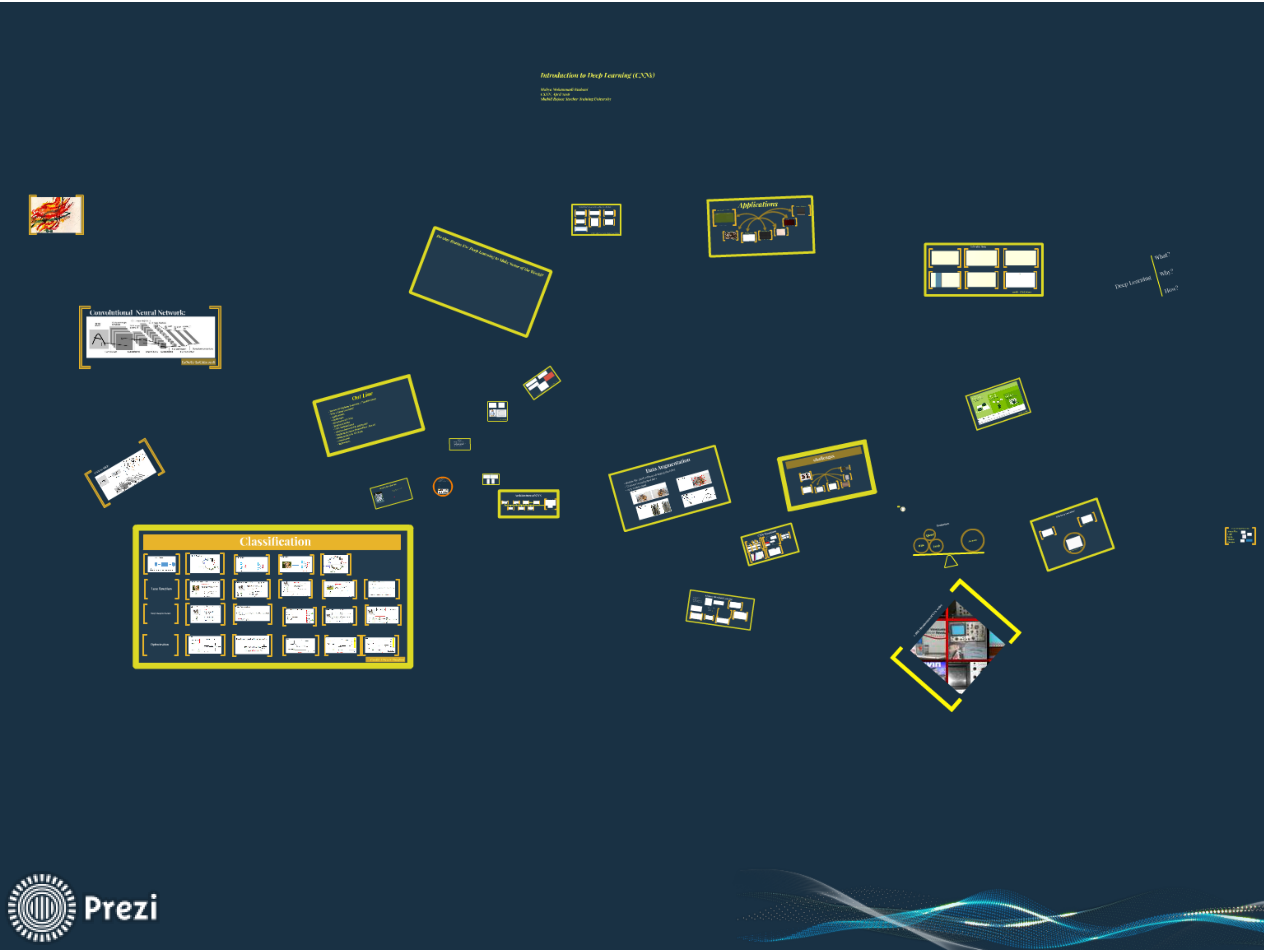


Introduction to Deep Learning (CNNs)
Applications
Deep Learning
What?
Why?
How?
Convolutional Neural Network:
Out Line
Data Augmentation
Classification



Introduction to Deep Learning (CNNs)
Applications
Deep Learning
What?
Why?
How?
Convolutional Neural Network:
Out Line
Data Augmentation
Classification
Loss function
Optimization
Prezi

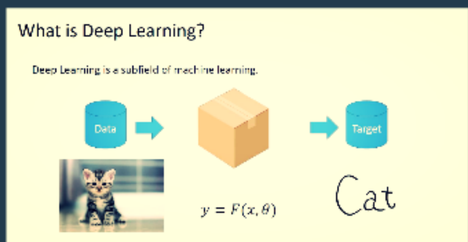
Introduction to Deep Learning (CNNs)

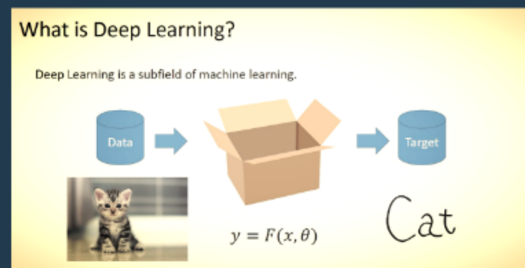
Mahya Mohammadi Kashani
CSNN, April 2018
Shahid Rajaee Teacher Training University

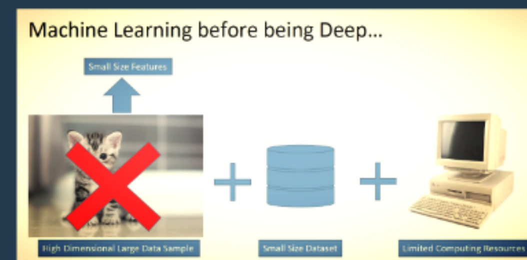
Out Line

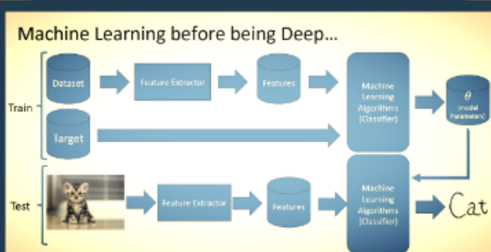
- Review of Machine Learning (Classification)
- Why is Deep Learning?
- Applications
- Challenges
- Structures of CNNs
- Learning tricks
- CNN Architectures
- relation between DL and Brain!
- Implementations (TensorFlow , Keras)
- Deep models in AIA Task
- Conclusion
- References

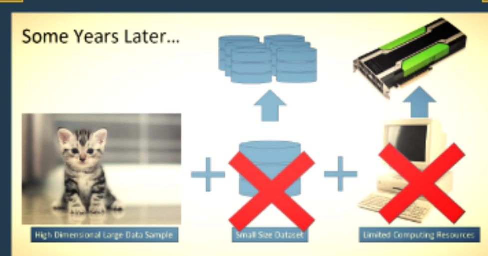
Introduction

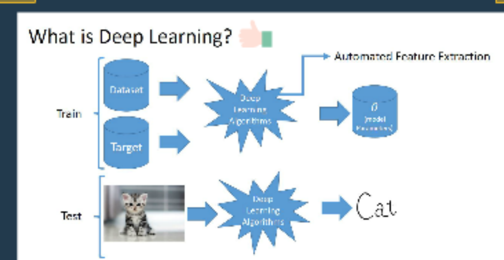










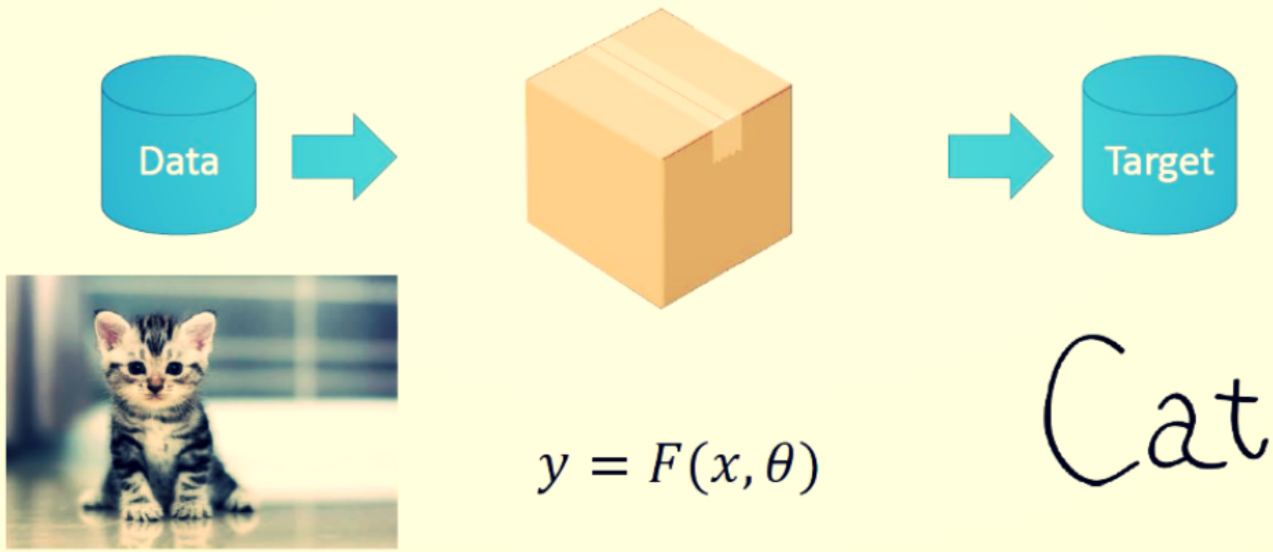


credit : UTDLSS2017

Prezi

What is Deep Learning?

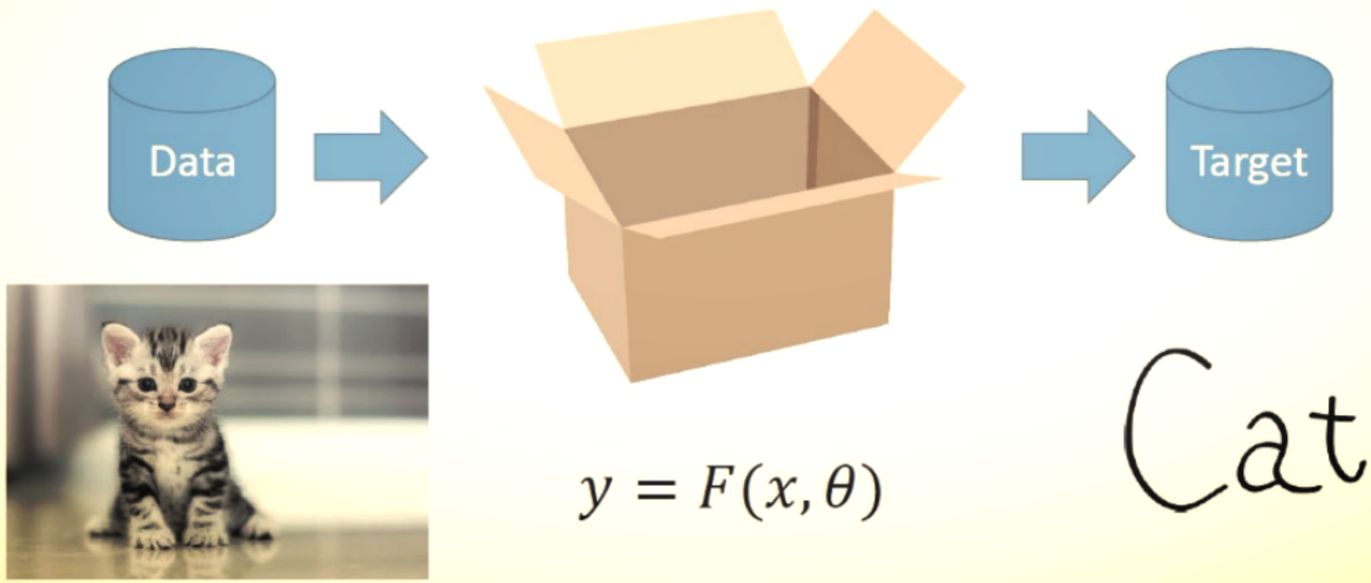
Deep Learning is a subfield of machine learning.



Introduction

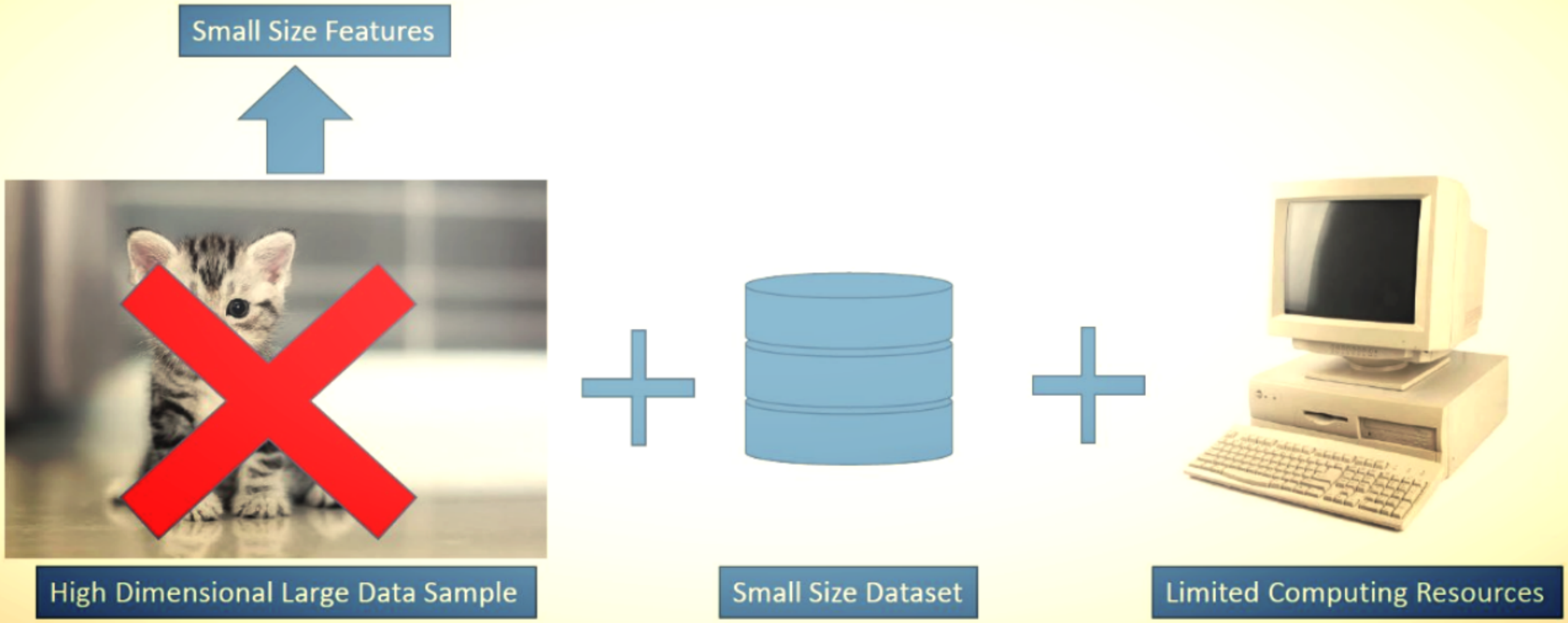
What is Deep Learning?

Deep Learning is a subfield of machine learning.

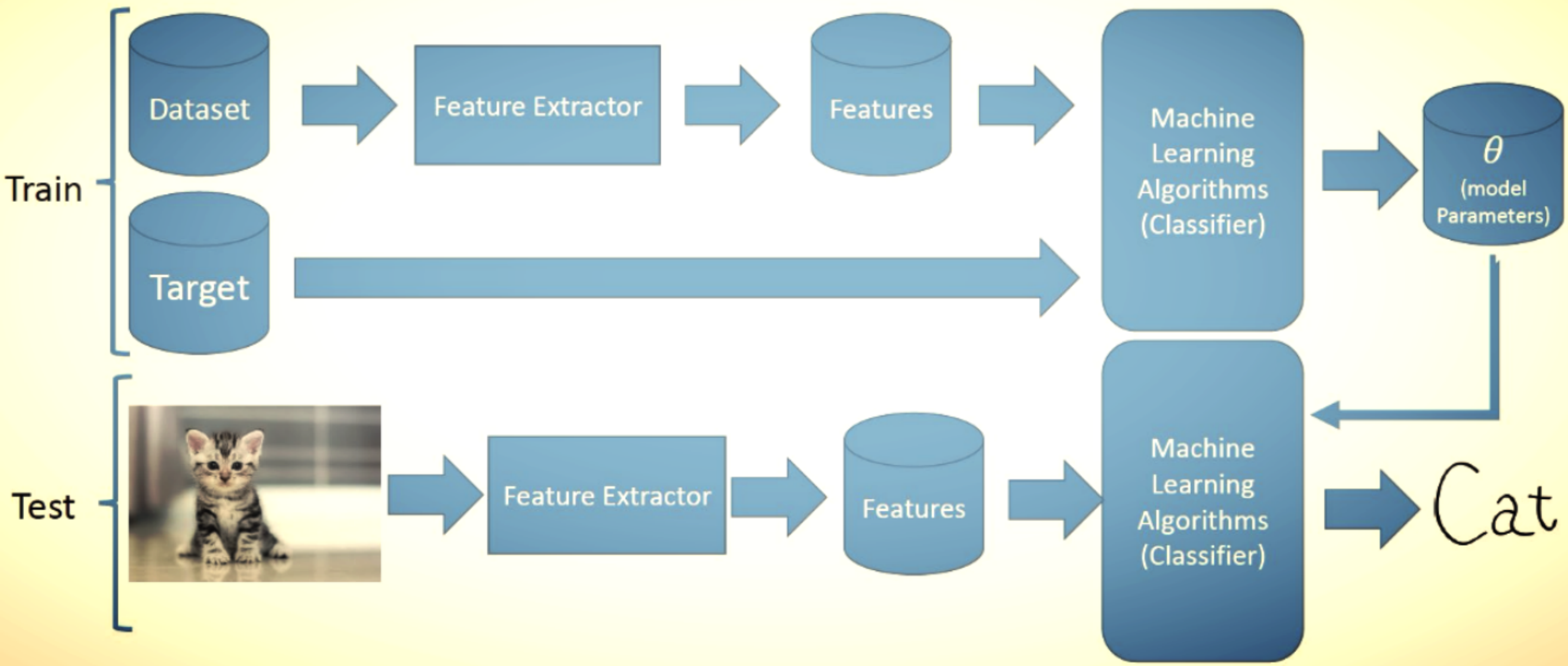


$$y = F(x, \theta)$$

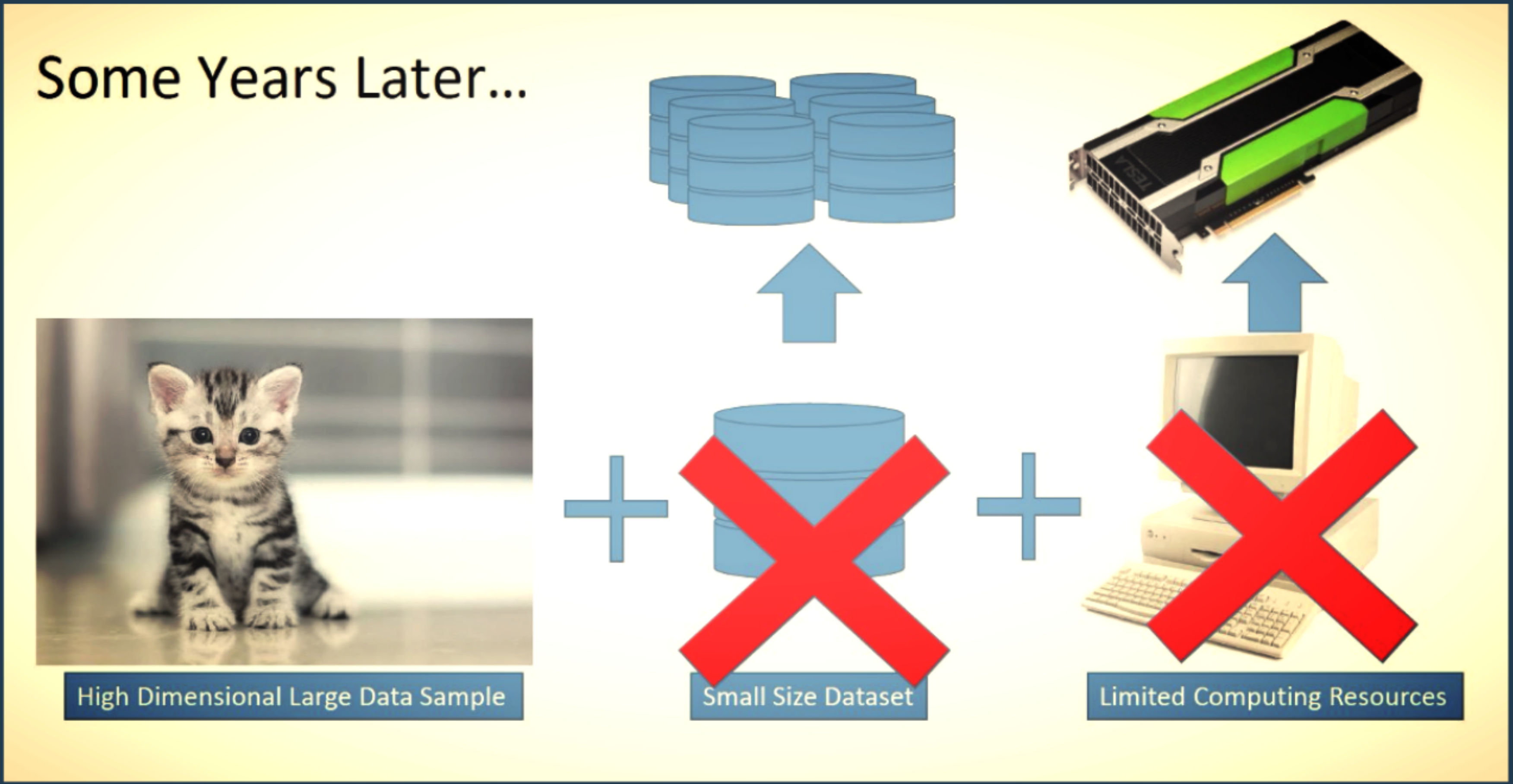
Machine Learning before being Deep...



Machine Learning before being Deep...

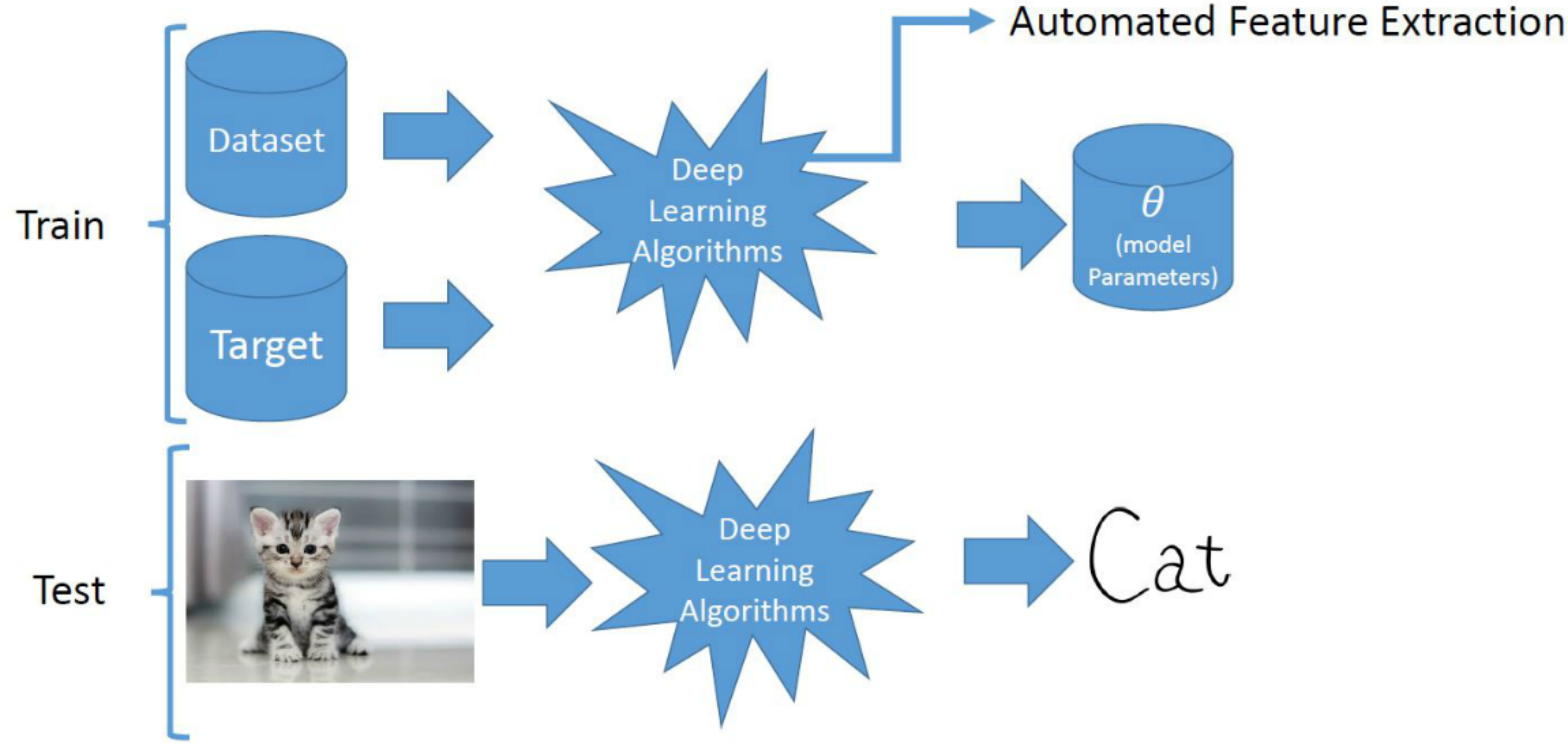


Prezi

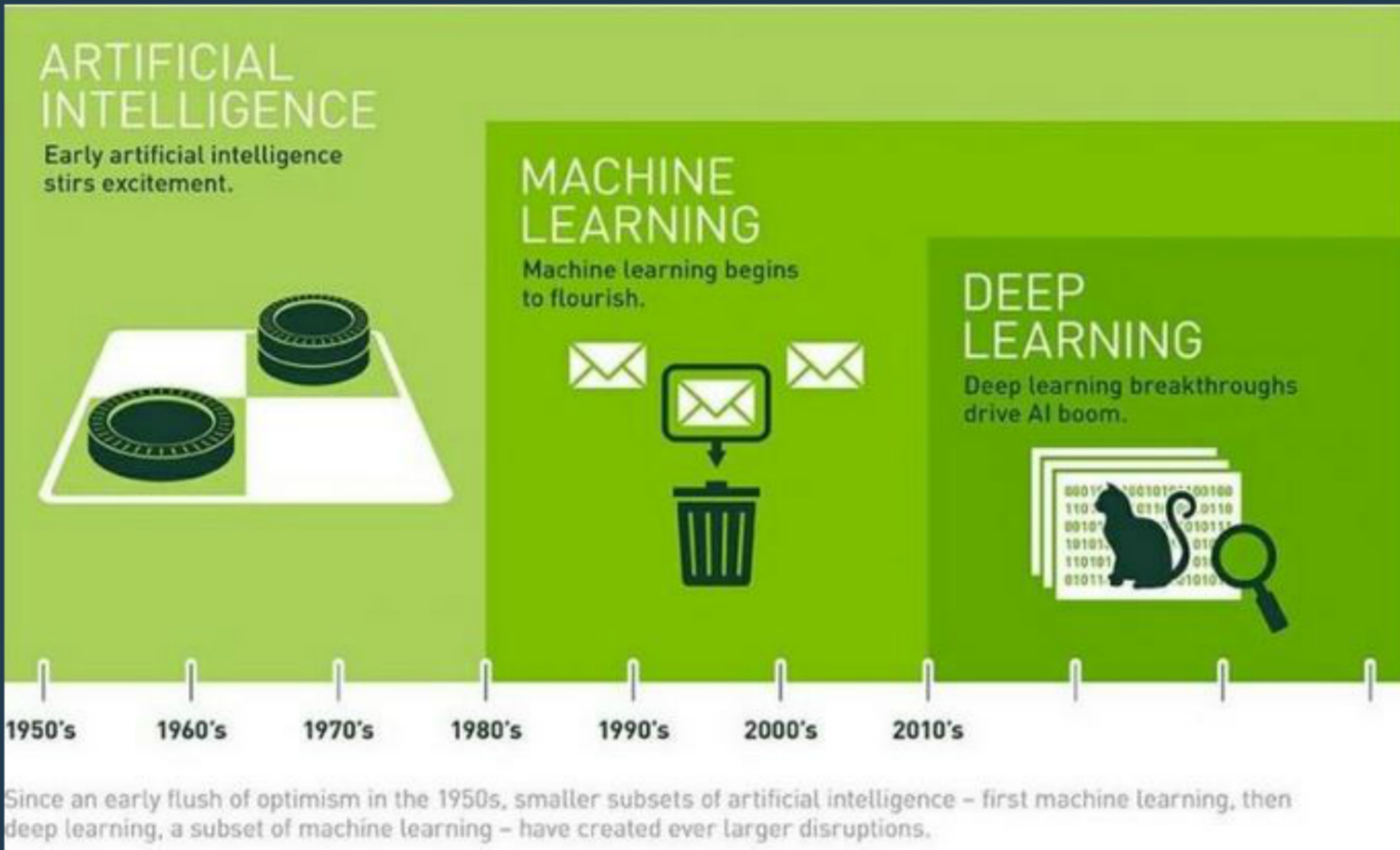

Some Years Later...
High Dimensional Large Data Sample
Small Size Dataset
Limited Computing Resources

Prezi

What is Deep Learning?



credit : UTDLSS2017



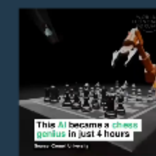
credit : Nvidia

Applications

Autonomous navigation of cars and drones

https://www.cityscapes-dataset.com/examples/

Alpha Zero : Genius of Chess



Deep Learning in Speech Separation and Recognition

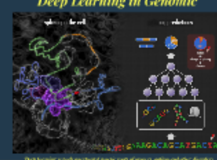
Deep Learning in Conversational Interface

Robots use deep learning

credit : Nvidia

Health Care

Deep Learning in Genomic



Autonomous navigation of cars and drones

https://www.cityscapes-dataset.com/examples/

Prezi

Autonomous navigation of cars and drones

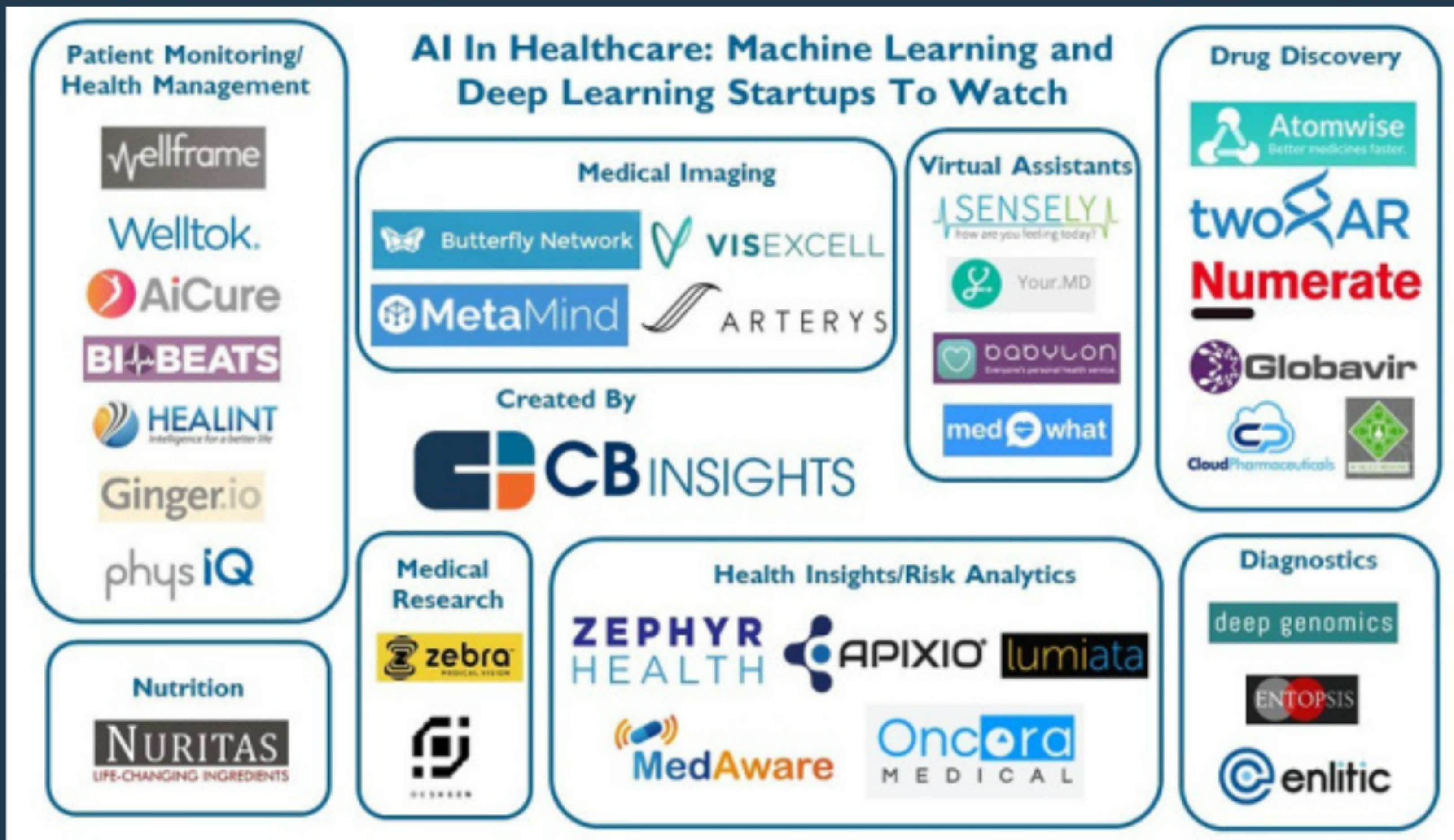
https://www.cityscapes-dataset.com/examples/

Prezi

Robots use deep learning

credit : Nvidia

Health Care



AI In Healthcare: Machine Learning and Deep Learning Startups To Watch
Patient Monitoring/ Health Management
Wellframe
Welltok.
AiCure
BIOBEATS
HEALINT
Ginger.io
physIQ
Nutrition
NURITAS
LIFE-CHANGING INGREDIENTS
Medical Imaging
Butterfly Network
VISEXCELL
MetaMind
ARTERYS
Created By
CB INSIGHTS
Medical Research
zebra
Virtual Assistants
SENSELY
Your.MD
babylon
med what
Drug Discovery
Atomwise
twoXAR
Numerate
Globavir
Cloud Pharmaceuticals
Health Insights/Risk Analytics
ZEPHYR HEALTH
APIXIO
lumiata
MedAware
Oncora MEDICAL
Diagnostics
deep genomics
ENTOPSIS
enlitic

Deep Learning in Conversational Interface

Prezi

Deep Learning in Speech Separation and Recognition

https://research.googleblog.com/2018/04/looking-to-listen-audio-visual-speech.html

Deep Learning in Speech Separation and Recognition

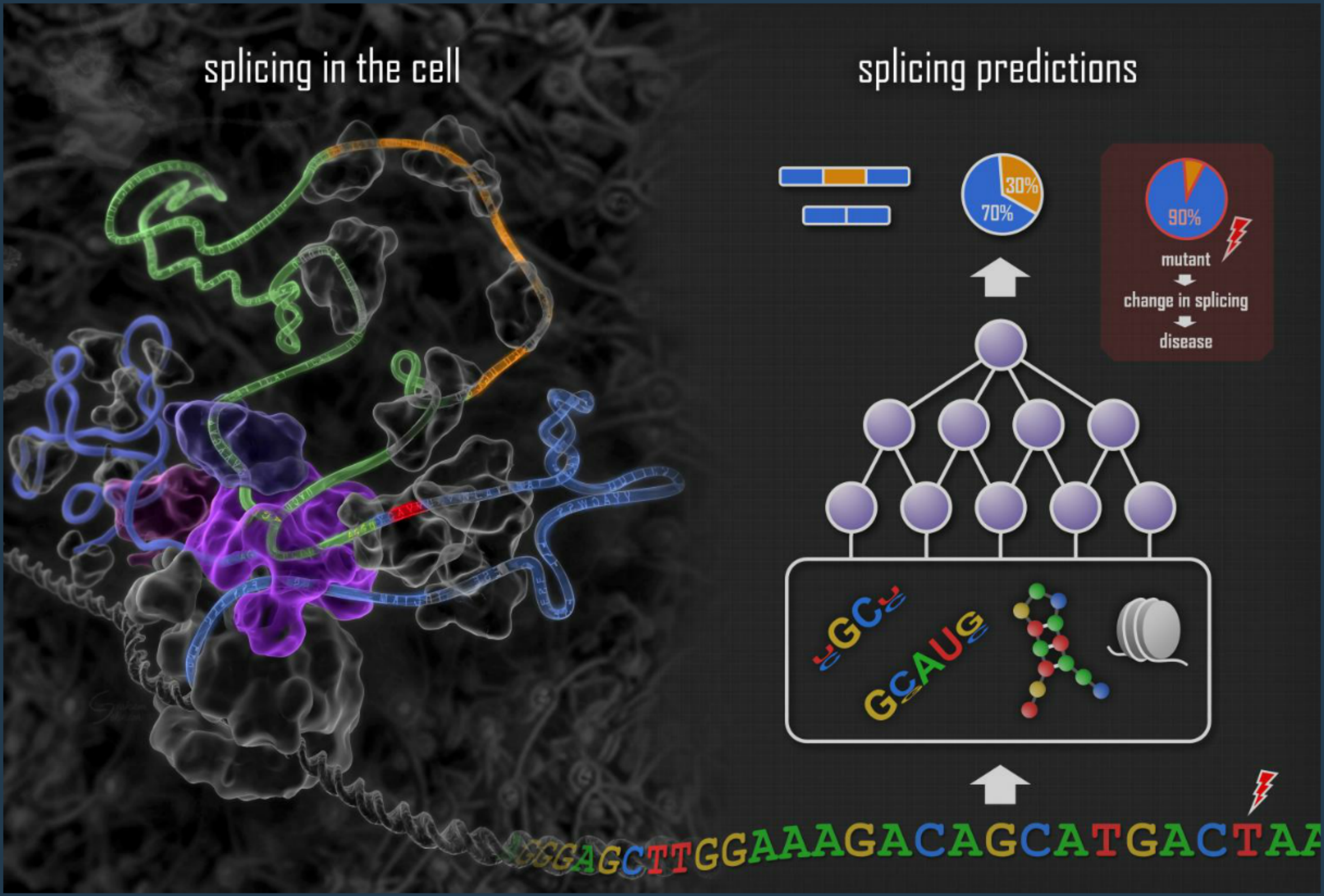
https://research.googleblog.com/2018/04/looking-to-listen-audio-visual-speech.html

Prezi

Alpha Zero : Genius of Chess


WORLD
ECONOMIC
FORUM
This AI became a chess genius in just 4 hours
Source: Cornell University

Deep Learning in Genomic



'Deep learning' reveals unexpected genetic roots of cancers, autism and other disorders

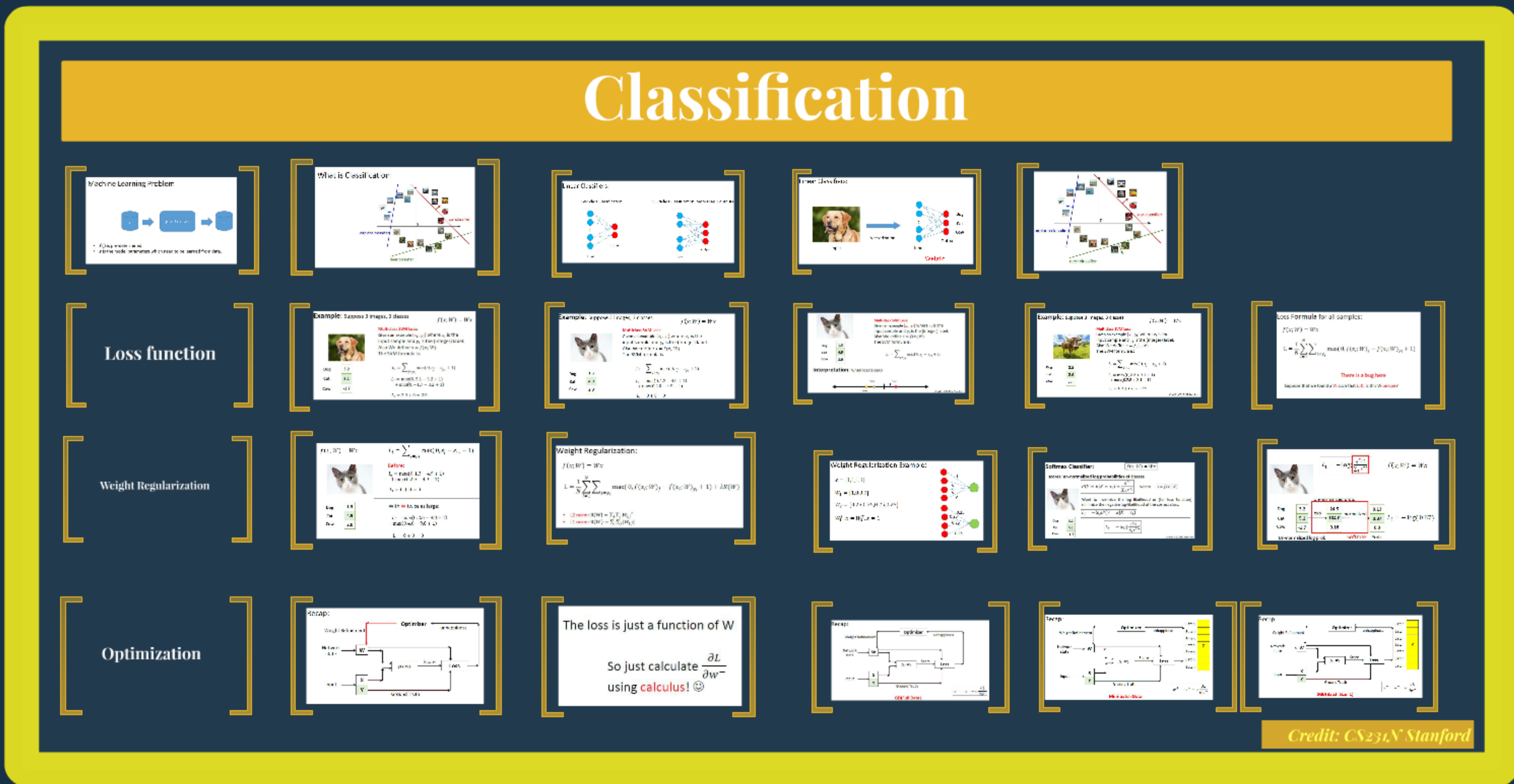
Prezi

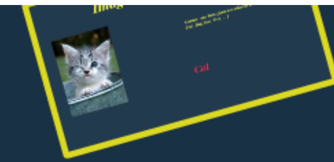
Image Classification

Assume: you have given a set dicrete labels :
{Cat, Dog, Bus, Tree, ... }

Cat

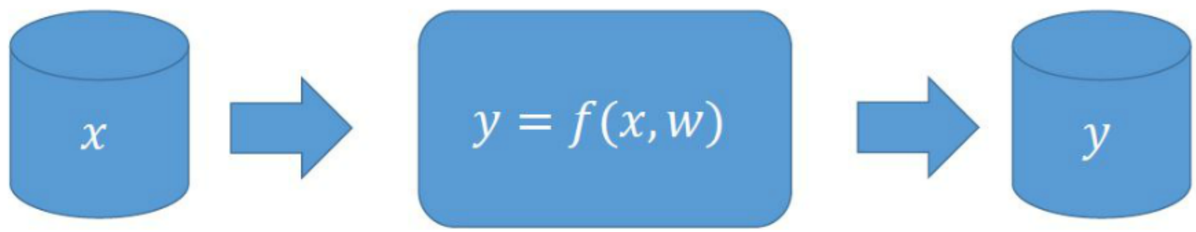
Prezi


Classification
Machine Learning Problem
What is Classification
Linear Classifiers
Loss function
Weight Regularization
Weight Regularization:
Weight Regularization Example:
Softmax Classifier:
Loss Formula for all samples:
Optimization
Recap:
The loss is just a function of W
So just calculate $\frac{\partial L}{\partial w}$
using calculus! ☺
Credit: CS231N Stanford



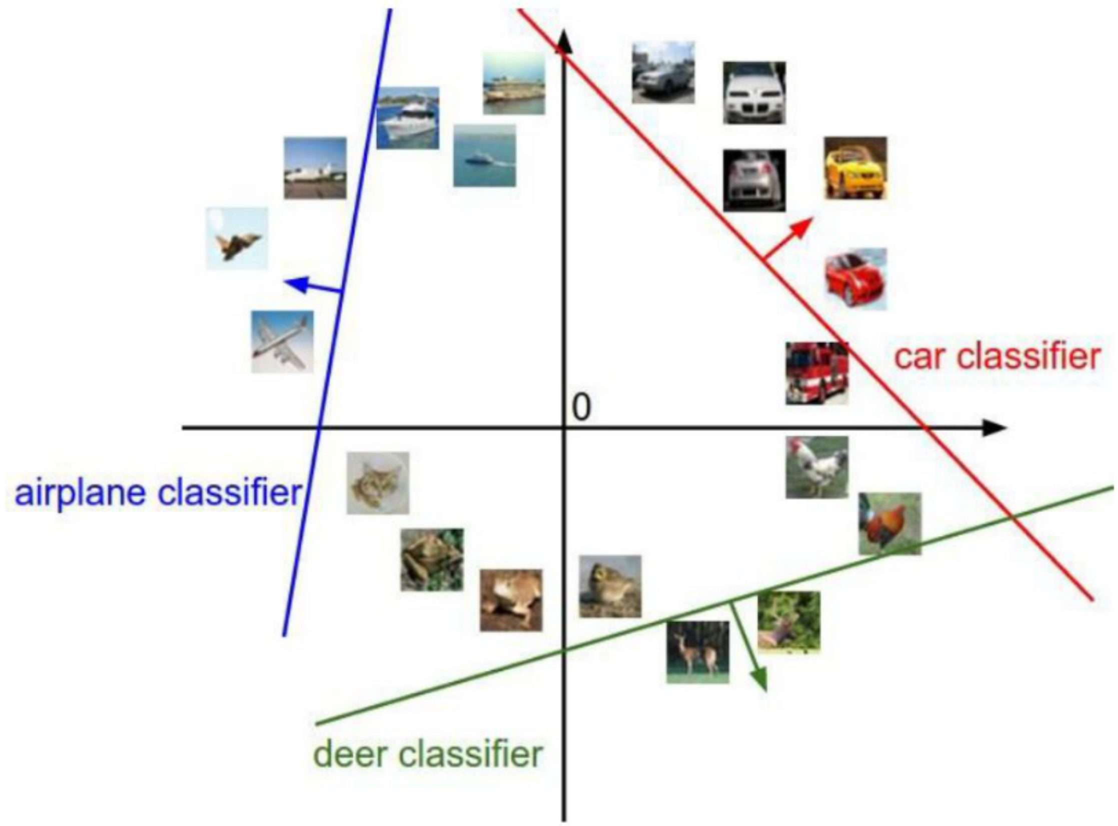
Prezi

Machine Learning Problem



- $f()$ is pre-determined.
- w is the model parameters which need to be learned from data.

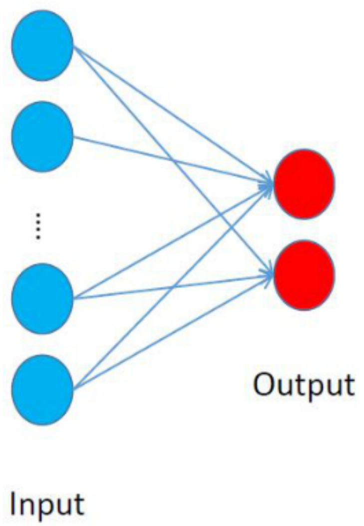
What is Classification:



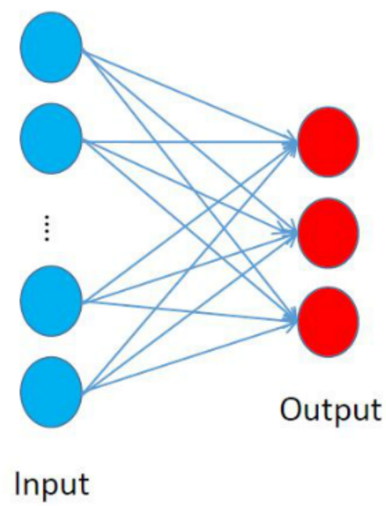
Prezi

Linear Classifiers:

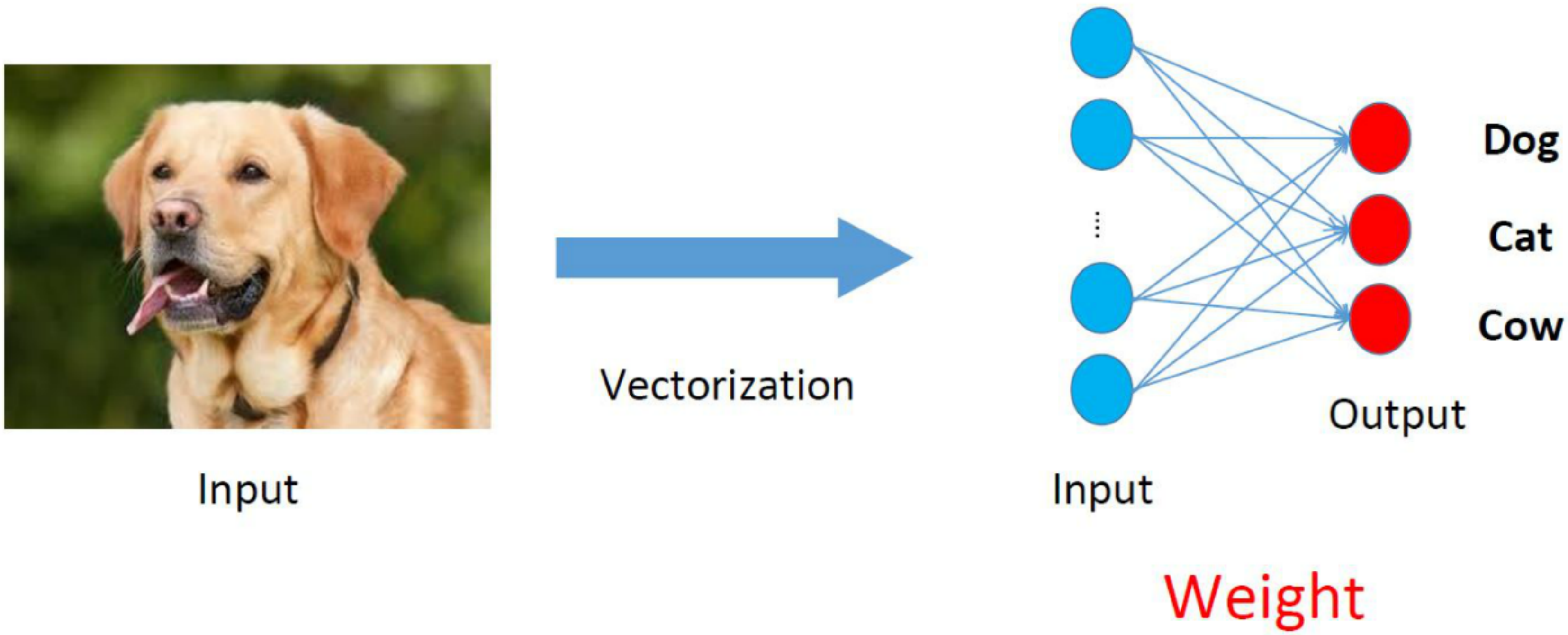
Two class Classification



Multiclass Classification: More than 2 outputs



Linear Classifiers:



Loss function

Example: Suppose 3 images, 3 classes

$$f(x; W) = Wx$$

Dog	**3.2**
Cat	**5.1**
Cow	**-1.7**

Multiclass SVM Loss:

Given an example (x_i, y_i) where x_i is the input sample and y_i is the (integer) label.

Also We define $s = f(x_i; \mathrm{W})$

The SVM formula is:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L_i = \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$$

$$L_i = 2.9 + 0 = 2.9$$

Example: Suppose 3 images, 3 classes

$$f(x;W) = Wx$$

Dog	1.3
Cat	4.9
Cow	2.0

Multiclass SVM Loss:

Given an example (x_i, y_i) where x_i is the input sample and y_i is the (integer) label.

Also We define $s = f(x_i; \mathrm{W})$

The SVM formula is:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L_i = \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$L_i = 0 + 0 = 0$$

Prezi

Dog	1.3
Cat	4.9
Cow	2.0

Multiclass SVM Loss:

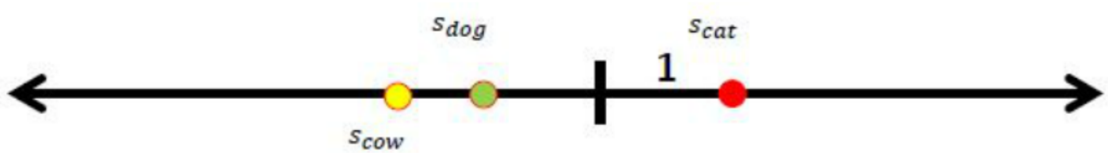
Given an example (x_i, y_i) where x_i is the input sample and y_i is the (integer) label.

Also We define $s = f(x_i; W)$

The SVM formula is:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Interpretation: when loss is zero



Credit: CS231N Stanford

Example: Suppose 3 images, 3 classes

$$f(x; W) = Wx$$

Multiclass SVM Loss:

Given an example (x_i, y_i) where x_i is the input sample and y_i is the (integer) label.

Also We define $s = f(x_i; W)$

The SVM formula is:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Dog	2.2
Cat	2.5
Cow	-3.1

$$L_i = \max(0, 2.2 + 3.1 + 1) + \max(0, 2.5 + 3.1 + 1)$$

$$L_i = 6.3 + 6.6 = 12.9$$

Credit: CS231N Stanford

Loss **Formula** for all samples:

$$f(x; W) = Wx$$

$$\mathrm{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

There is a bug here

Suppose that we found a W such that L=0, is this W unique?

Weight Regularization

$$f(x; W) = Wx$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Dog	1.3
Cat	4.9
Cow	2.0

Before:

$$L_i = \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$

$$L_i = 0 + 0 = 0$$

With W twice as large:

$$L_i = \max(0, 2.6 - 9.8 + 1) + \max(0, 4.0 - 9.8 + 1)$$

$$L_i = 0 + 0 = 0$$

Weight Regularization:

$$f(x; W) = Wx$$

$$\mathrm{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$$

- L2 norm: $\mathrm{R(W)} = \sum_i \sum_j {W_{i,j}}^2$
- L1 norm: $\mathrm{R}(W) = \sum_i \sum_j |W_{i,j}|$

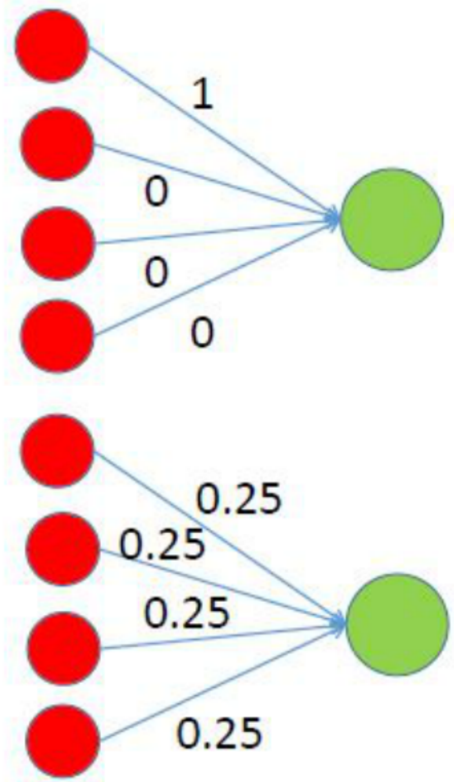
Weight Regularization Example:

$$x = [1,1,1,1]$$

$$W_1 = [1,0,0,0]$$

$$W_2 = [0.25,0.25,0.25,0.25]$$

$$W_1^T . x = W_2^T . x = 1$$



Softmax Classifier:

$$\boxed{f(x; W) = Wx}$$

scores: un-normalized log probabilities of classes

$$\boxed{P(Y = k|X = x_i) = \frac{e^{s_i}}{\sum_j e^{s_j}}} \quad \text{where} \quad \boxed{s = f(x_i; W)}$$

Want to maximize the log likelihood or (for loss function) minimize the negative log-likelihood of the correct class:

$$\boxed{L_i = -logP(Y = k|X = x_i)}$$

Dog	3.2
Cat	5.1
Cow	-1.7

$$\boxed{L_i = -\log(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}})}$$

Credit: CS231N Stanford

Prezi

$$L_i = -\log\left(\boxed{\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}\right)$$

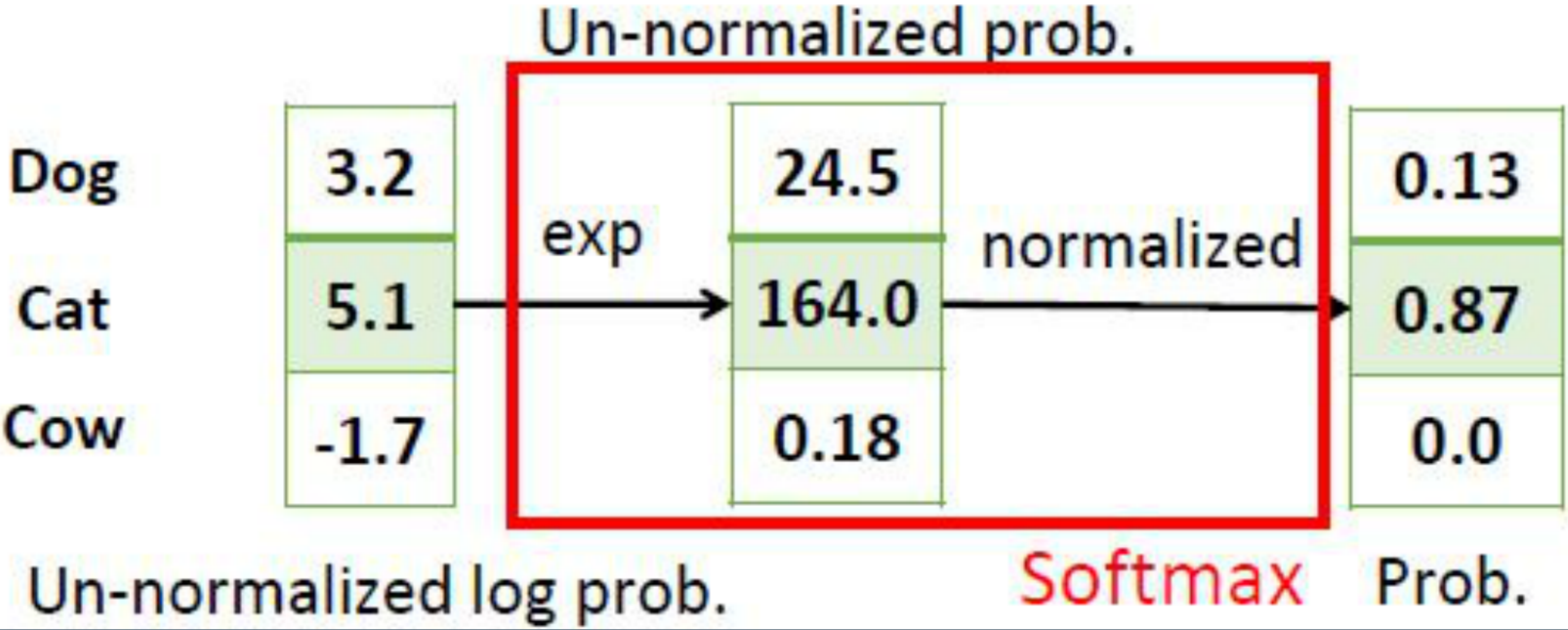
$$f(x;W) = Wx$$

	Un-normalized log prob.	exp	Un-normalized prob.	normalized	Prob.
Dog	3.2		24.5		0.13
Cat	5.1	→	164.0	→	0.87
Cow	-1.7		0.18		0.0

Softmax

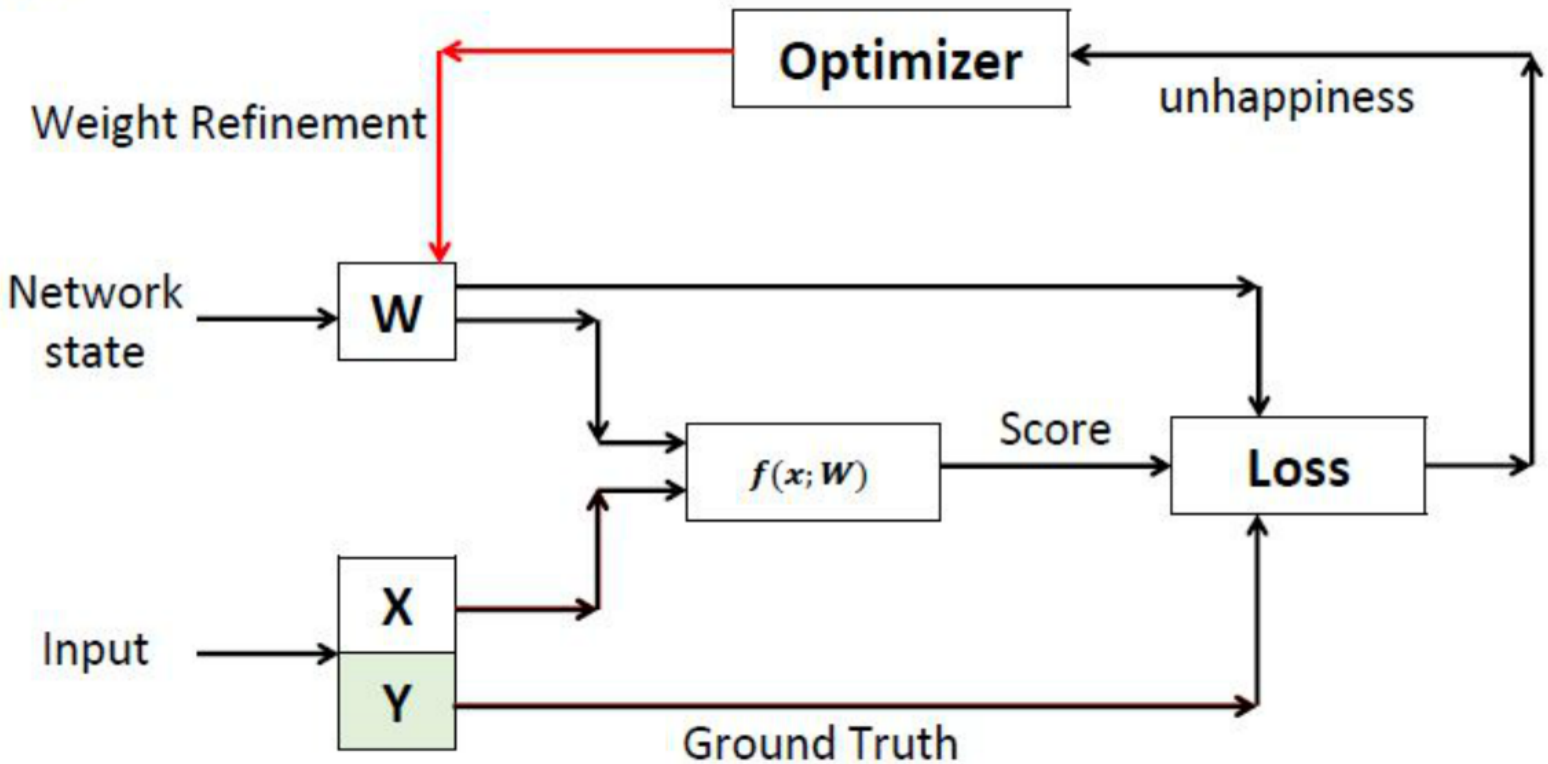
$$L_i = -\log(0.87)$$

$$L_i = -\log\left(\boxed{\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}}\right)$$


Un-normalized prob.
Dog
Cat
Cow
3.2
5.1
-1.7
exp
24.5
164.0
0.18
normalized
0.13
0.87
0.0
Un-normalized log prob.
Softmax
Prob.

Optimization

Recap:

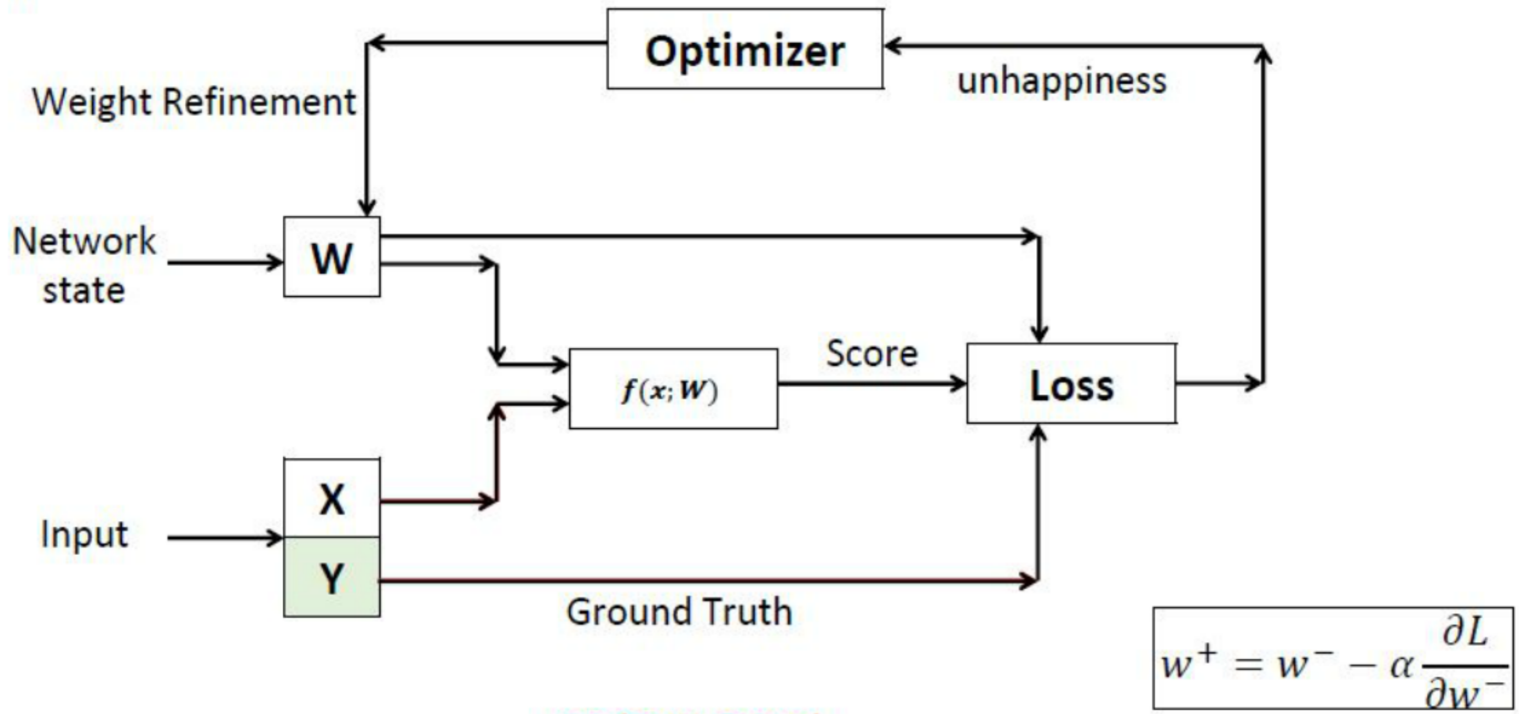


Optimizer
Weight Refinement
unhappiness
Network state
W
$f(x; W)$
Score
Loss
X
Input
Y
Ground Truth

The loss is just a function of W

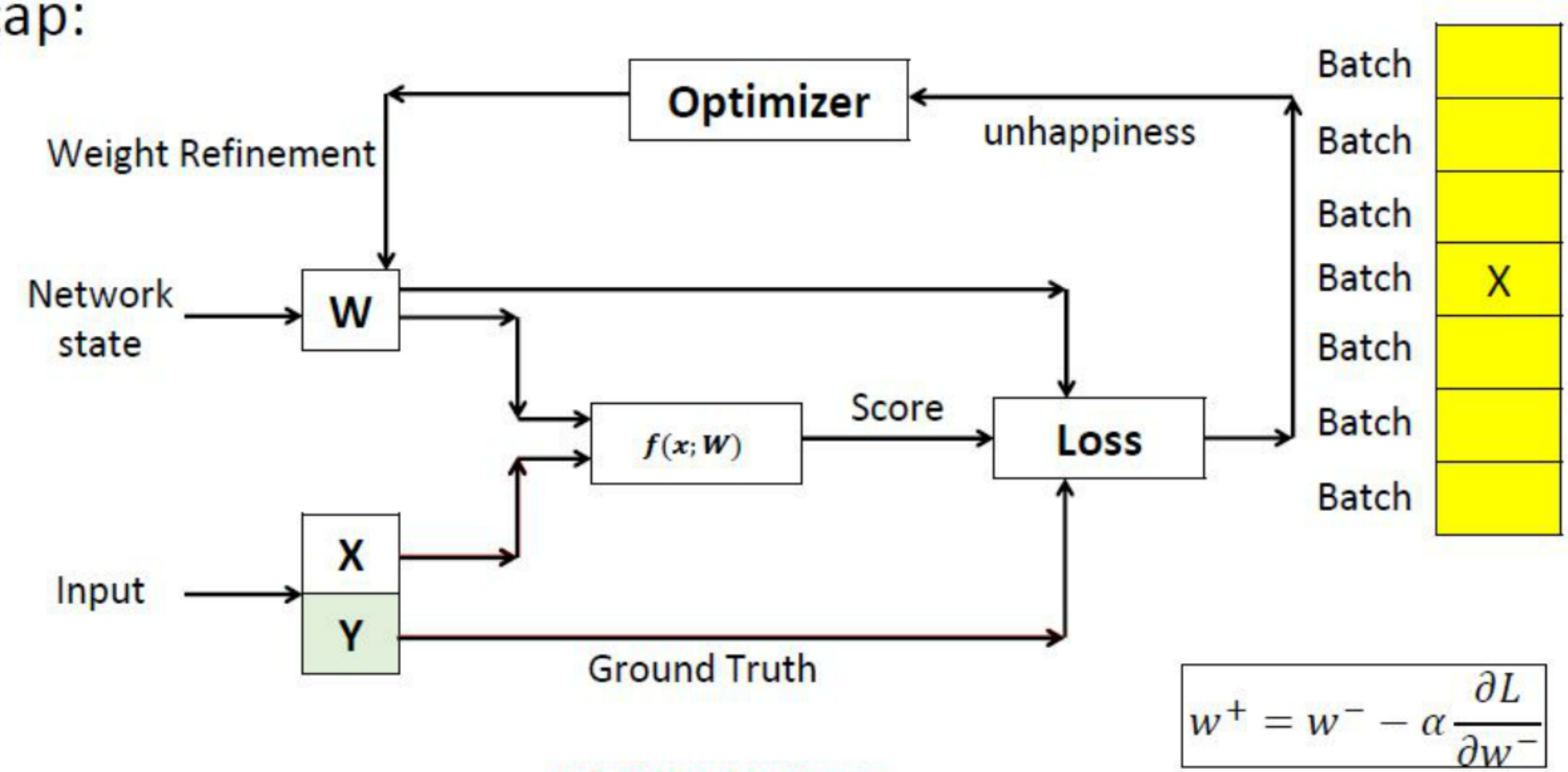
So just calculate $\frac{\partial L}{\partial w}$ using calculus! ☺

Recap:



GD(Full Data)

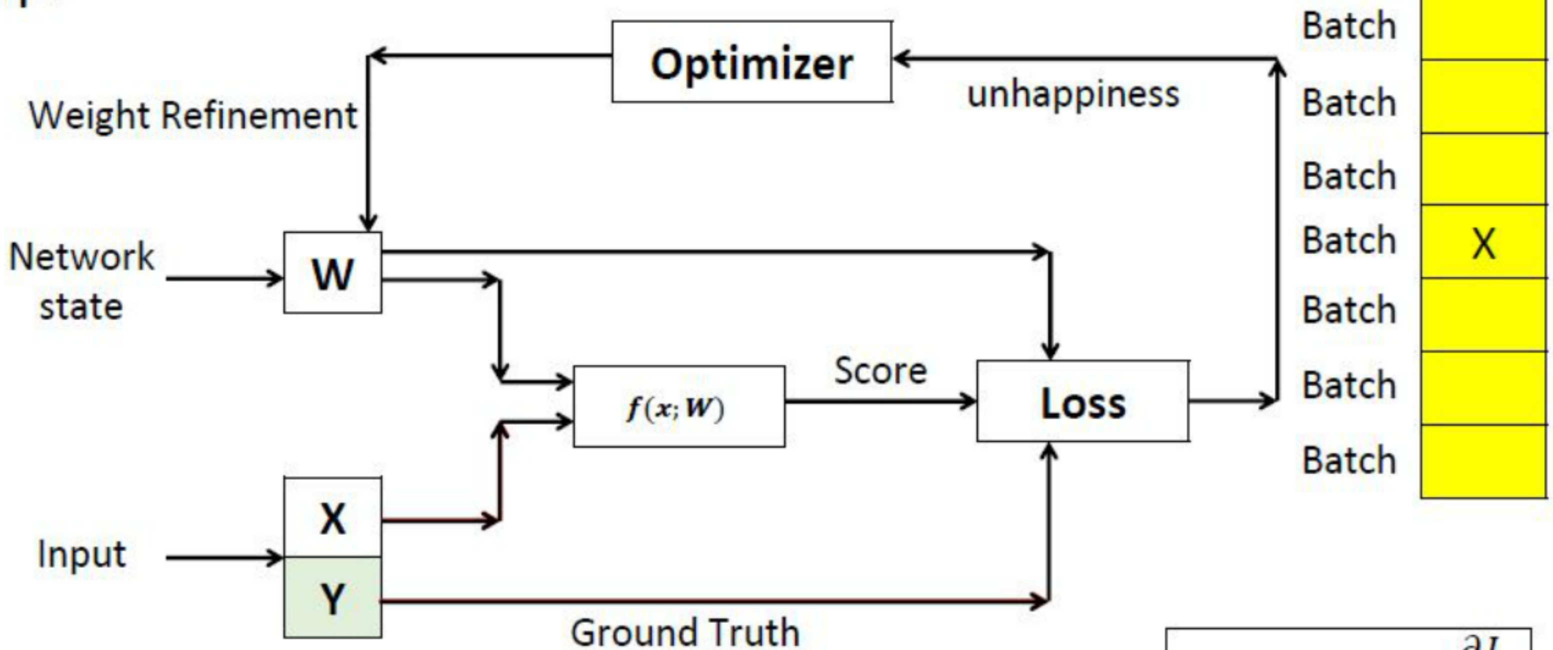
Recap:



Mini-batch Data

Prezi

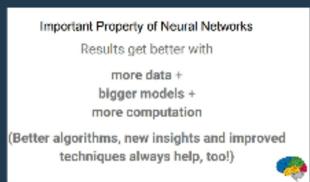
Recap:



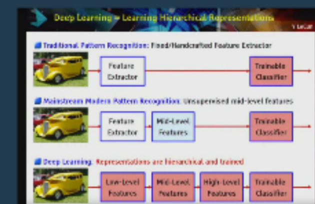
$$w^+ = w^- - \alpha \frac{\partial L}{\partial w^-}$$

SGD(Batch Size=1)

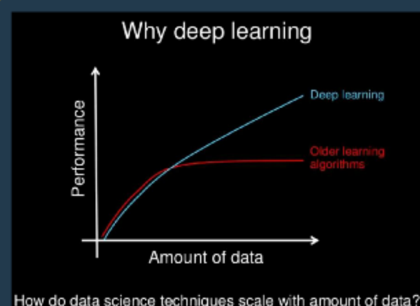
Why Deep Learning?



Result Get Better With More Data, Larger Model, More Computation, Slide by Jeff Dean

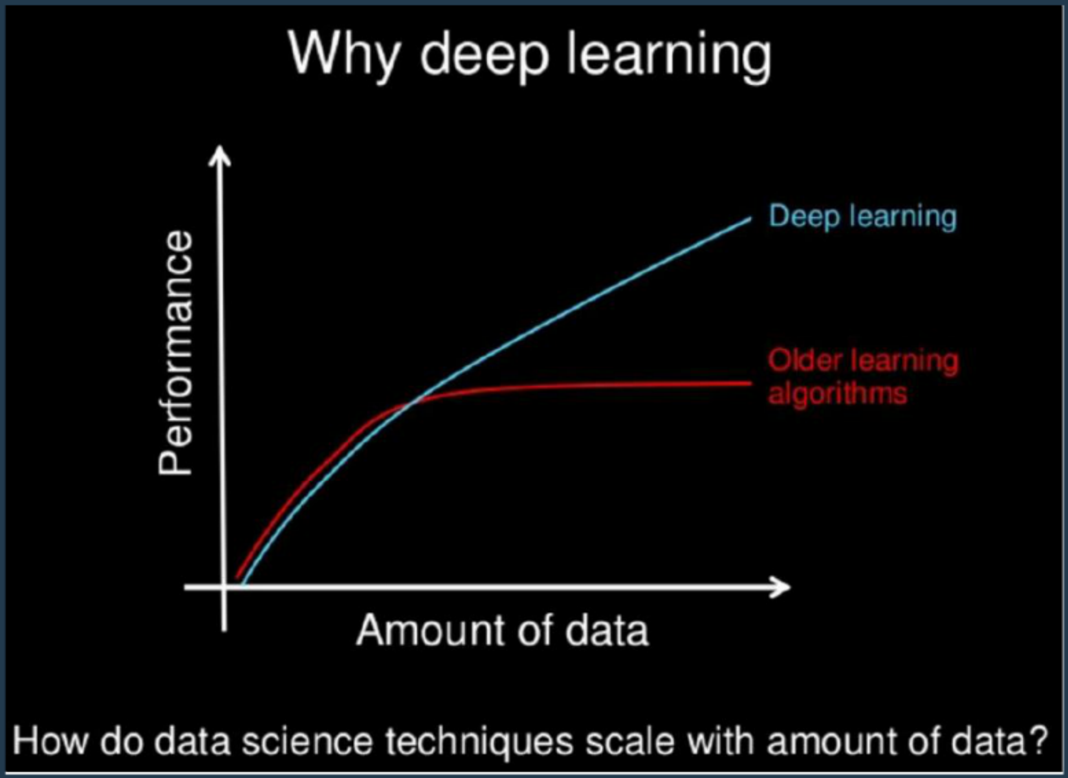


Deep Learning = Learning Hierarchical Representations Slide by Yann LeCun



Why Deep learning, Slide by Andrew NG

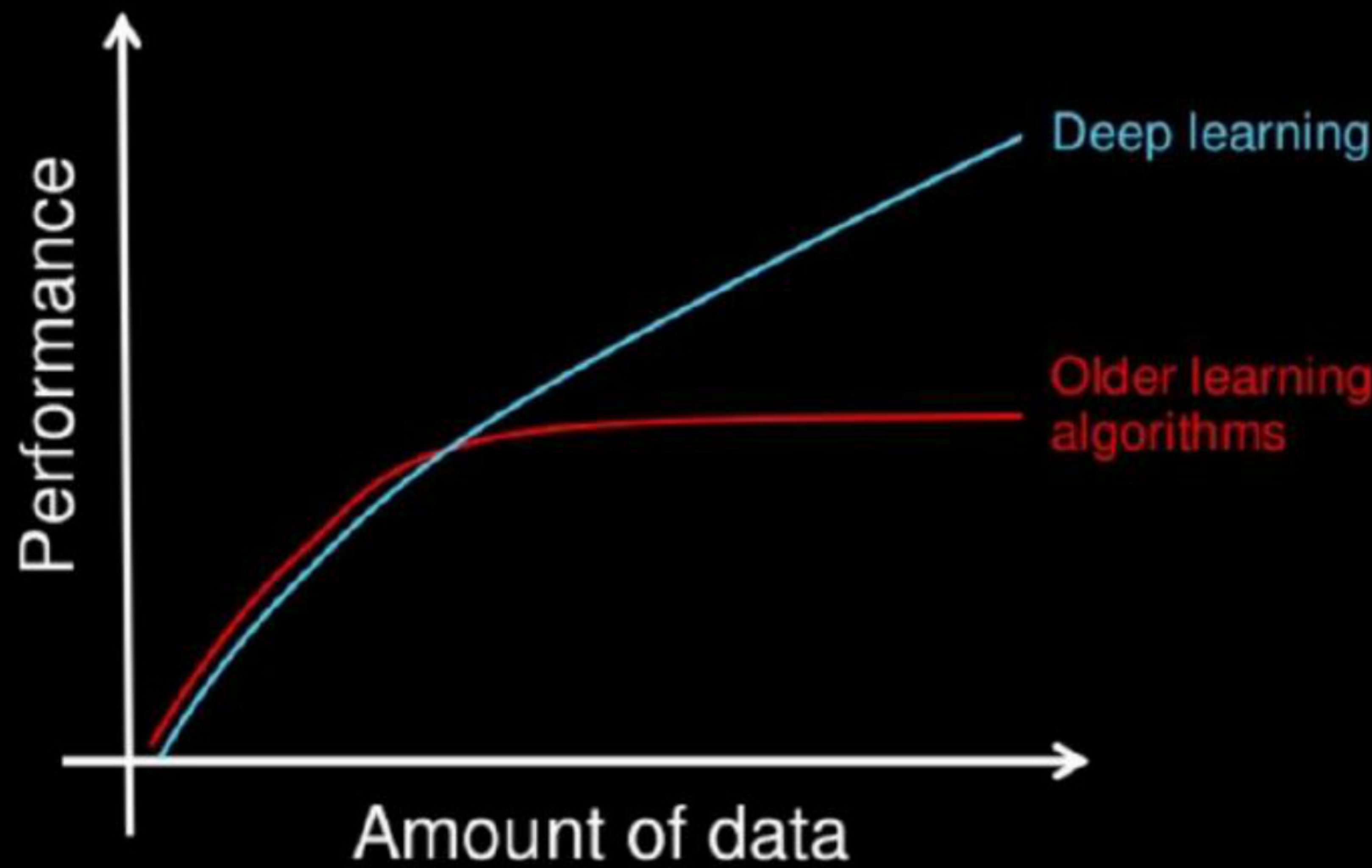
Prezi



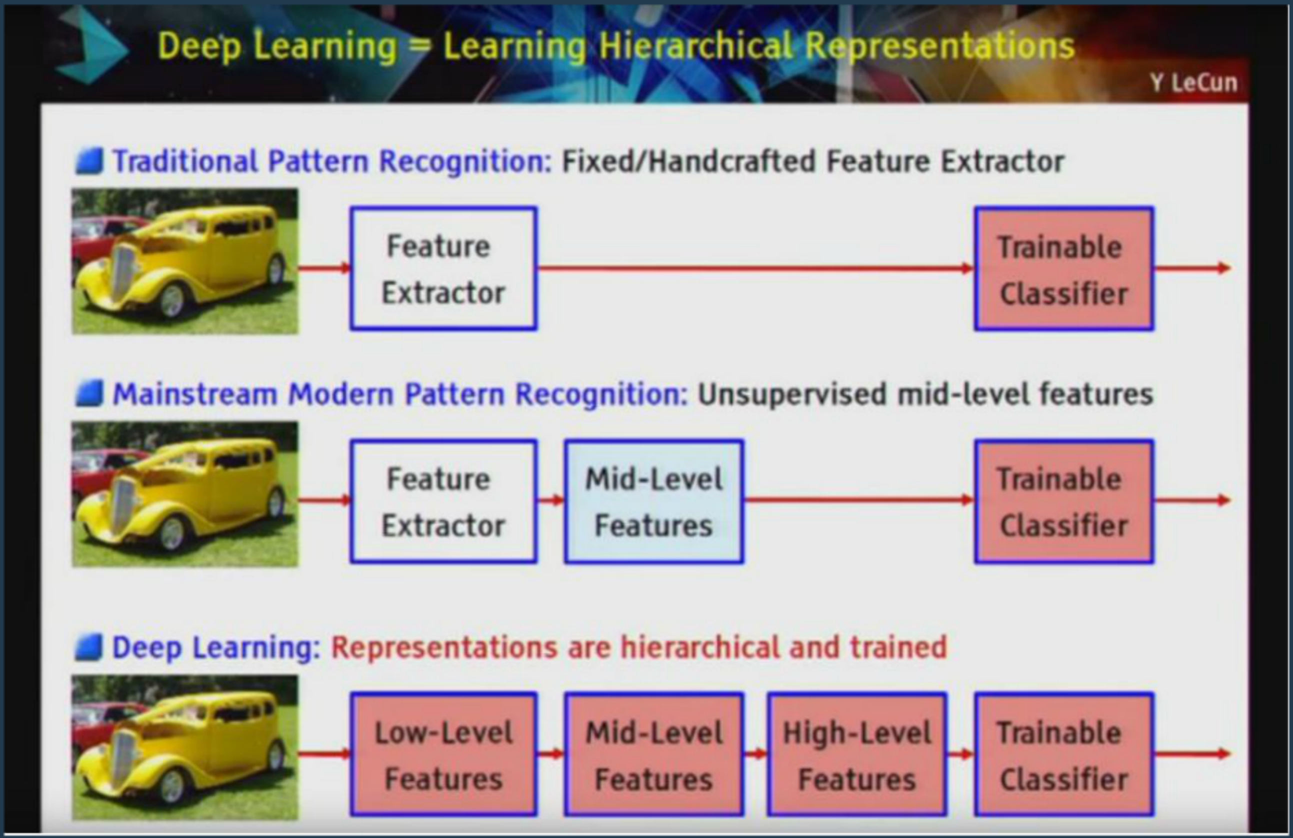
Why Deep learning, Slide by Andrew NG

Prezi

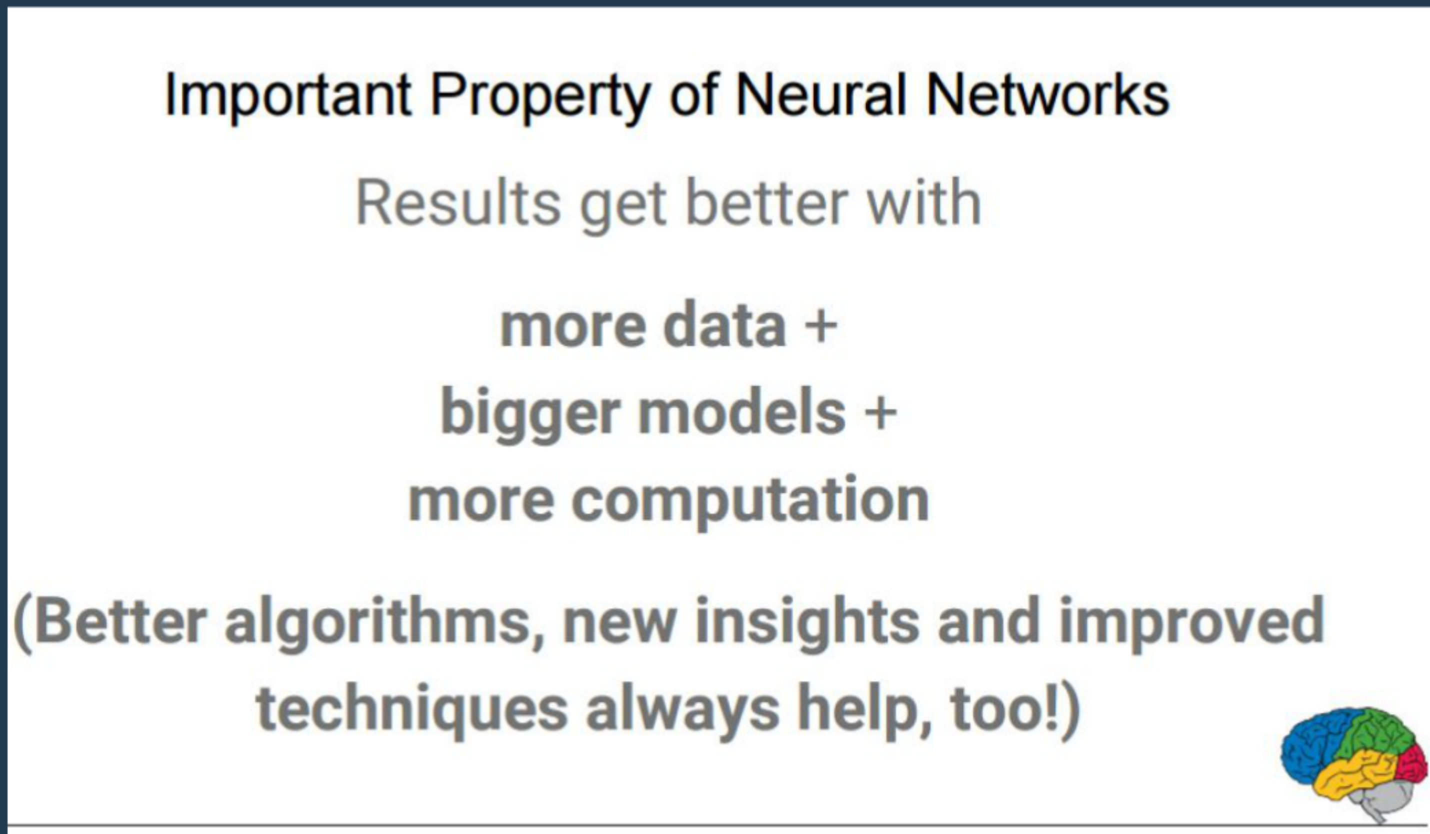
Why deep learning



 …w do data science techniques scale with amount of data?

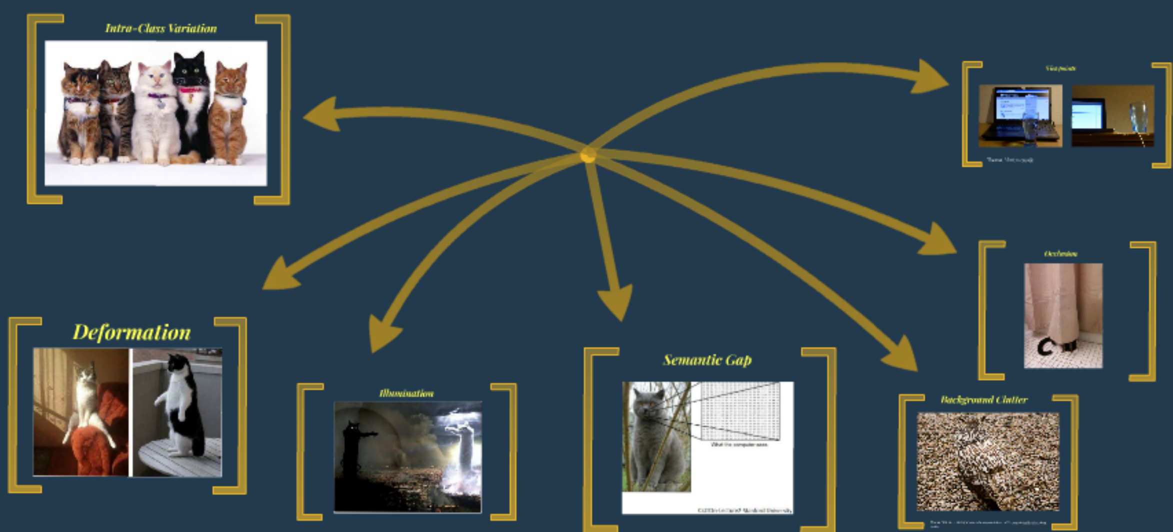


Deep Learning = Learning Hierarchical Representations Slide by Yann LeCun



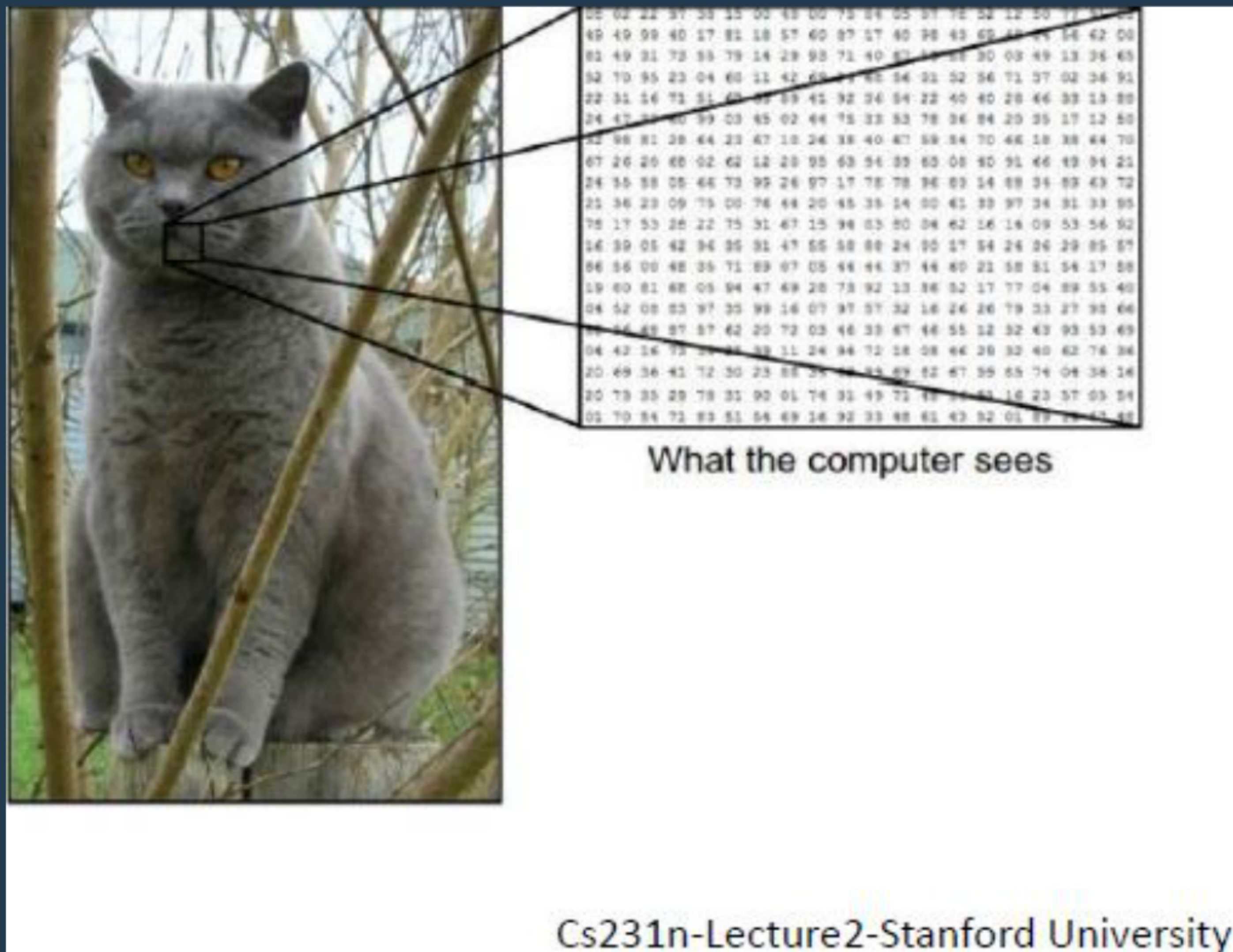
Result Get Better With More Data, Larger Model, More Computation, Slide by Jeff Dean

challenges

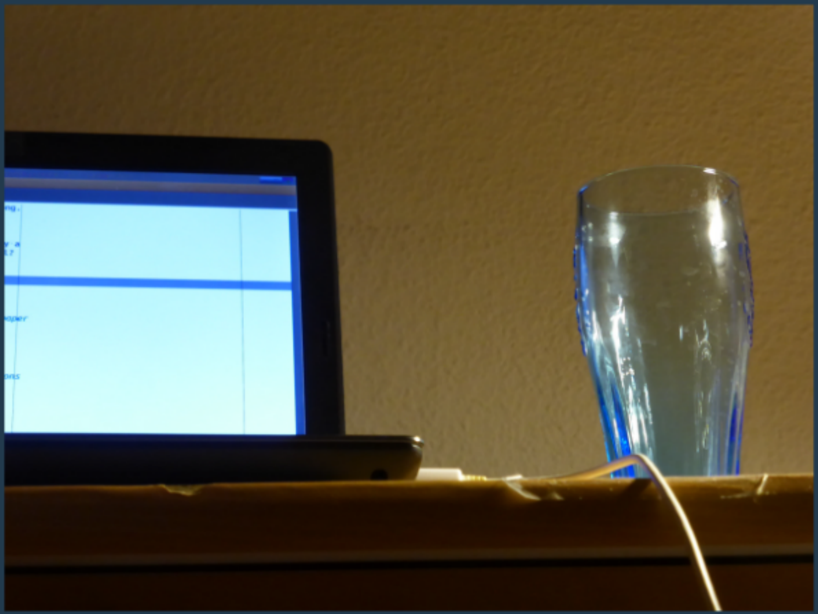


Intra-Class Variation
Deformation
Illumination
Semantic Gap
Background Clutter

Semantic Gap



Viewpoints



Thoma, Martin (2016).

Illumination

Deformation

Prezi

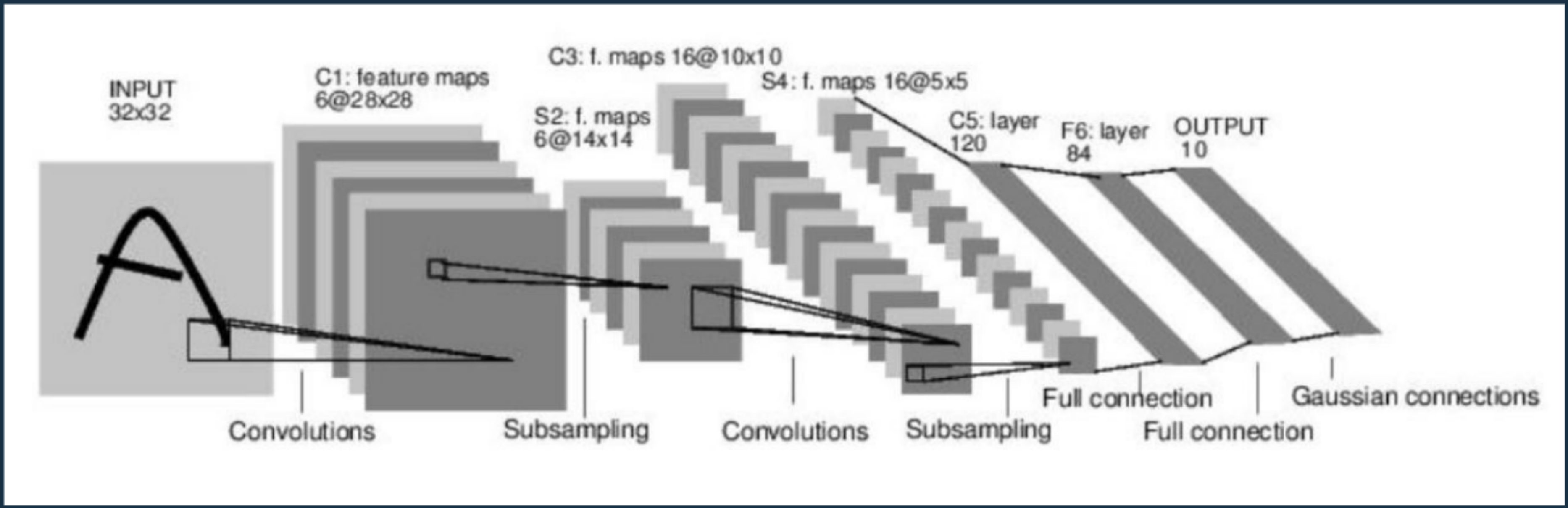
Background Clutter

Thoma, Martin. "A survey of semantic segmentation." arXiv preprint arXiv:1602.06541 (2016).

Intra-Class Variation

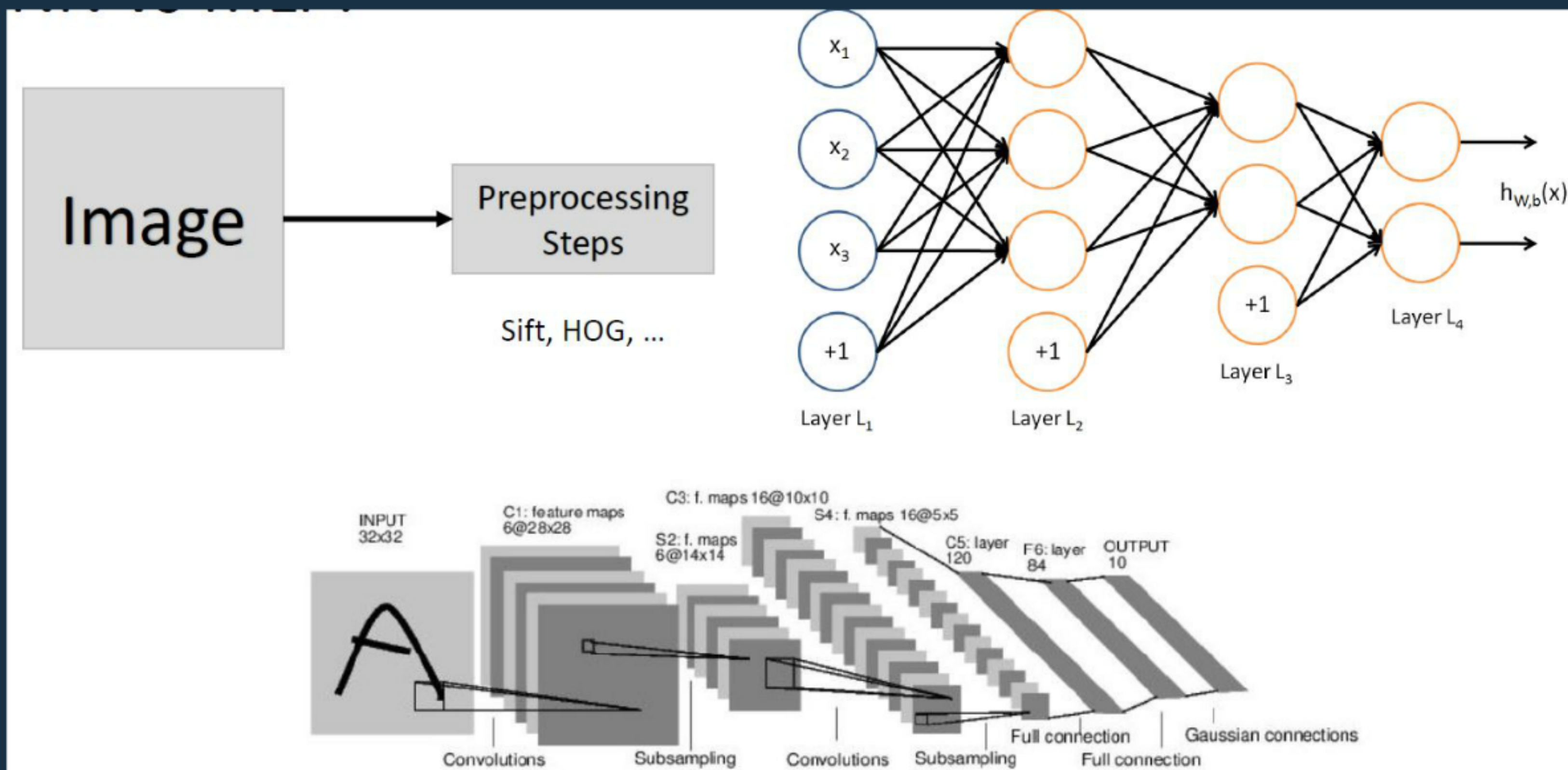
Prezi

Convolutional Neural Network:



LeNet5-LeCun 1998

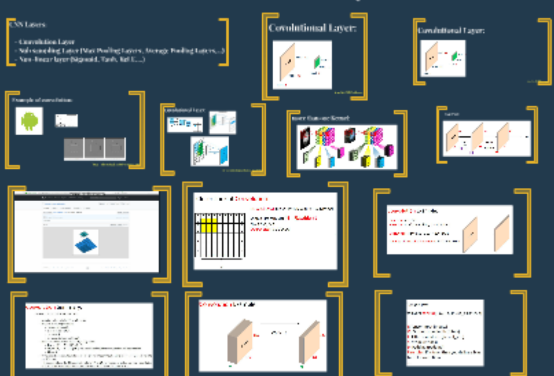
CNN vs MLP:



Prezi

CNN Structure

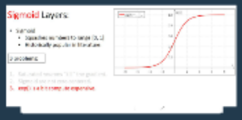
Convolution Layers

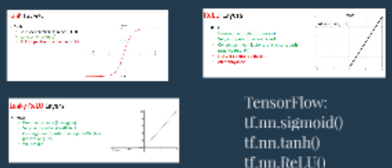


UTDLSS2017

Activation function Layer

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU

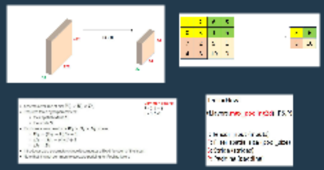




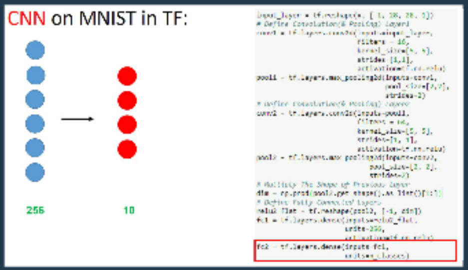
TensorFlow:
tf.nn.sigmoid()
tf.nn.tanh()
tf.nn.ReLU()

Cs231n-Lecture5,7 Stanford

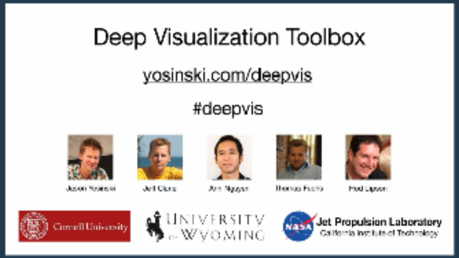
Pooling Layer



Cs231n-Lecture7 Stanford



deep visualization ToolBox

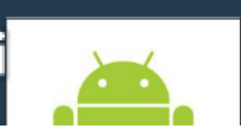


Co

CNN Layers:

- Convolution Layer
- Sub sampling Layer (Max Pooling Layers, Average Pooling Layers,...)
- Non-linear layer (Sigmoid, Tanh, ReLU,...)

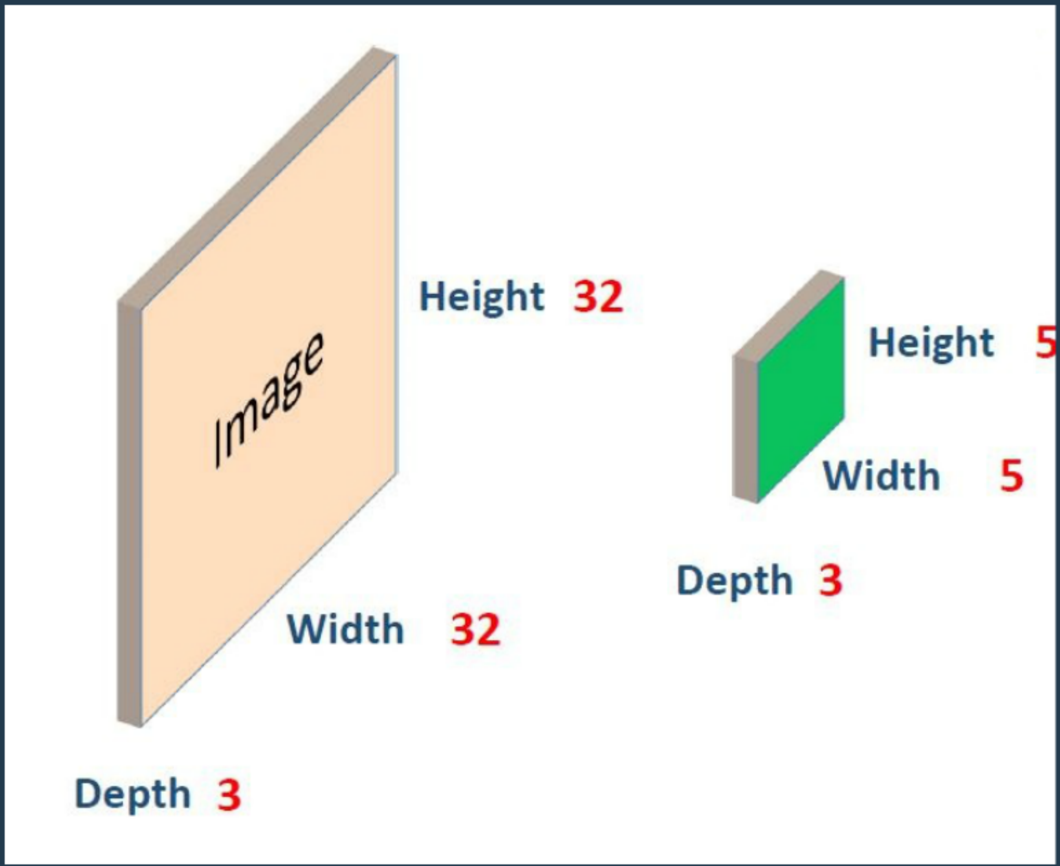
Example of convolution:



Covolutional Layer:

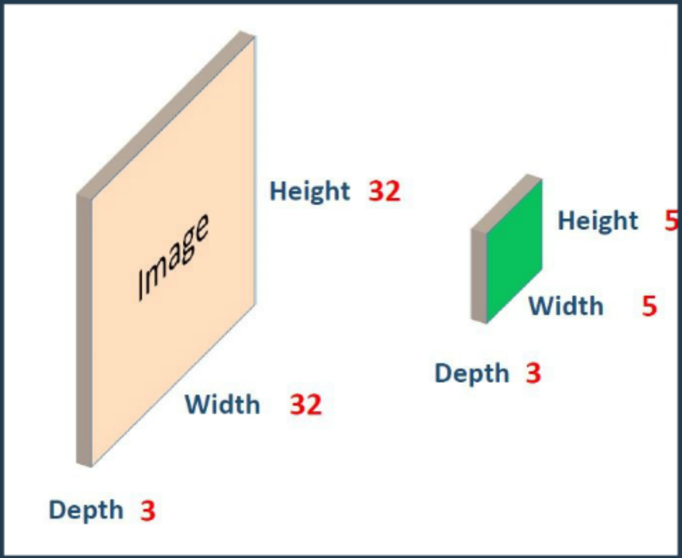
Prezi

Covolutional Layer:



Credit: UTDLSS2017

Covolutional Layer:



Credit: UTDLSS2017

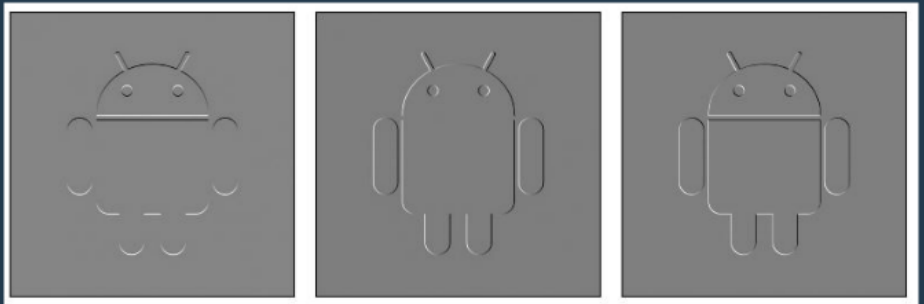
Example of convolution:

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

Finds horizontals

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

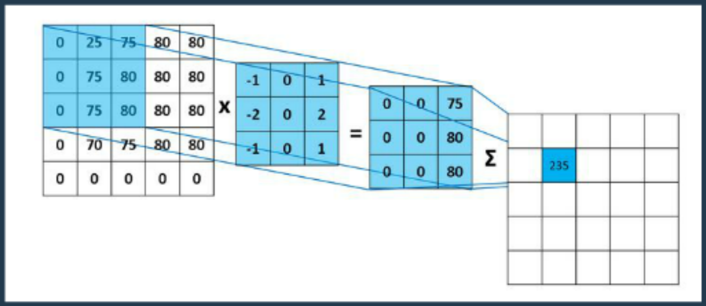
Finds verticals

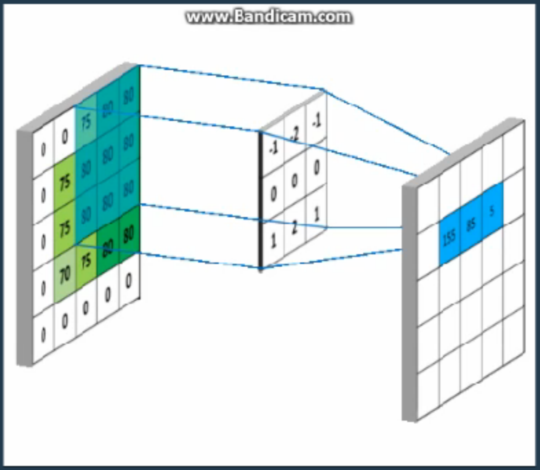


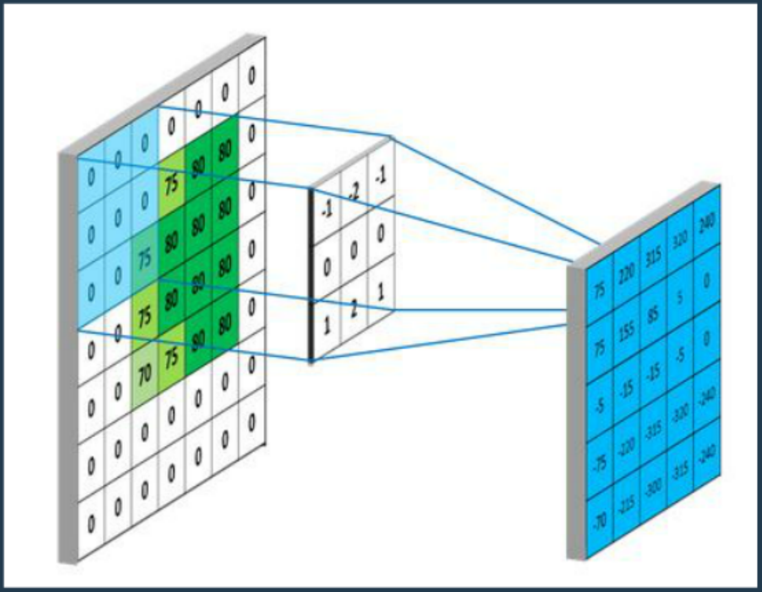
Horizontal Sobel Vertical Sobel Combined Sobel

https://mlnotebook.github.io/post/CNN1/

Covolutional Layer:



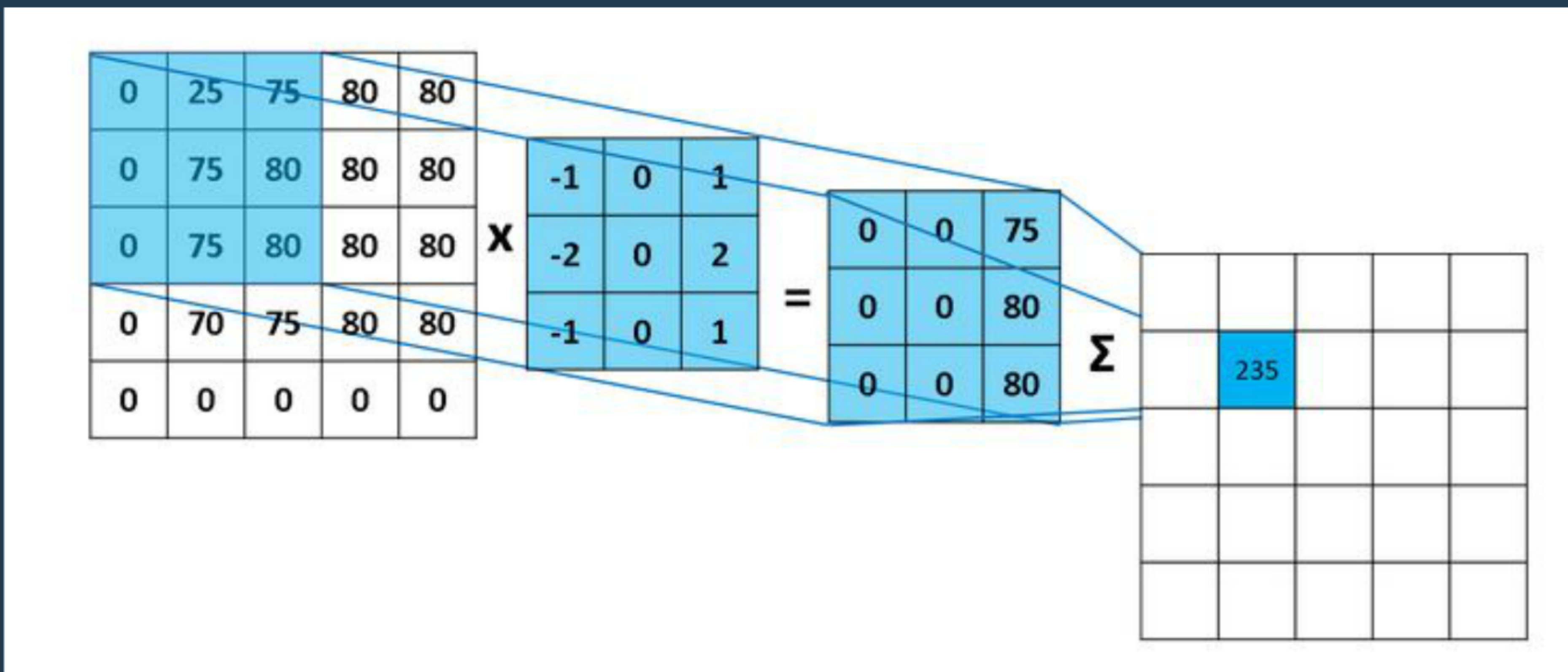


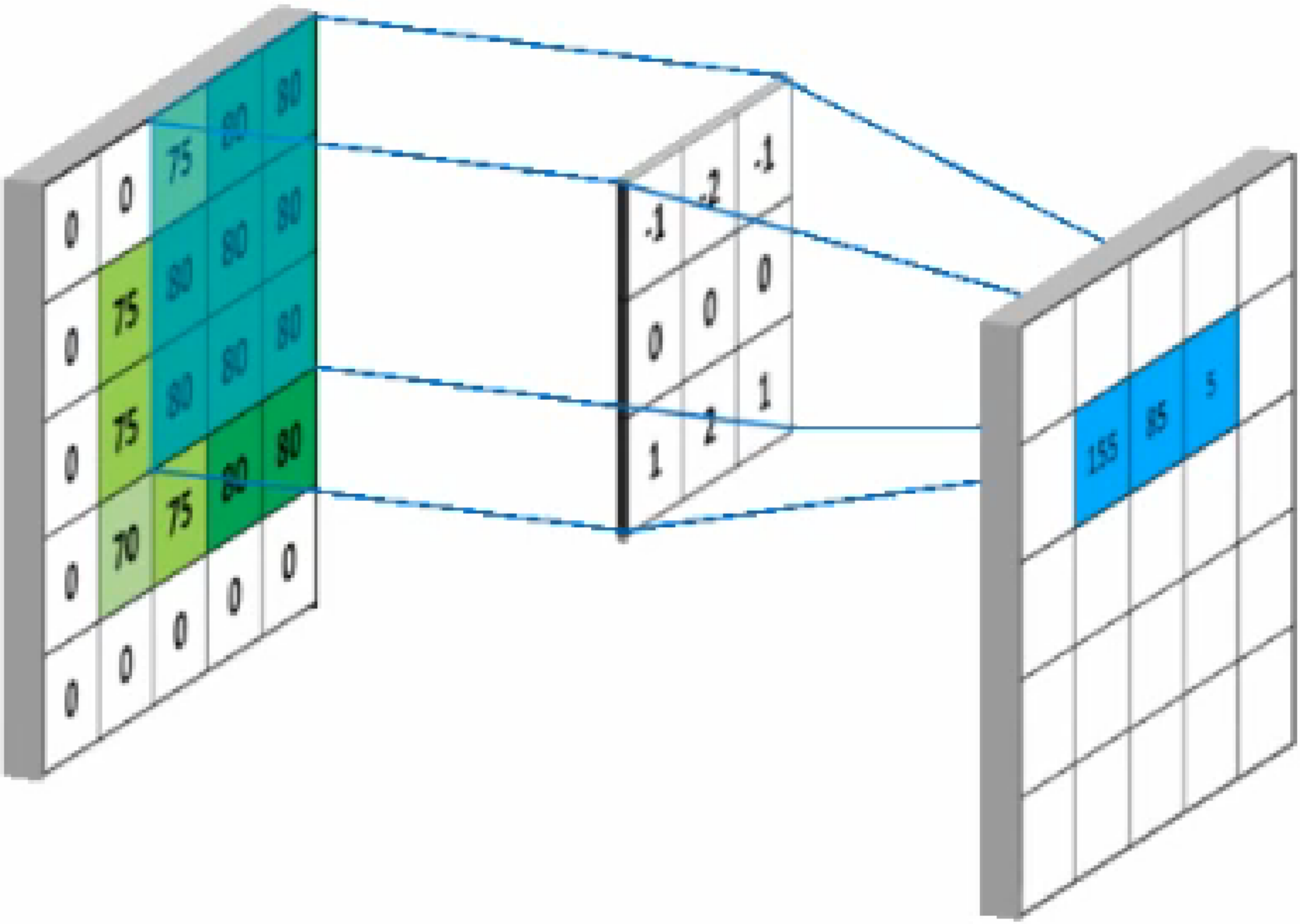


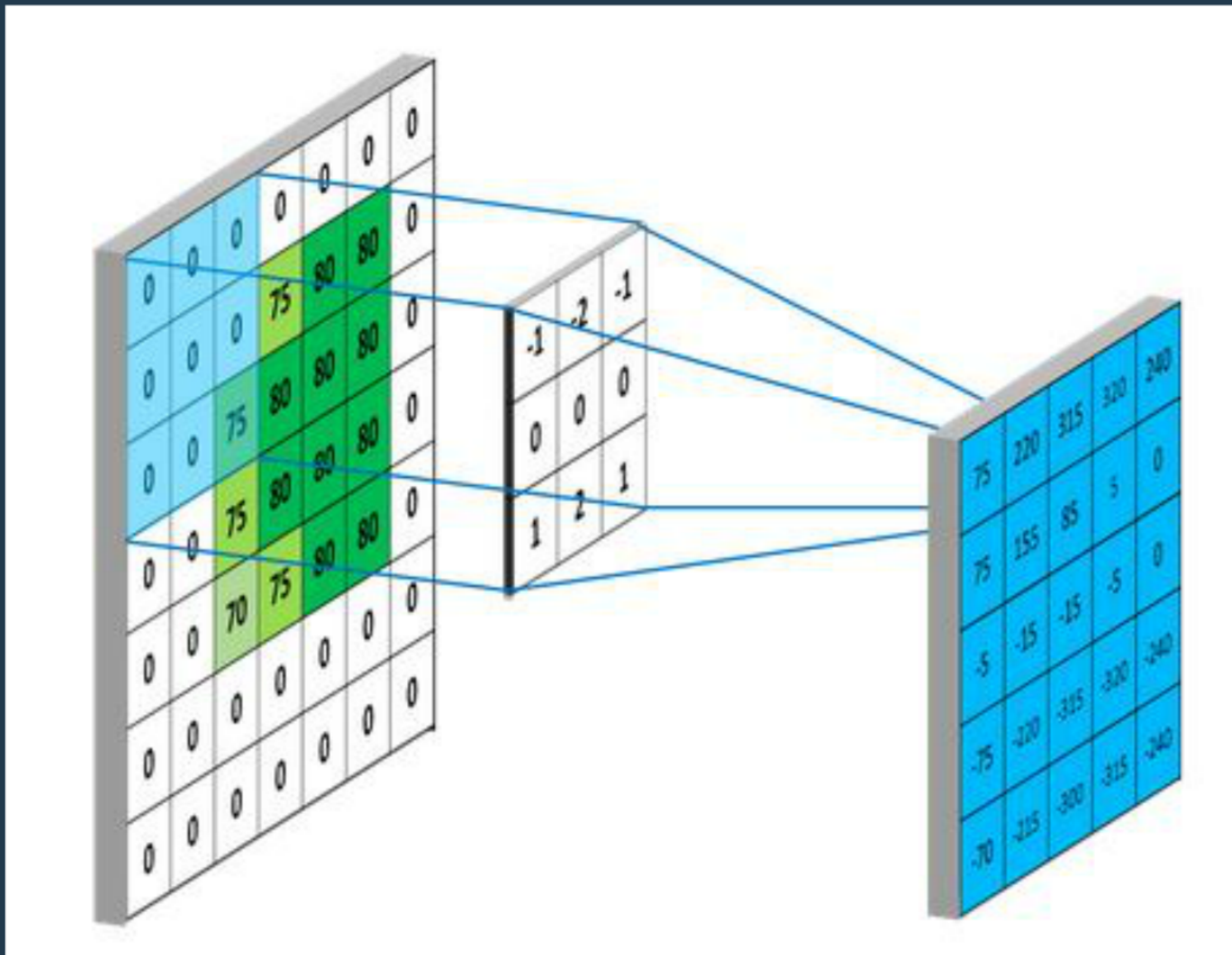
https://mlnotebook.github.io/post/CNN1/

Prezi

Covolutional Layer

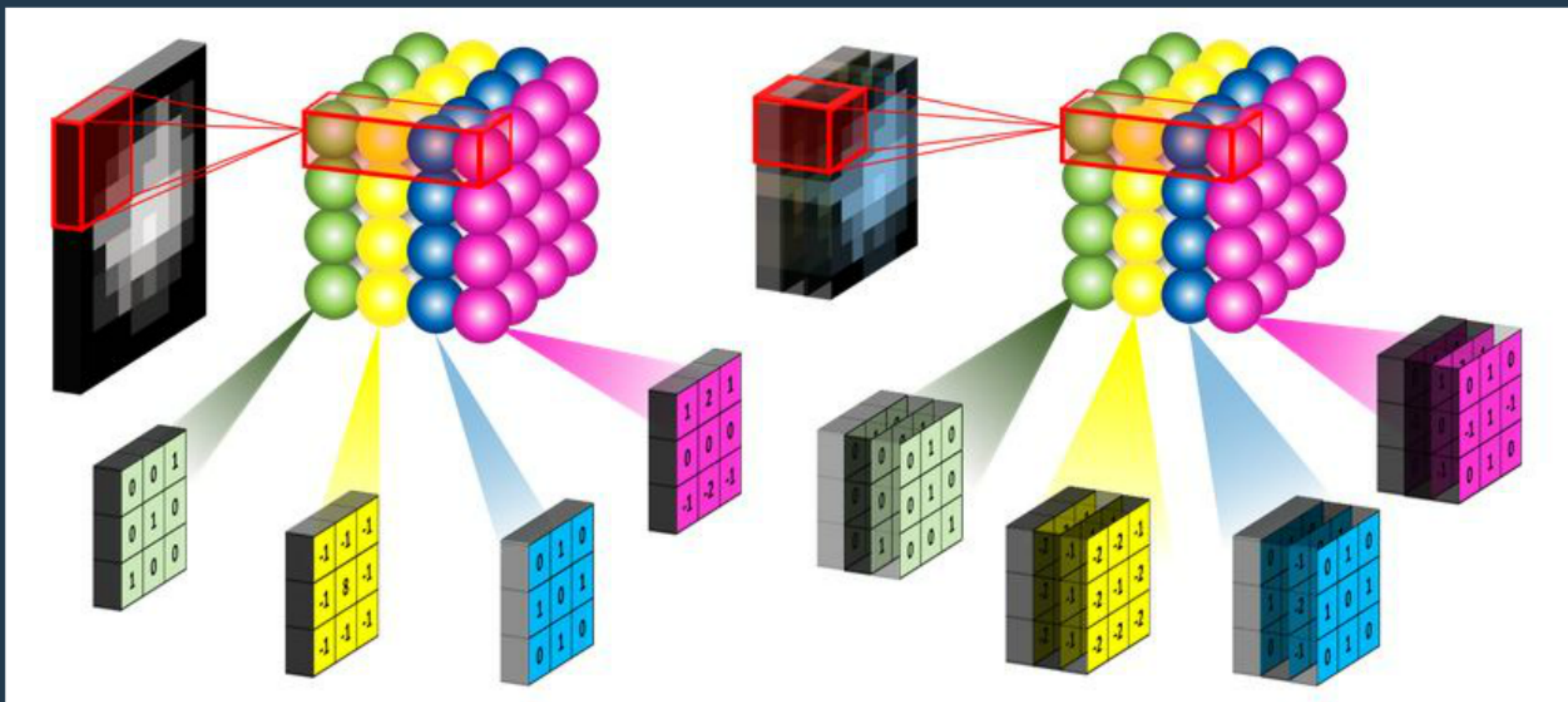



0 0 75 80 80
0 75 80 80 80
0 75 80 80 80
0 70 75 80 80
0 0 0 0 0
.1 .2 .1
0 0 0
1 2 1
155 85 5

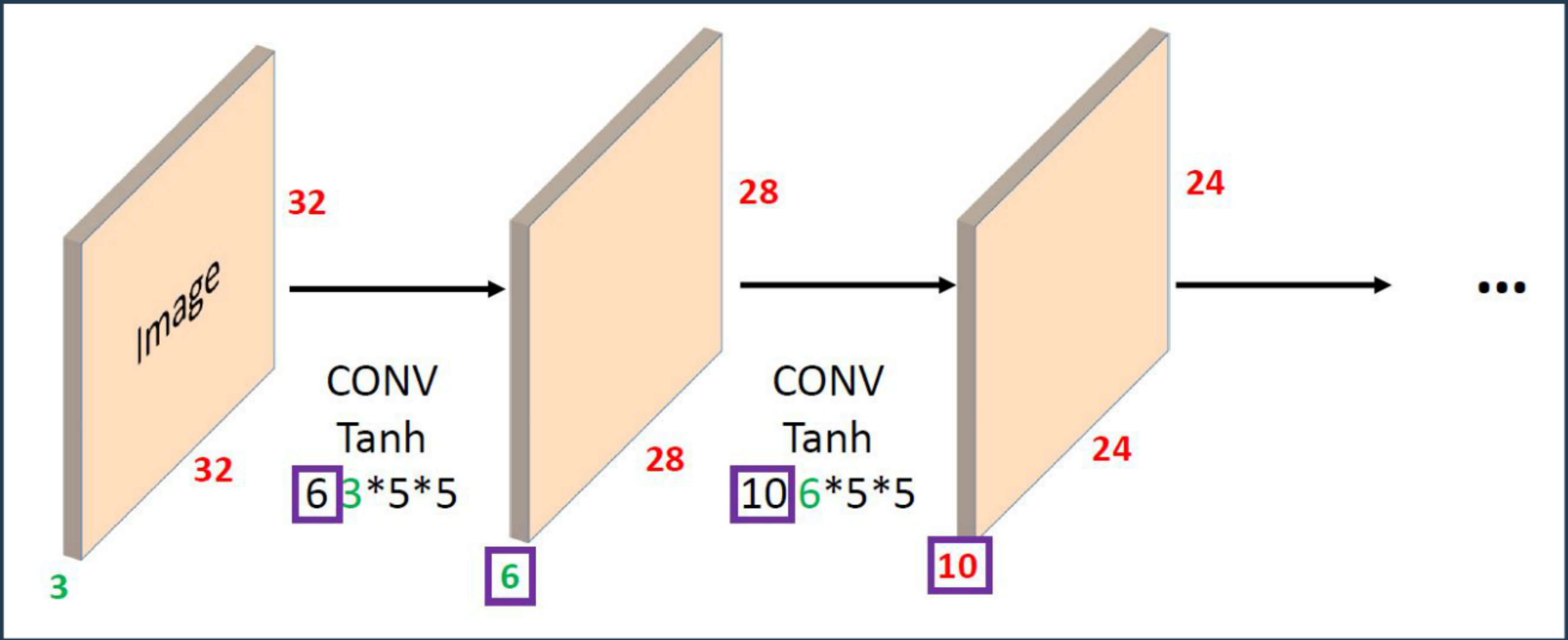


https://mlnotebool

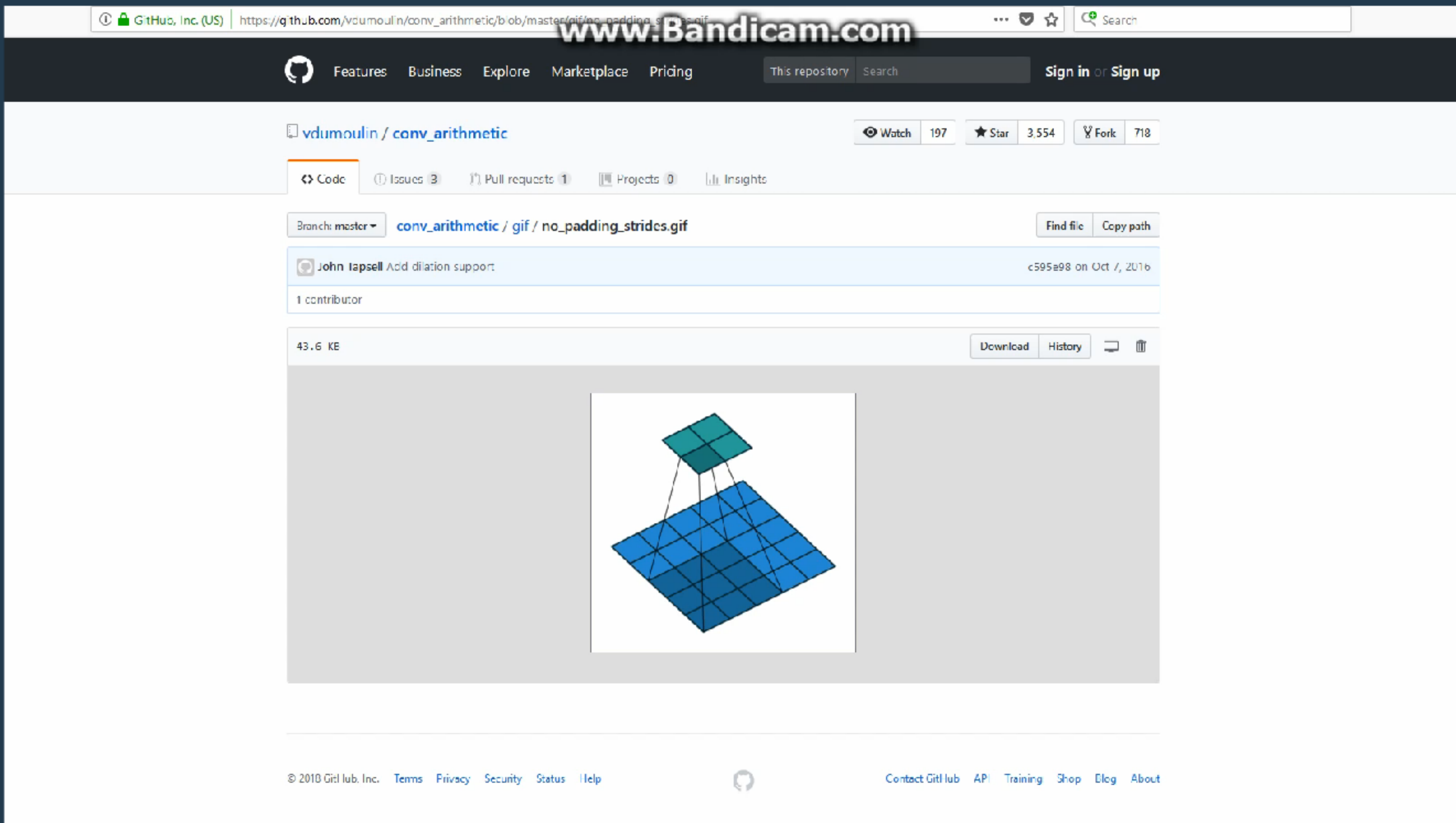
more than one Kernel:



ConvNet:



Prezi


GitHub, Inc. (US) https://github.com/vdumoulin/conv_arithmetic/blob/master/gif/no_padding_strides.gif
www.Bandicam.com
Features Business Explore Marketplace Pricing
This repository Search
Sign in or Sign up
vdumoulin / conv_arithmetic
Watch 197
Star 3,554
Fork 718
Code
Issues 3
Pull requests 1
Projects 0
Insights
Branch: master
conv_arithmetic / gif / no_padding_strides.gif
Find file
Copy path
John Tapsell Add dilation support
c595e98 on Oct 7, 2016
1 contributor
43.6 KB
Download
History
© 2018 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About

Prezi

Closer Look at Convolution:

Practical Note: In practice, It is common to zero pad.

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

N

Output Size formula: $(N - F)/stride + 1$

N = 9, S=1, F=3

Output Size: (9-3)/1+1=7

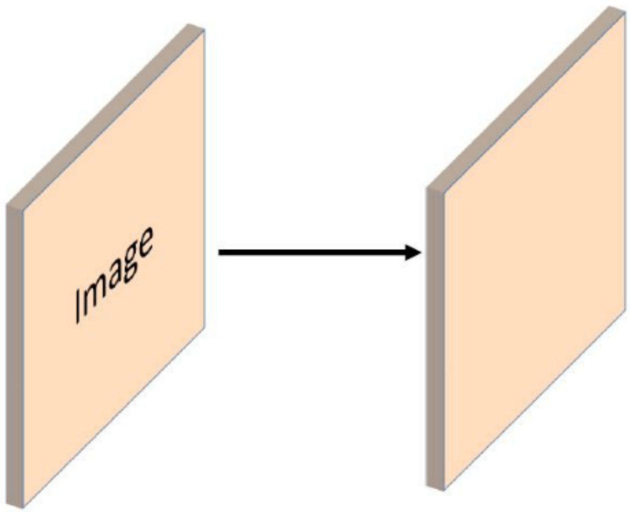
Convolution Example:

Input size: 3*32*32

Filter size: 20 3*5*5, Padding=2 and Stride 1

Output size: (32+4-5)/1+1=32 -> 20*32*32

Number of Learnable Parameters: 20*3*5*5=1500



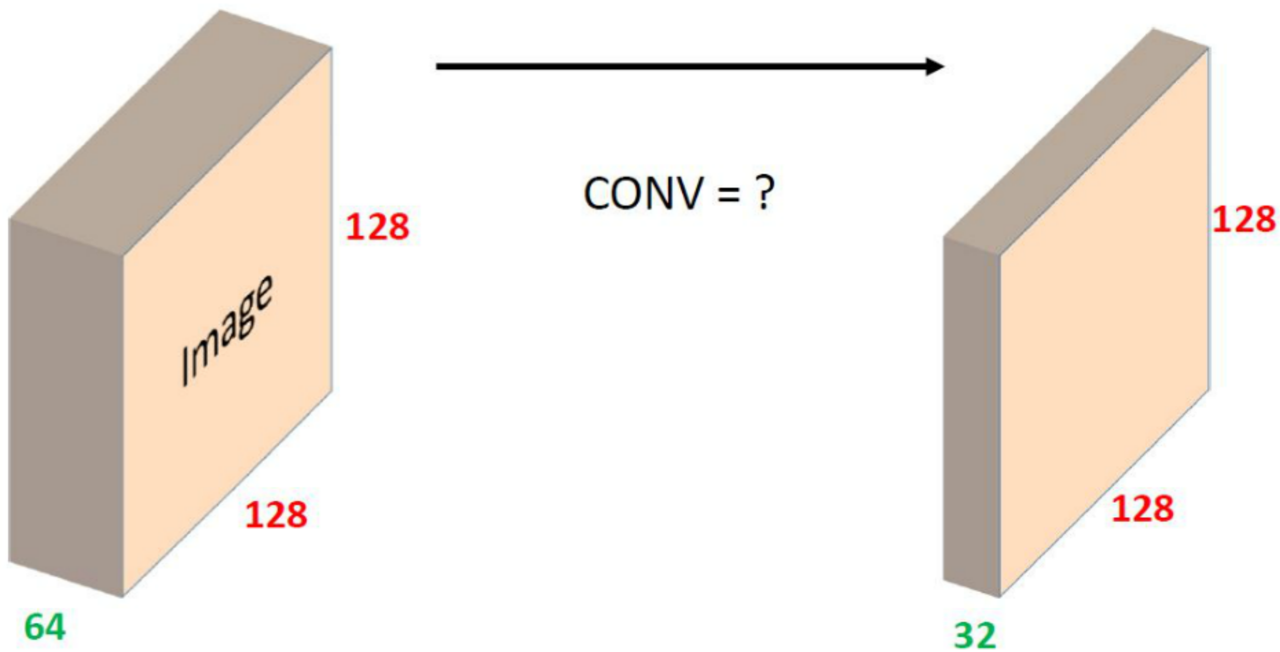
Prezi

Convolution Summary:

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Convolution Example:

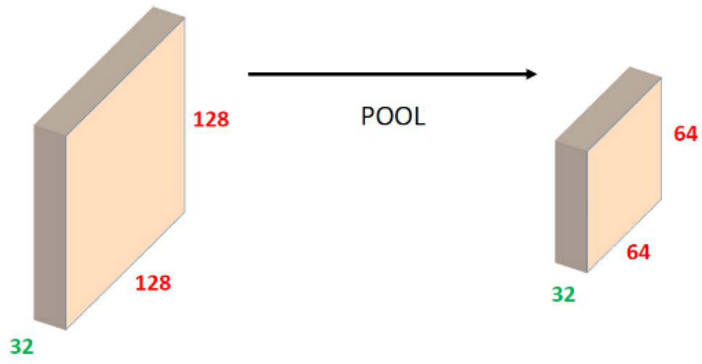


TensorFlow:

tf.layers.conv2d(I, O, F, S, P, use_bias=True)

I: Tensor Input (inputs)
O: Output cube depth (filters)
F: Filter spatial size (Kernel_size)
S: Stride (strides)
P: Padding (padding)
Use_bias: If is true, then you define a bias term for each filter
...

Pooling Layer



1	2	6	5
0	3	1	5
7	2	9	1
4	8	10	0

→

3	6
8	10

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common Setting:
F = 2, S = 2
F = 3, S=2

TensorFlow:

tf.layers.max_pooling2d(I,F,S,P)

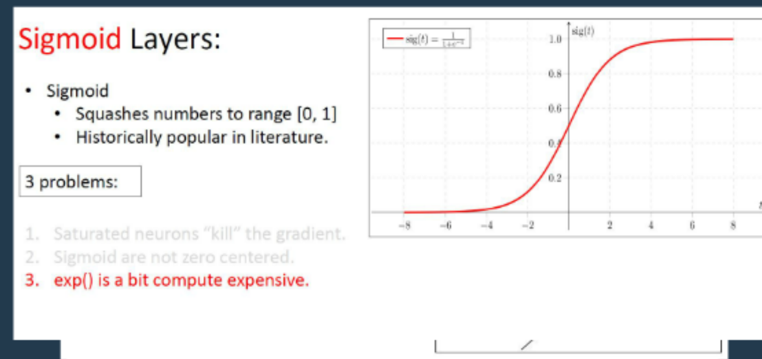
I: Tensor Input (inputs)
F: Filter spatial size (pool_size)
S: Stride (strides)
P: Padding (padding)

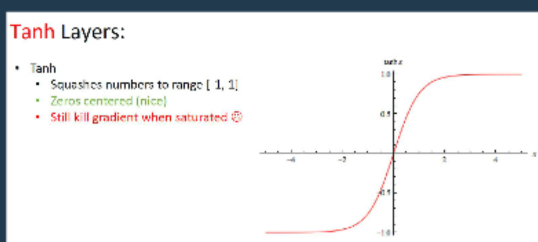
Cs231n-Lecture7 Stanford

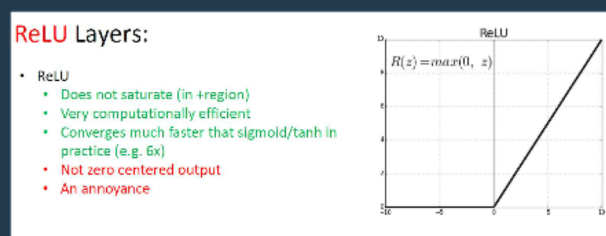
Prezi

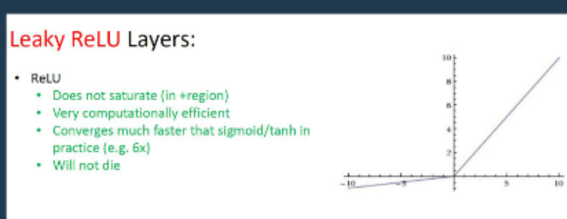
Activation function Layer

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU









TensorFlow:
tf.nn.sigmoid()
tf.nn.tanh()
tf.nn.ReLU()

Cs231n-Lecture5,7 Stanford

Prezi

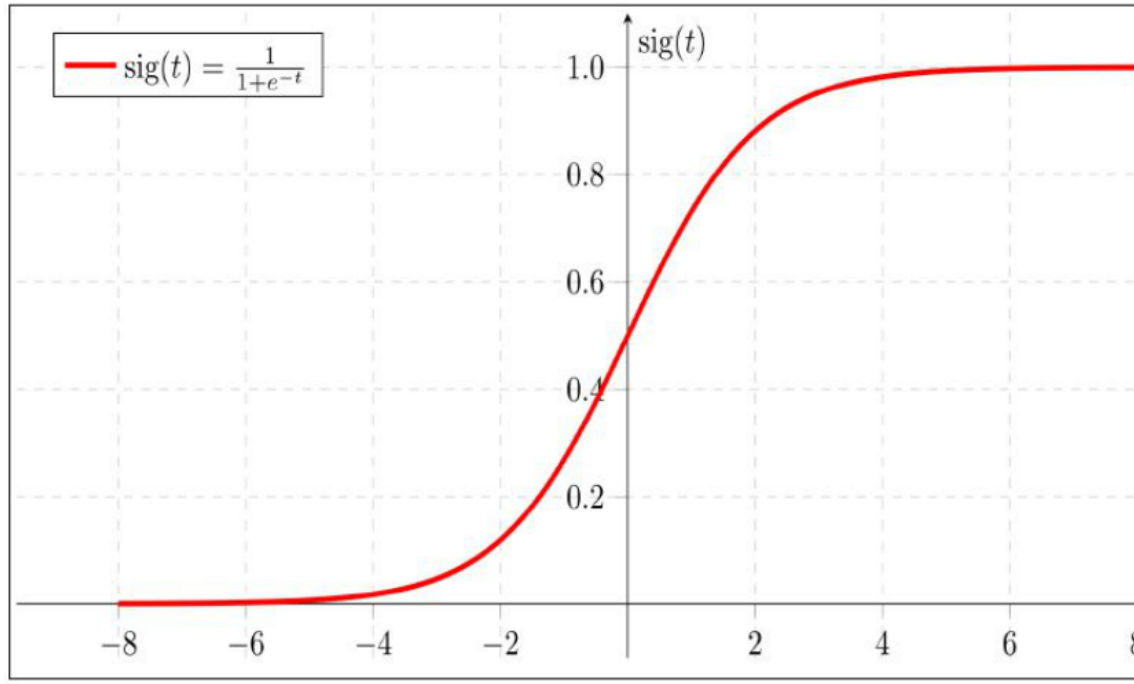
- Sigmoid
- Tanh
- ReLU
- Leaky ReLU

gmoid Layers:

Sigmoid
- Squashes numbers to range [0, 1]
- Historically popular in literature.

roblems:

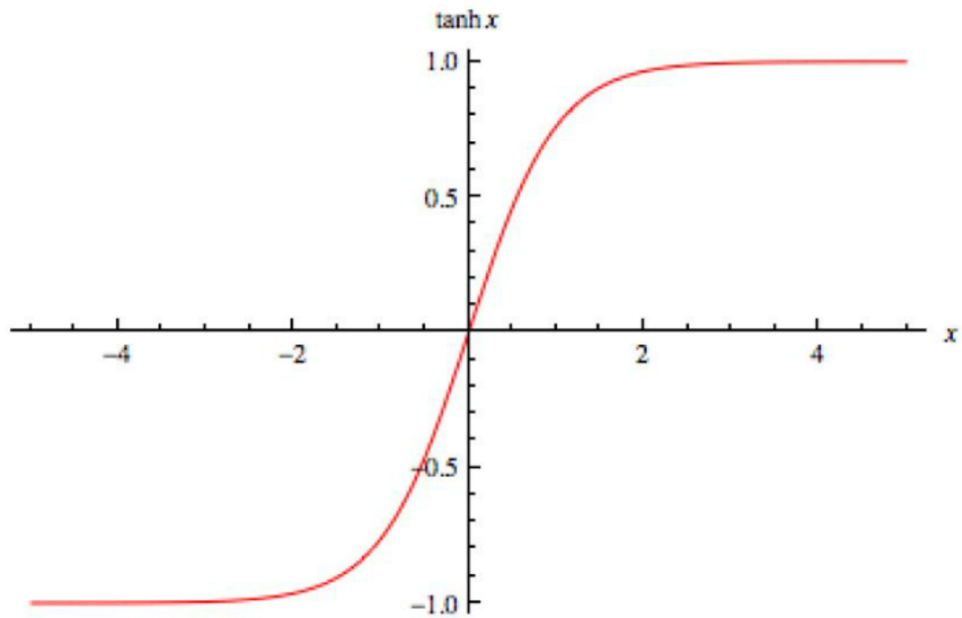
Saturated neurons “kill” the gradient.
Sigmoid are not zero centered.
exp() is a bit compute expensive.



Leaky ReLU

Tanh Layers:

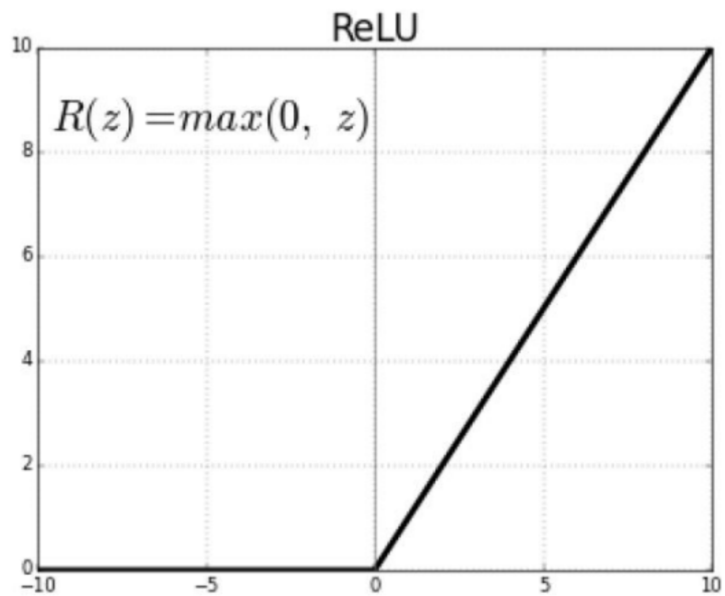
- Tanh
  - Squashes numbers to range [-1, 1]
  - Zeros centered (nice)
  - Still kill gradient when saturated ☹



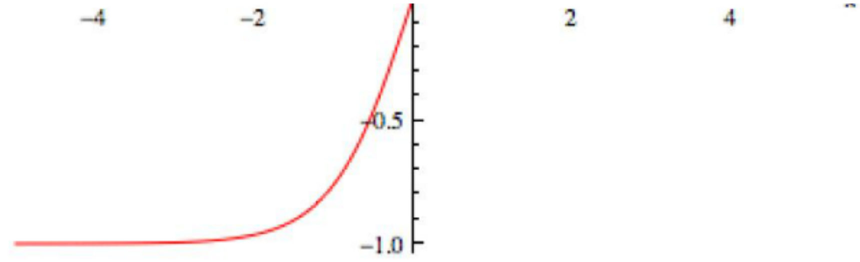
Leaky ReLU Layers:

ReLU Layers:

- ReLU
  - Does not saturate (in +region)
  - Very computationally efficient
  - Converges much faster that sigmoid/tanh in practice (e.g. 6x)
  - Not zero centered output
  - An annoyance

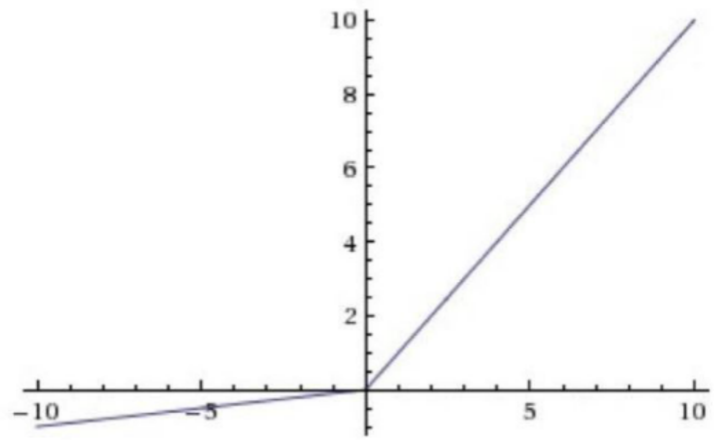


TensorFlow:



Leaky ReLU Layers:

- ReLU
  - Does not saturate (in +region)
  - Very computationally efficient
  - Converges much faster that sigmoid/tanh in practice (e.g. 6x)
  - Will not die



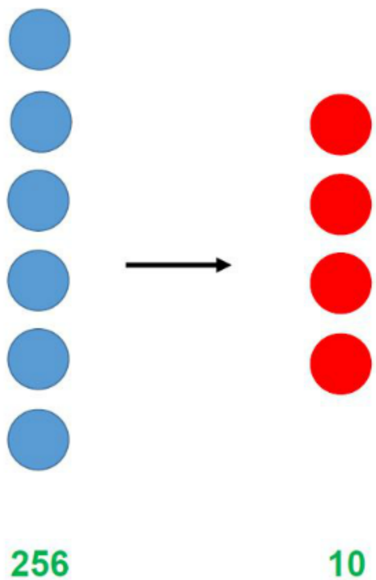
TensorFlow:

tf.nn.sigmoid()

tf.nn.tanh()

tf.nn.ReLU()

CNN on MNIST in TF:




```
input_layer = tf.reshape(x, [-1, 28, 28, 1])
# Define Convolution(& Pooling) Layer1
conv1 = tf.layers.conv2d(inputs=input_layer,
                         filters = 16,
                         kernel_size=[5, 5],
                         strides=[1,1],
                         activation=tf.nn.relu)
pool1 = tf.layers.max_pooling2d(inputs=conv1,
                                pool_size=[2,2],
                                strides=2)
# Define Convolution(& Pooling) Layer2
conv2 = tf.layers.conv2d(inputs=pool1,
                         filters = 64,
                         kernel_size=[5, 5],
                         strides=[1, 1],
                         activation=tf.nn.relu)
pool2 = tf.layers.max_pooling2d(inputs=conv2,
                                pool_size=[2, 2],
                                strides=2)
# Multiply The Shape of Previous Layer
dim = np.prod(pool2.get_shape().as_list()[1:])
# Define Fully Connected Layers
relu2_flat = tf.reshape(pool2, [-1, dim])
fc1 = tf.layers.dense(inputs=relu2_flat,
                      units=256,
                      activation=tf.nn.relu)
fc2 = tf.layers.dense(inputs=fc1,
                      units=n_classes)
```

Prezi

deep visualization ToolBox

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

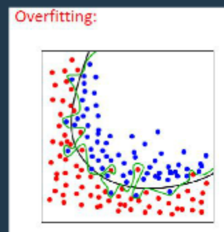
Cornell University

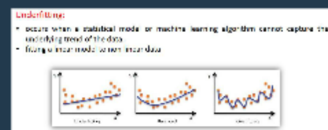
University of Wyoming

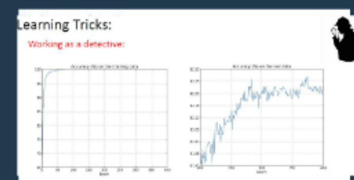
Jet Propulsion Laboratory
California Institute of Technology

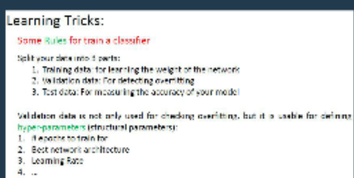
Advance info about Learning

- Learning Tricks
- Dropout
- Weight Initialization
- Batch Normalization



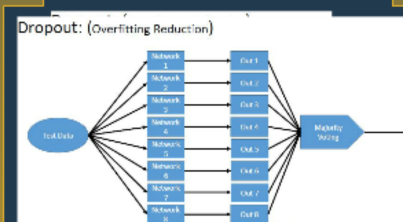




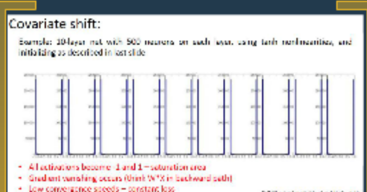


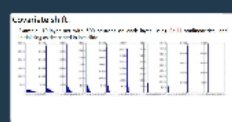
Regularization Technique:

- L2 norm
- L1 norm
- Dropout!



a simple way to prevent neural networks from over fitting (nitish srivasta, 2014)

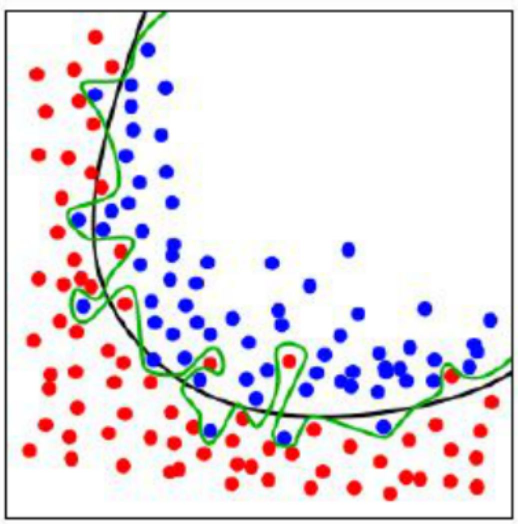




Prezi

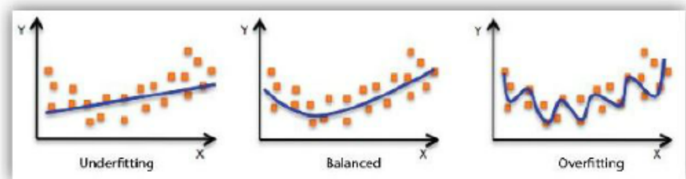
Advance info about Lea

Overfitting:



Underfitting:

- occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data.
- fitting a linear model to non-linear data



Regularization Technique:

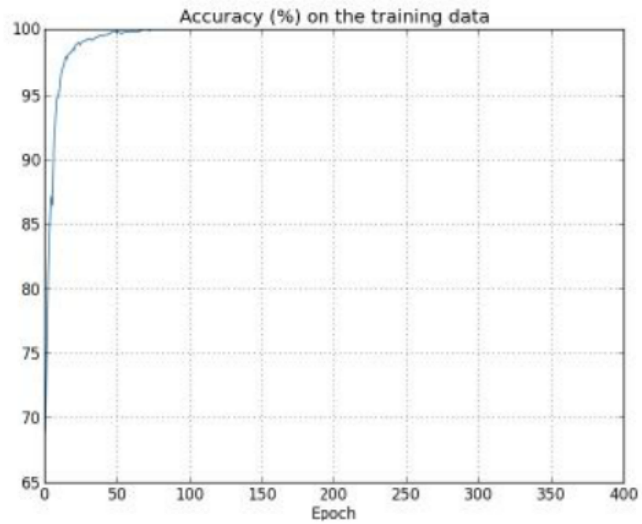
- L2 norm
- L1 norm
- Dropout!

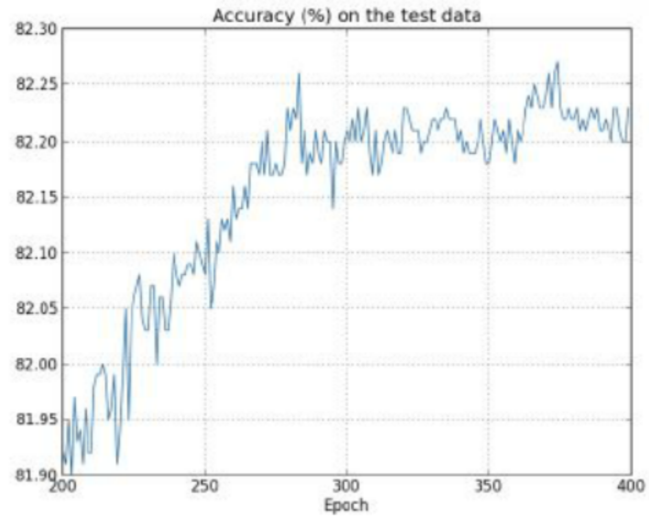
Prezi

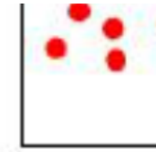
ing

Learning Tricks:

Working as a detective:







Learning Tricks:

Some Rules for train a classifier

Split your data into 3 parts:

1. Training data: for learning the weight of the network
2. Validation data: For detecting overfitting
3. Test data: For measuring the accuracy of your model

Validation data is not only used for checking overfitting, but it is usable for defining hyper-parameters (structural parameters):

1. # epochs to train for
2. Best network architecture
3. Learning Rate
4. ...

Covariate shift:

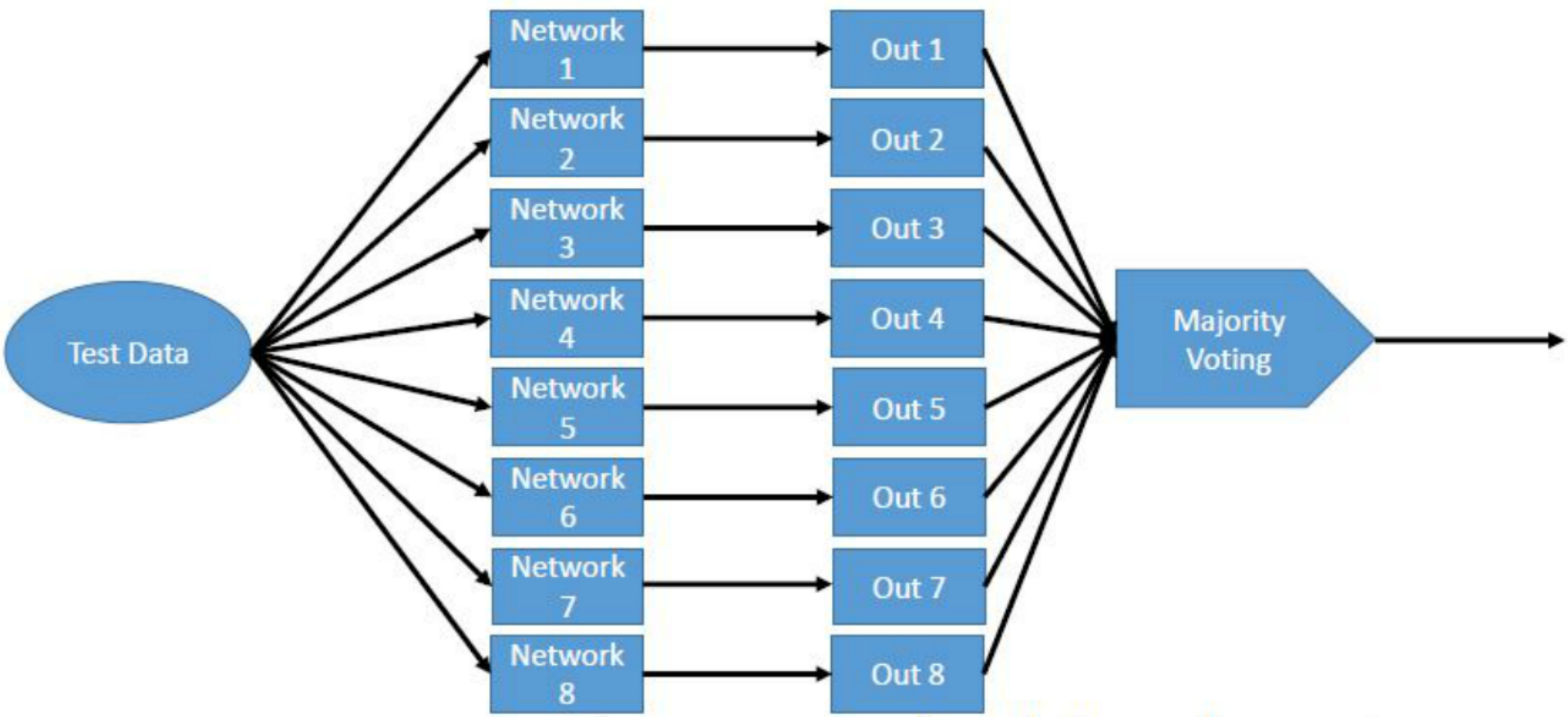
Example: 10-layer net with 500 neurons on each layer, using tanh nonlinearities, and initializing as described in last slide

Regularization Technique:

- L2 norm
- L1 norm
- Dropout!

Batch Normalization:

Dropout: (Overfitting Reduction)

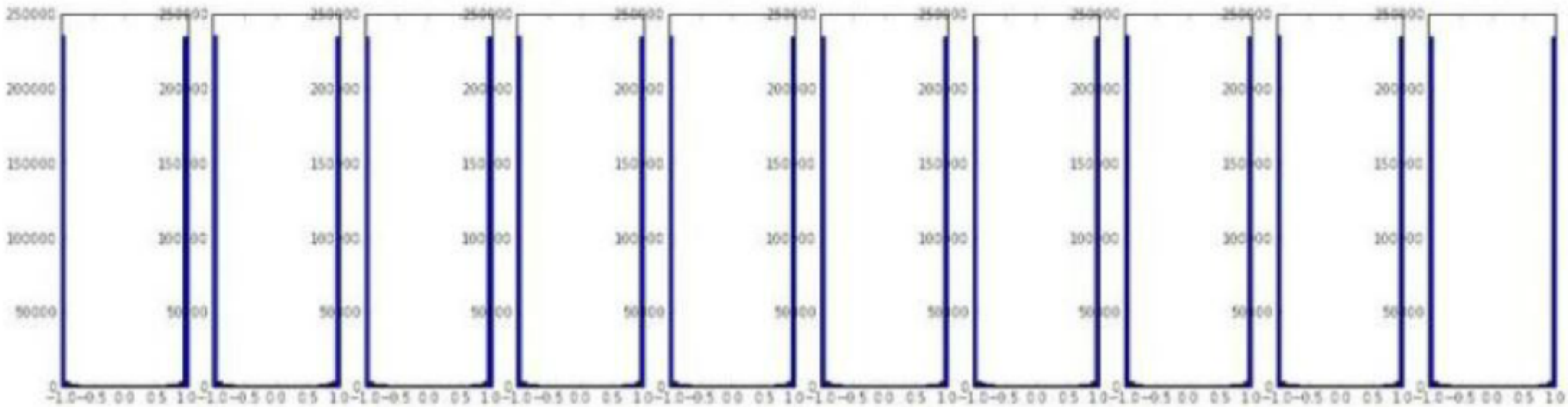


a simple way to prevent neural networks from over fitting (nitish sirivasta, 2014)

Prezi

Covariate shift:

Example: 10-layer net with 500 neurons on each layer, using tanh nonlinearities, and initializing as described in last slide



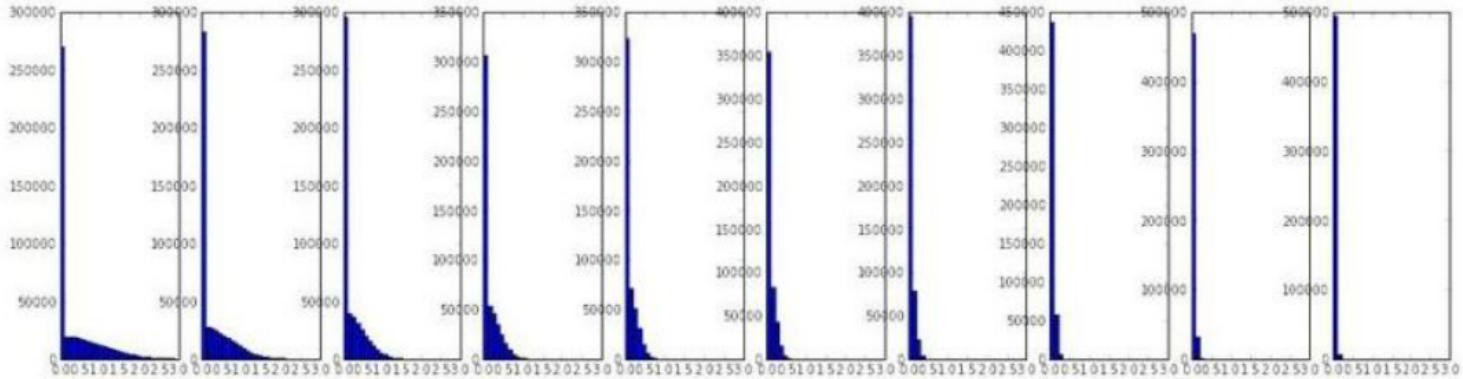
- All activations become -1 and 1 – saturation area
- Gradient vanishing occurs (think W*X in backward path)
- Low convergence speeds – constant loss

Cs231n-Lecture5-Stanford University

Prezi

Covariate shift:

Example: 10-layer net with 500 neurons on each layer, using ReLU nonlinearities, and initializing as described in last slide



Cs231n-Lecture5-Stanford University

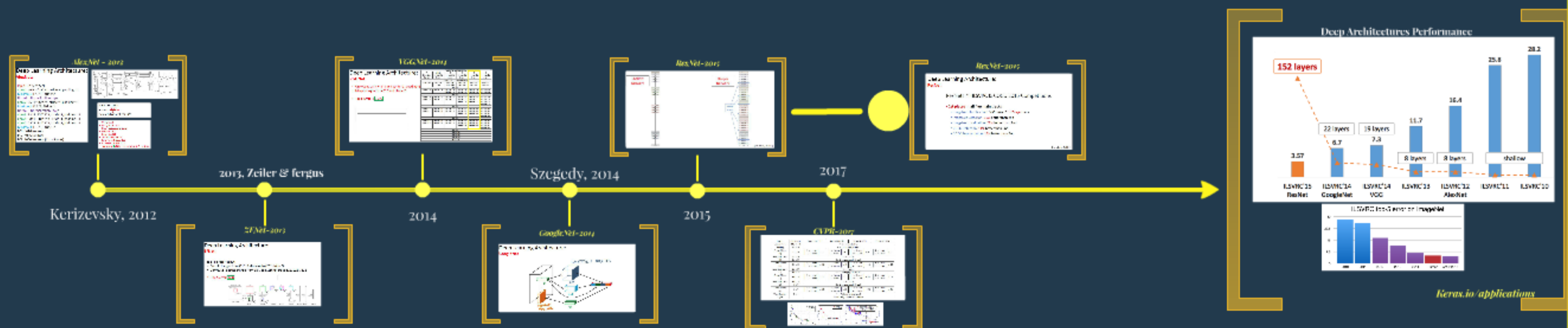
Prezi

Batch Normalization:

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe, Christian Szegedy - 2015

Architecture of CNN

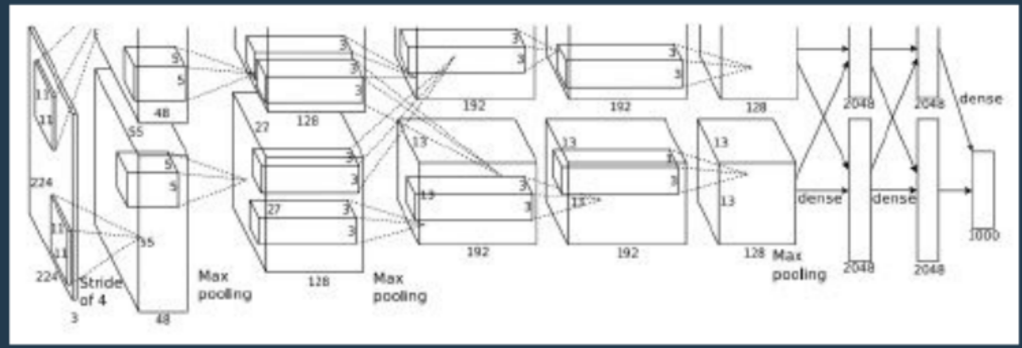


AlexNet - 2012

Deep Learning Architecture:

AlexNet:

Input: [3*227*227]
Conv1: 96 F=3*11*11, stride=4, padding =0
MaxPool1: F=3*3, Stride=2
Norm1 = Normalization Layer
Conv2: 256 F=96*5*5, stride=1, padding =2
MaxPool2: F=3*3, Stride=2
Norm2 = Normalization Layer
Conv3: 384 F=256*3*3, stride=1, padding = 1
Conv4: 384 F=384*3*3, stride=1, padding = 1
Conv5: 256 F=384*3*3, stride=1, padding = 1
MaxPool3: F=3*3, Stride=2
FC6: 4096 Neurons
FC7: 4096 Neurons
FC8: 1000 Neurons (Class Scores)



Input: 256*13*13
Formula: $\frac{(N-F)}{S} + 1$
Output (MaxPool3): 256*6*6

- Using ReLU
- Using Norm Layer
- Heavy data augmentation
- Dropout = 0.5
- Batch size 128
- SGD momentum 0.9
- Lr=0.01, with decay 0.1
- L2 weight decay 0.0005
- Top 5 error: $\boxed{18.2\%}$ -> ensemble 7 CNN (15.04%)

ZFNet-2013

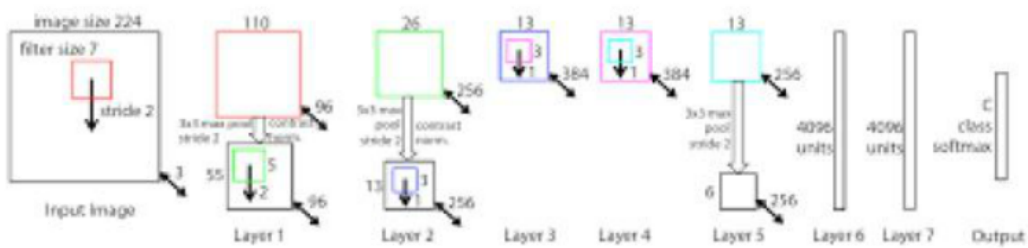
Deep Learning Architecture:

ZFNet:

It is like AlexNet but:

- Conv1 changes from (11*11 stride 4) to (7*7 stride 2)
- Conv3,4,5: depths change from 384, 384, 256 filters to 512, 1024, 512

- Top 5 error: 14.8



Prezi

VGGNet-2014

Deep Learning Architecture:

VGGNet

- Only Conv with F=3*3 and stride=1, padding=1
- Maxpooling with F=2*2 and Stride 2

- Top 5 Error is: $\boxed{7.3\%}$

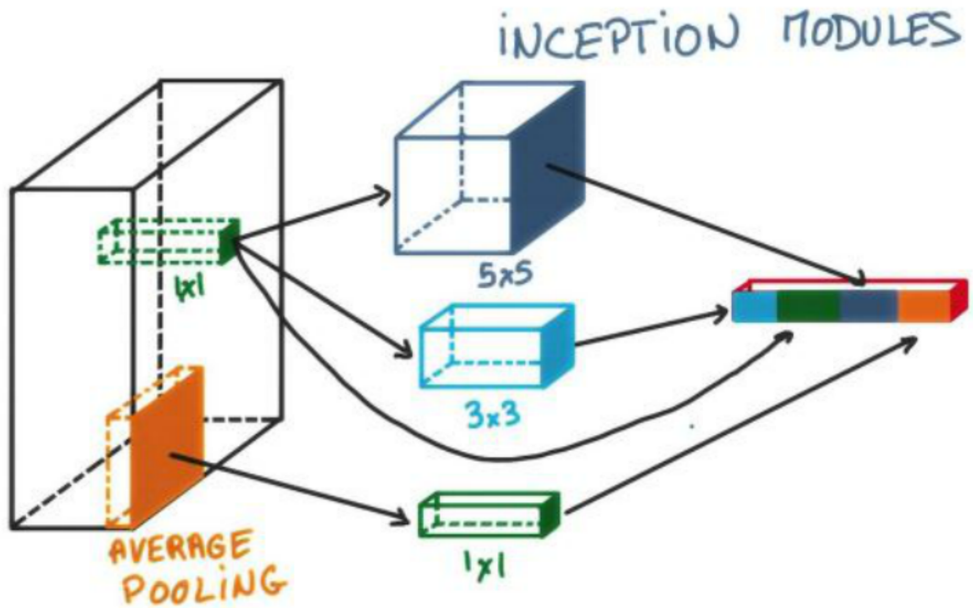
...

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

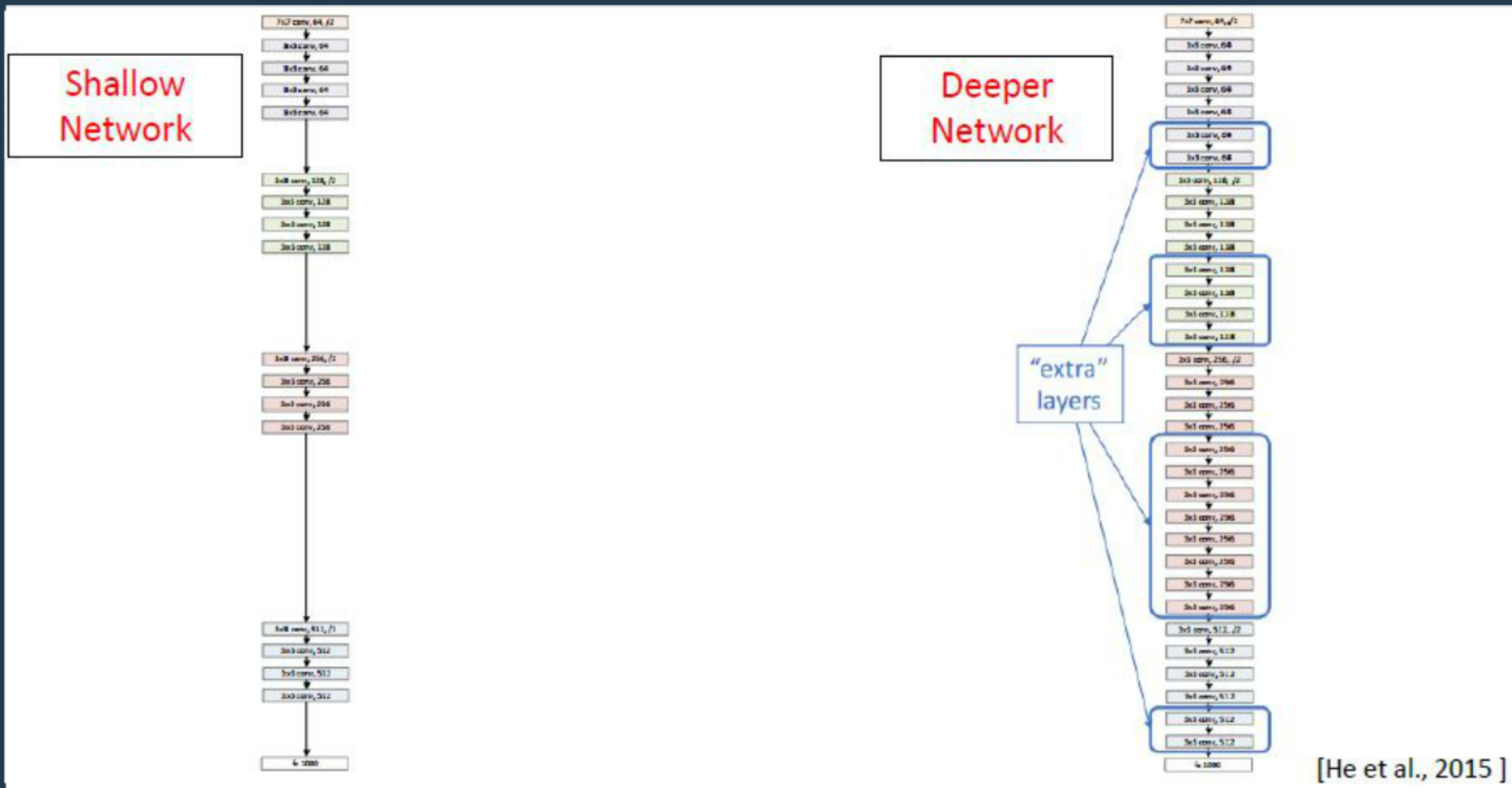
GoogleNet-2014

Deep Learning Architecture:

GoogleNet



ResNet-2015



Prezi

ResNet-2015

Deep Learning Architecture:

ResNet:

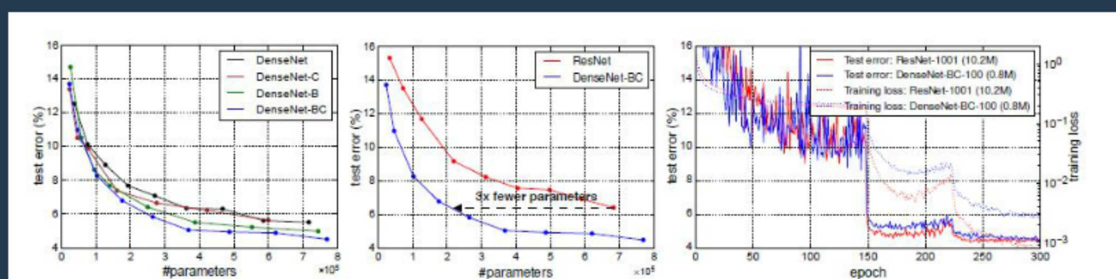
ResNets @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

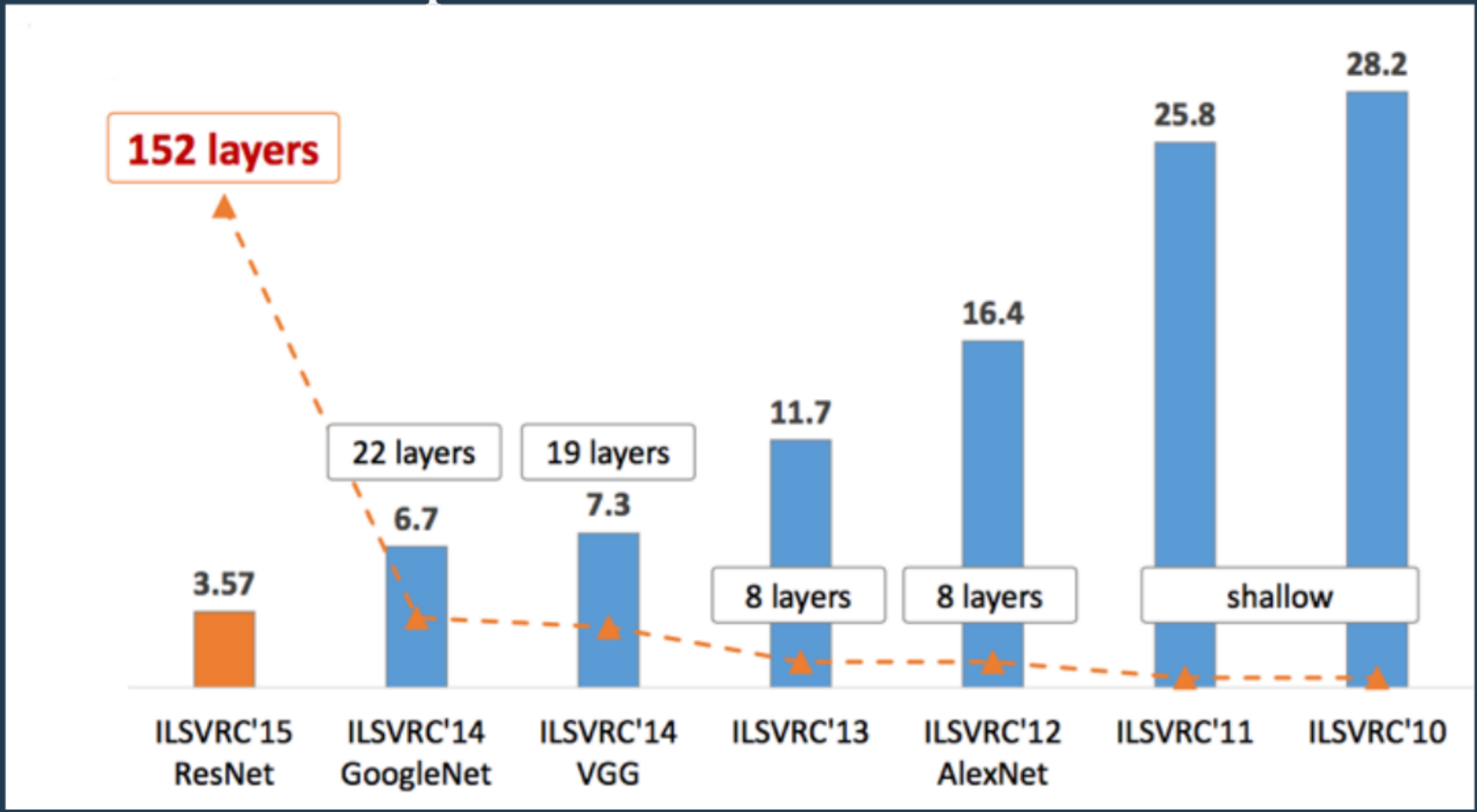
[He et al., 2015]

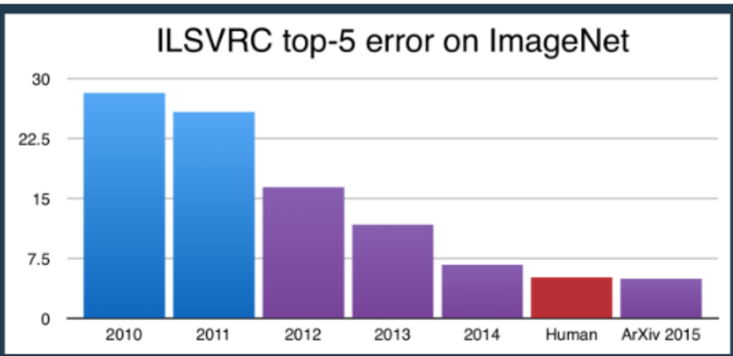
CVPR-2017

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			



Deep Architectures Performance

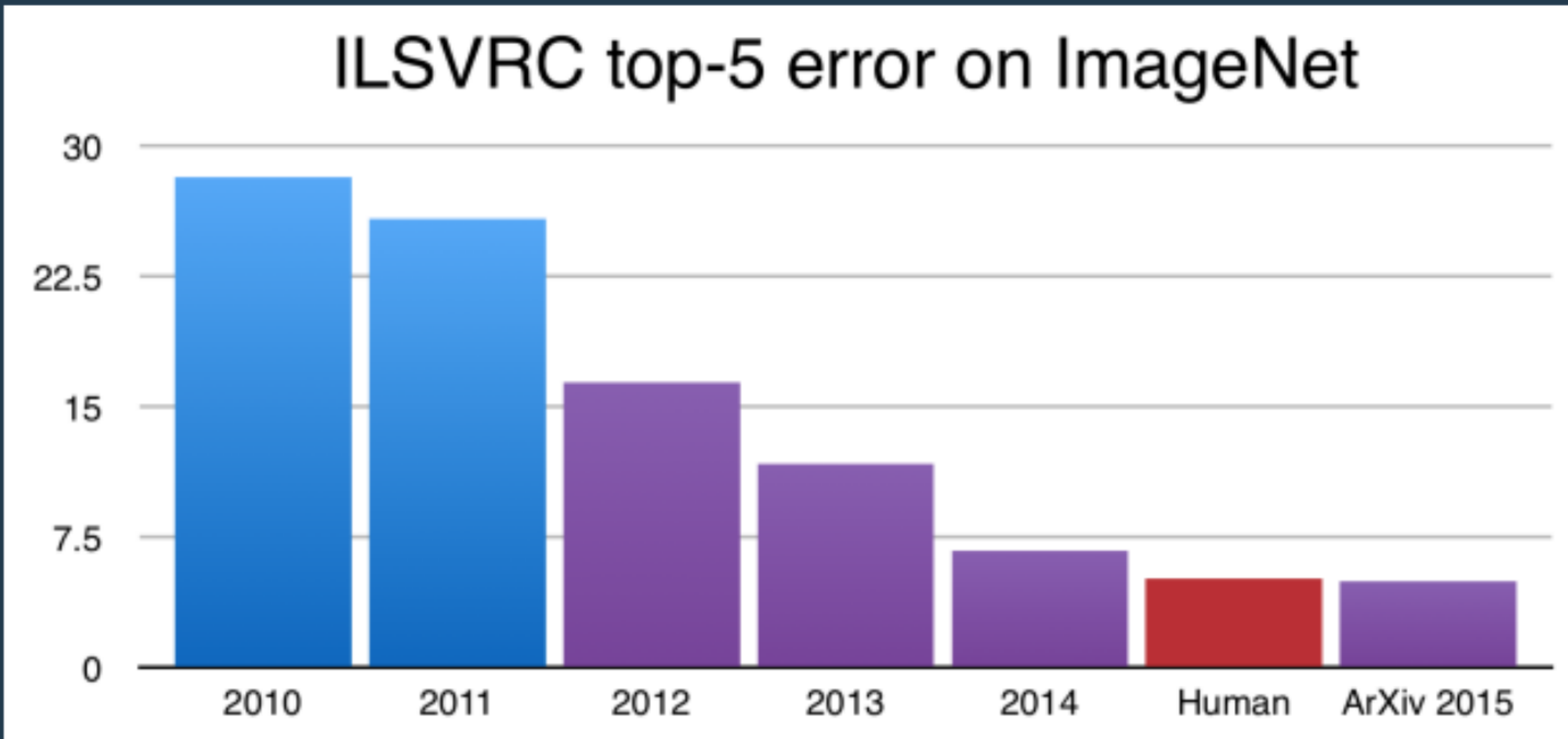




Keras.io/applications

Prezi

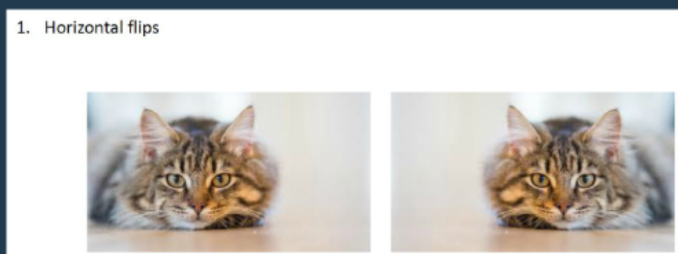
ILSVRC'14	ILSVRC'14	ILSVRC'13	ILSVRC'1
GoogleNet	VGG		AlexNet

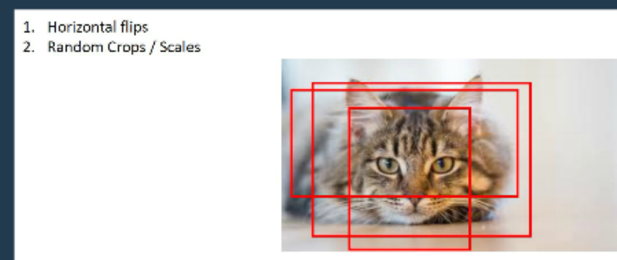


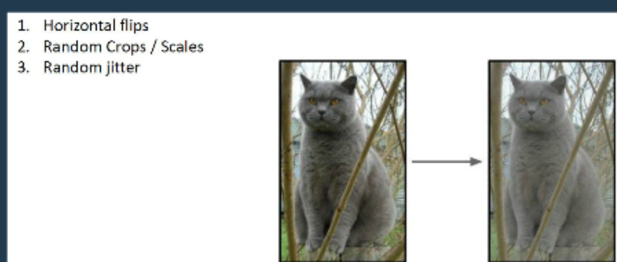
t-SNE visualization of CNN codes

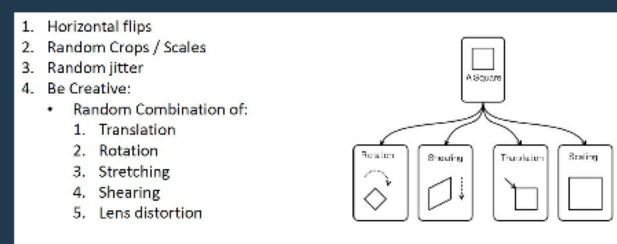
Data Augmentation

- change the pixel without changing the label
- Train on transformed data
- very widely used









Transfer Learning

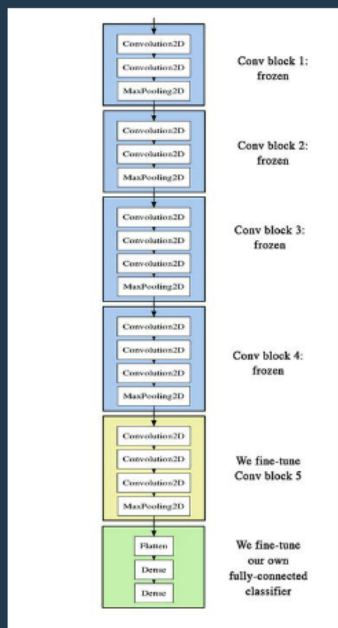
Transfer Learning or Domain Adaptation:

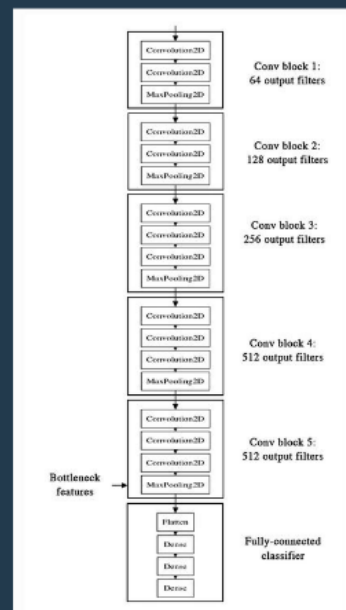
- Very few people train an entire CNN from scratch, because we do not have enough data
- It is common to use from pretrained model trained on very large dataset which contains 1.2 million large images with 1000 categories.
- Using ConvNet as an initialization or feature extractor!
- Two different scenario for using pretrained model:
  - ConvNet as fixed feature extractor: CNN Codes
  - Fine-tuining the Convnet:
    - Replace the last fully-connected layers with new one with random weight and train their weights again.
    - If you have more data, you can retrain more layers with "backprop"

Transfer Learning or Domain Adaptation:

When and how to fine tune:

1. New dataset is small and is similar to original dataset:
   - No need to retrain high level features in CNN – If you do, you overfit
   - Just need to retrain the classifier part of model
2. New dataset is large and is similar to original dataset:
   - Retrain all the weights in the network
3. New dataset is small and are totally different from original dataset:
   - Train the classifier not on top of the network but on top of earlier layers
4. New dataset is large and are totally differenet
   - Just fine tune all the weights but using the pretrained model as an initialization





Prezi

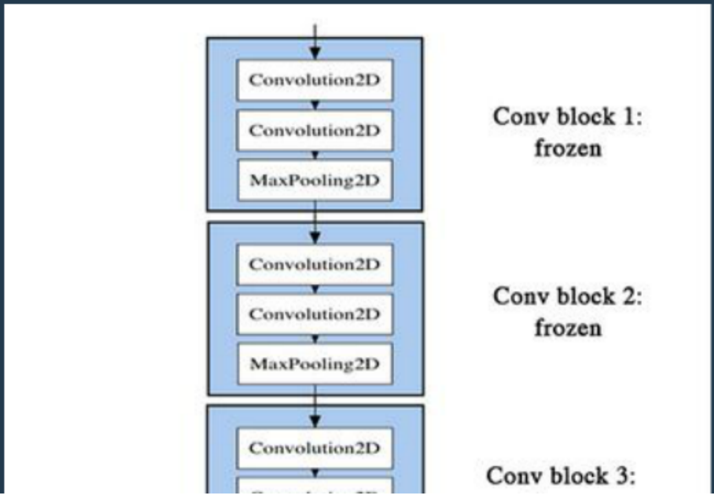
Transfer

Transfer Learning or Domain Adaptation:

- Very few people train an entire CNN from scratch, because we do not have enough data
- It is common to use from pretrained model trained on very large dataset which contains 1.2 million large images with 1000 categories.
- Using ConvNet as an initialization or feature extractor!
- Two different scenario for using pretrained model:
  - ConvNet as fixed feature extractor: CNN Codes
  - Fine-tuining the Convnet:
    - Replace the last fully-connected layers with new one with random weight and train their weights again.
    - If you have more data, you can retrain more layers with “backprop”

Transfer Learning or Domain Adaptation:

- Very few people train an entire CNN from scratch, because we do not have enough data
- It is common to use from pretrained model trained on very large dataset which contains 1.2 million large images with 1000 categories.
- Using ConvNet as an initialization or feature extractor!
- Two different scenario for using pretrained model:
    - ConvNet as fixed feature extractor: CNN Codes
    - Fine-tuining the Convnet:
        - Replace the last fully-connected layers with new one with random weight and train their weights again.
        - If you have more data, you can retrain more layers with "backprop"



er Learning

Transfer Learning or Domain Adaptation:

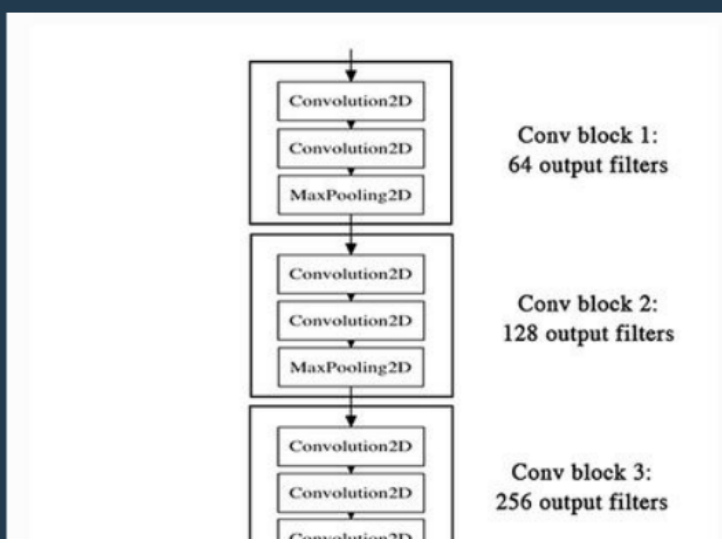
When and how to fine tune:

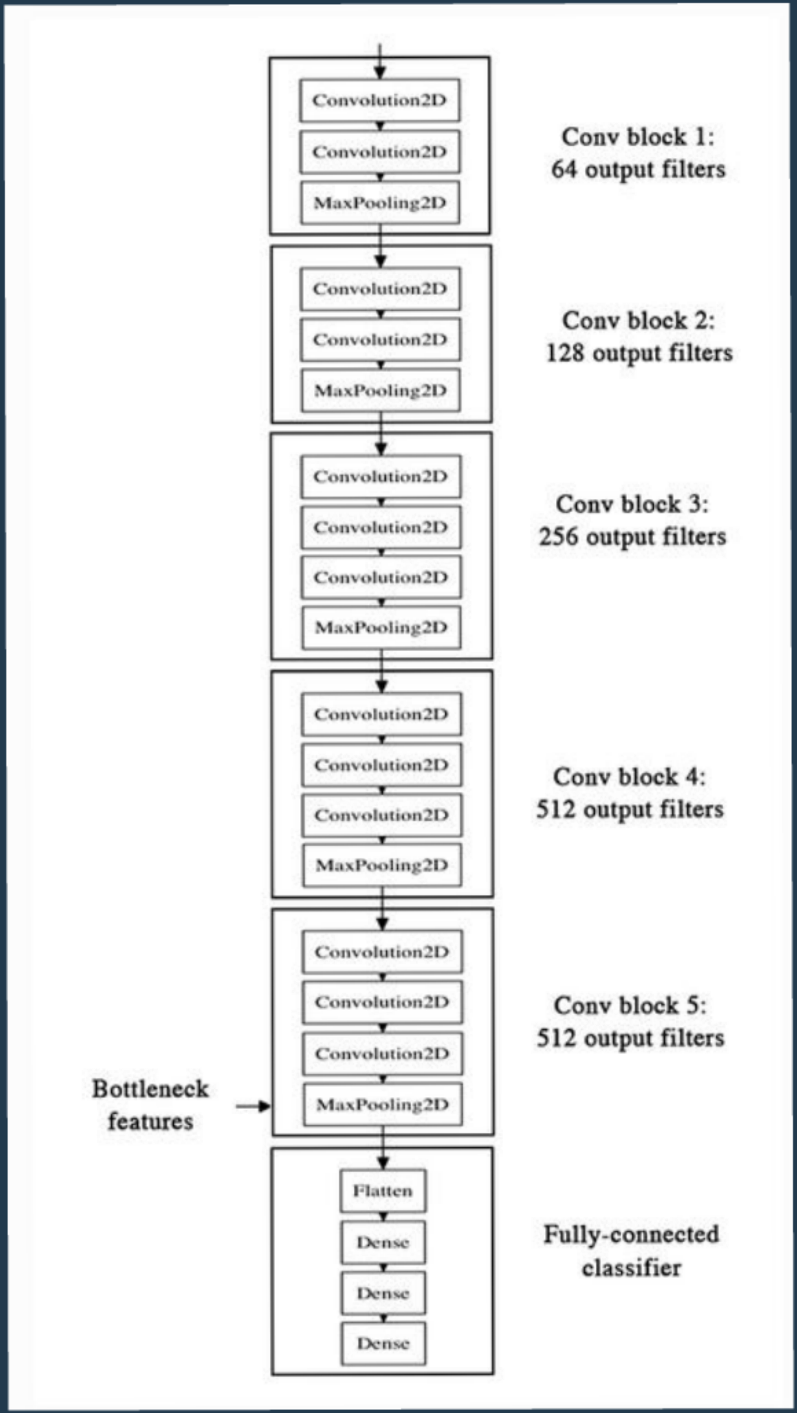
1. New dataset is small and is similar to original dataset:
   - No need to retrain high level features in CNN – If you do, you overfit
   - Just need to retrain the classifier part of model
2. New dataset is large and is similar to original dataset:
   - Retrain all the weights in the network
3. New dataset is small and are totally different from original dataset:
   - Train the classifier not on top of the network but on top of earlier layers
4. New dataset is large and are totally differenet
   - Just fine tune all the weights but using the pretrained model as an initialization

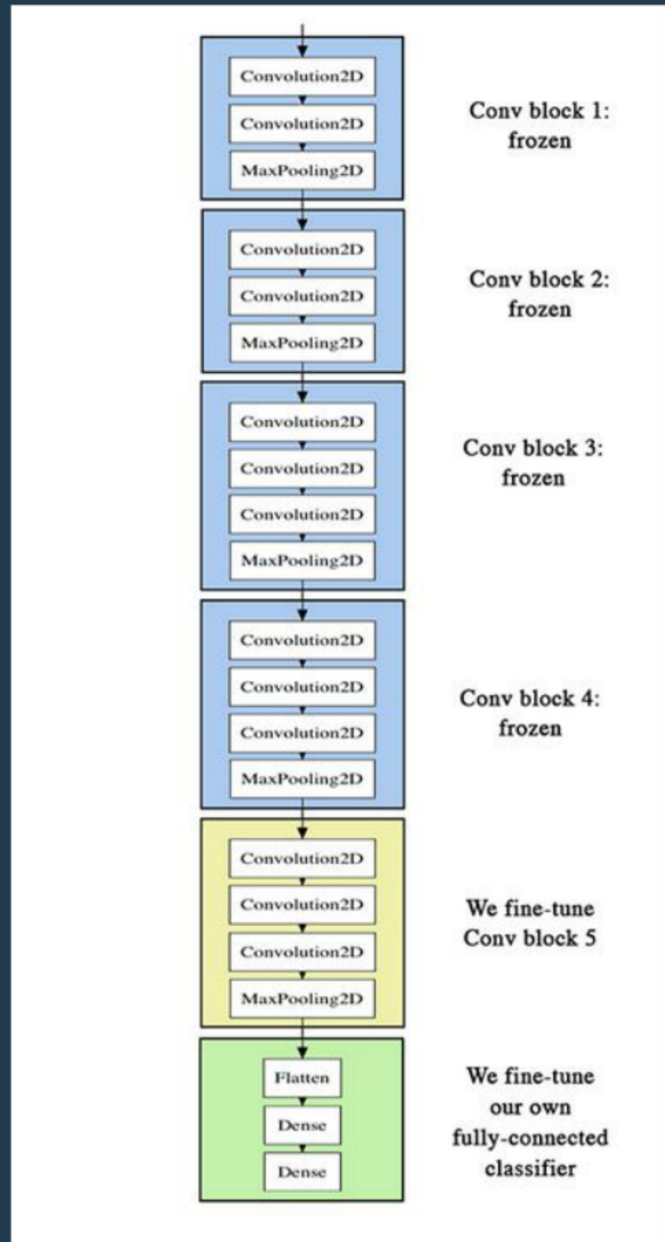
Transfer Learning or Domain Adaptation:

When and how to fine tune:

1. New dataset is small and is similar to original dataset:
   - No need to retrain high level features in CNN – If you do, you overfit
   - Just need to retrain the classifier part of model
2. New dataset is large and is similar to original dataset:
   - Retrain all the weights in the network
3. New dataset is small and are totally different from original dataset:
   - Train the classifier not on top of the network but on top of earlier layers
4. New dataset is large and are totally differenet
   - Just fine tune all the weights but using the pretrained model as an initialization




Convolution2D
Convolution2D
MaxPooling2D
Conv block 1:
64 output filters
Convolution2D
Convolution2D
MaxPooling2D
Conv block 2:
128 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 3:
256 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 4:
512 output filters
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 5:
512 output filters
Bottleneck
features
Flatten
Dense
Dense
Dense
Fully-connected
classifier


Convolution2D
Convolution2D
MaxPooling2D
Conv block 1: frozen
Convolution2D
Convolution2D
MaxPooling2D
Conv block 2: frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 3: frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Conv block 4: frozen
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
We fine-tune Conv block 5
Flatten
Dense
Dense
We fine-tune our own fully-connected classifier

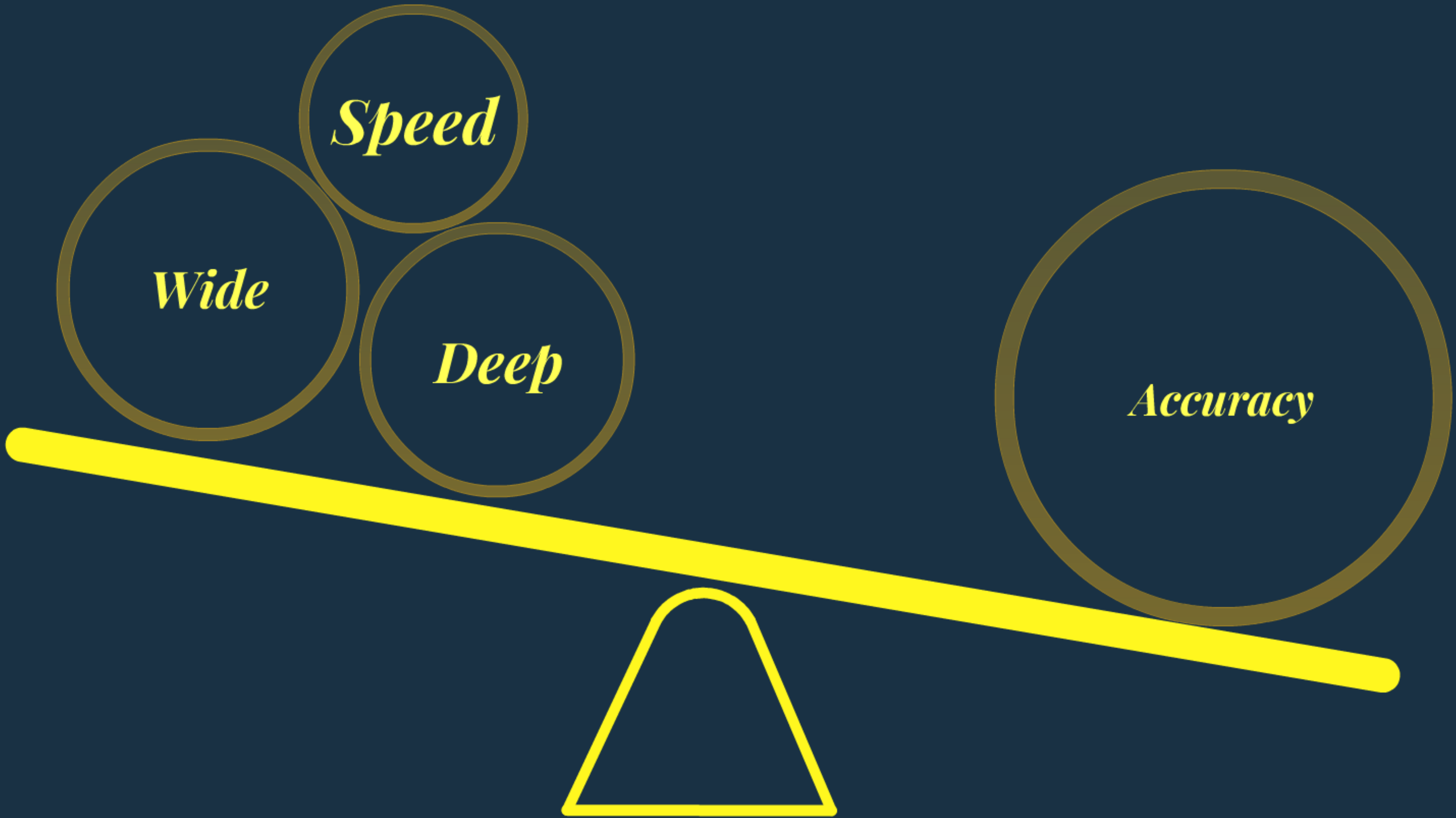
some useful links:

- https://mitpress.mit.edu/books/deep-learning
- https://github.com/Alireza-Akhavan/TensorFlow-Examples
- http://bigdataworkgroup.ir/
- https:/flyyufelix.github.io/2016/10/03/fine-tuning-in-keras-part1.html
- https://github.com/Alireza-Akhavan/class.vision
-
-
- Telegram:
-
- @cvision
-
- @deeplearning.ir
-
- @http://qa.deeplearning.ir
-
- @irandeeplearning

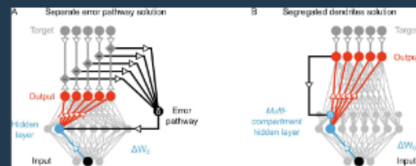
Kaggle.com

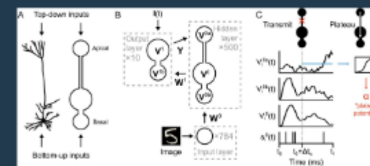
Prezi

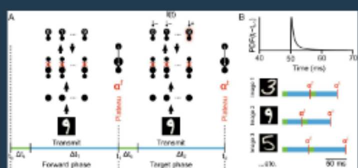
Evaluation

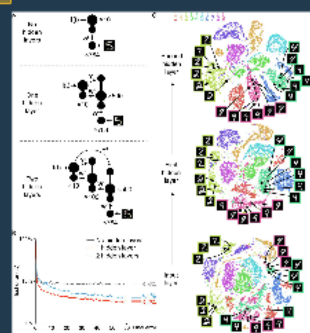


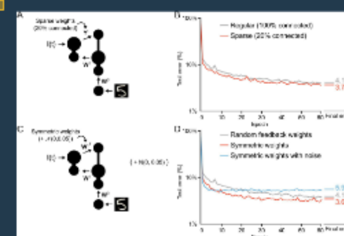
Do Our Brains Use Deep Learning to Make Sense of the World?

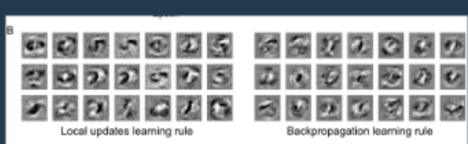








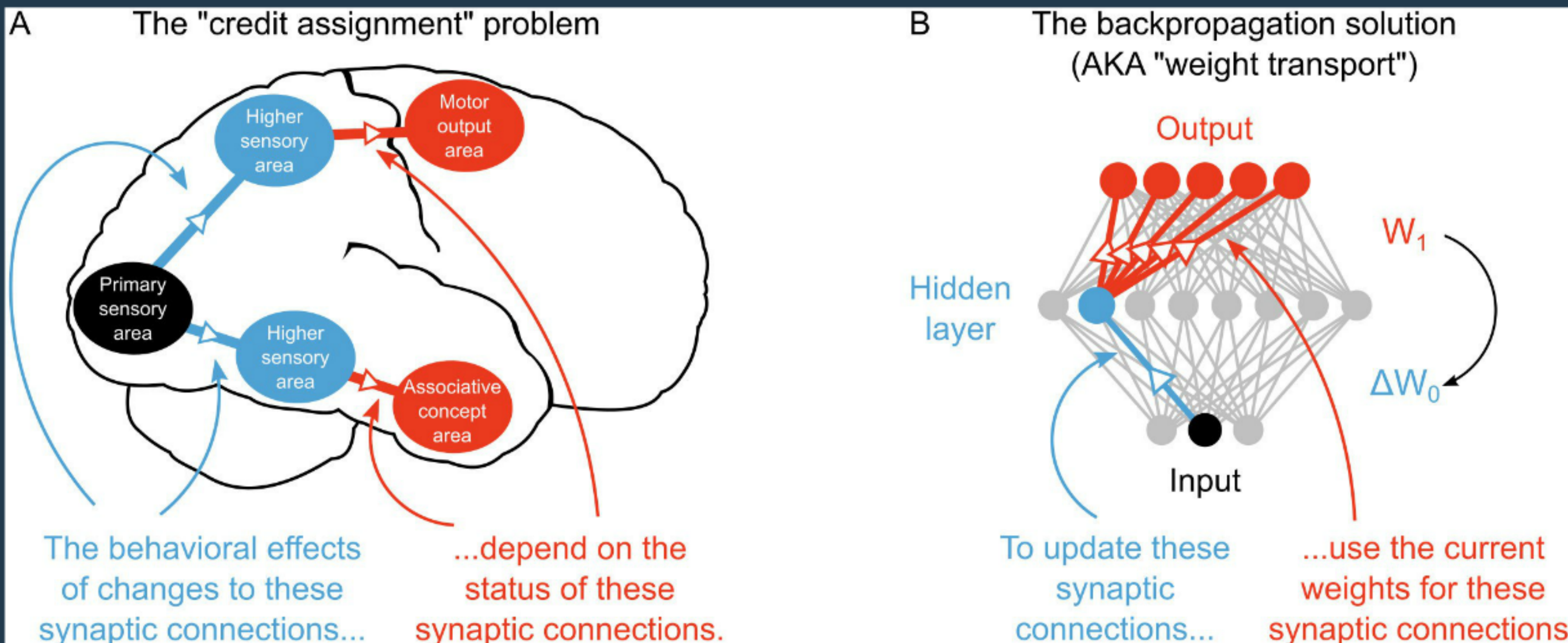




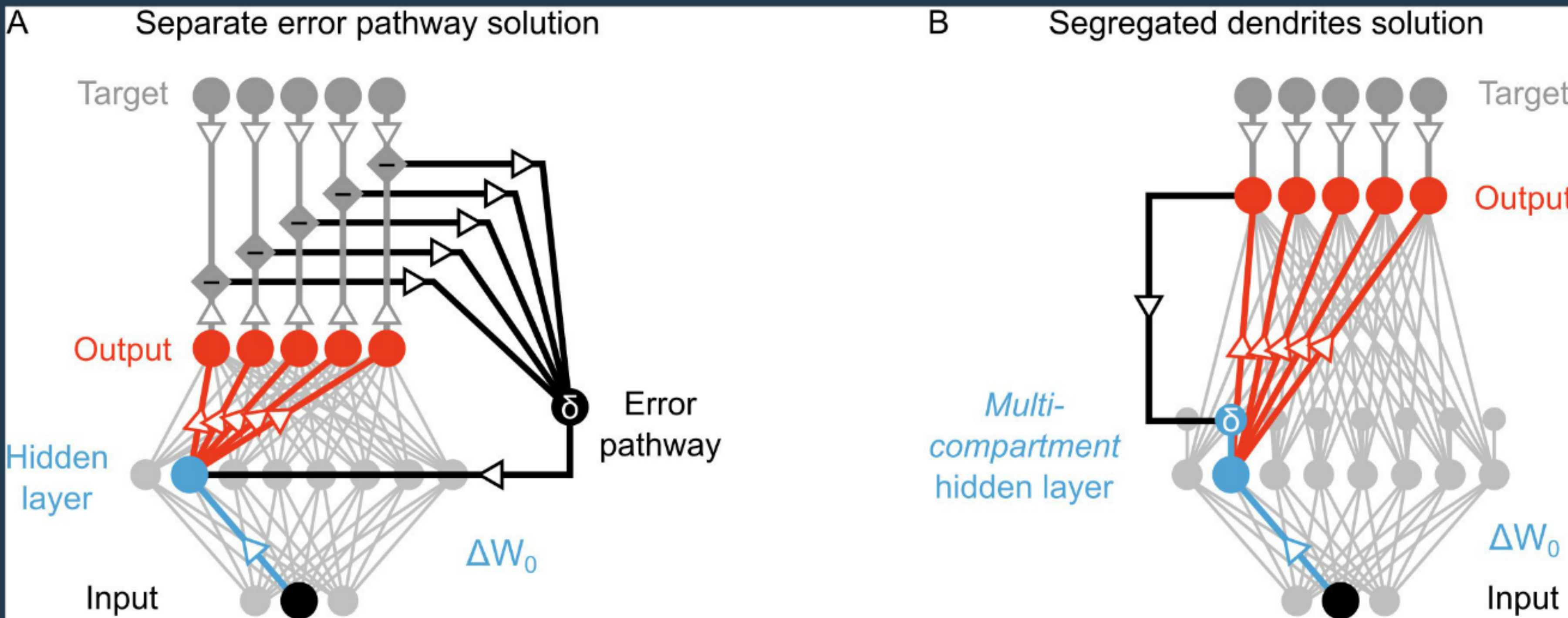
https://elifesciences.org/articles/22901/figures

Prezi

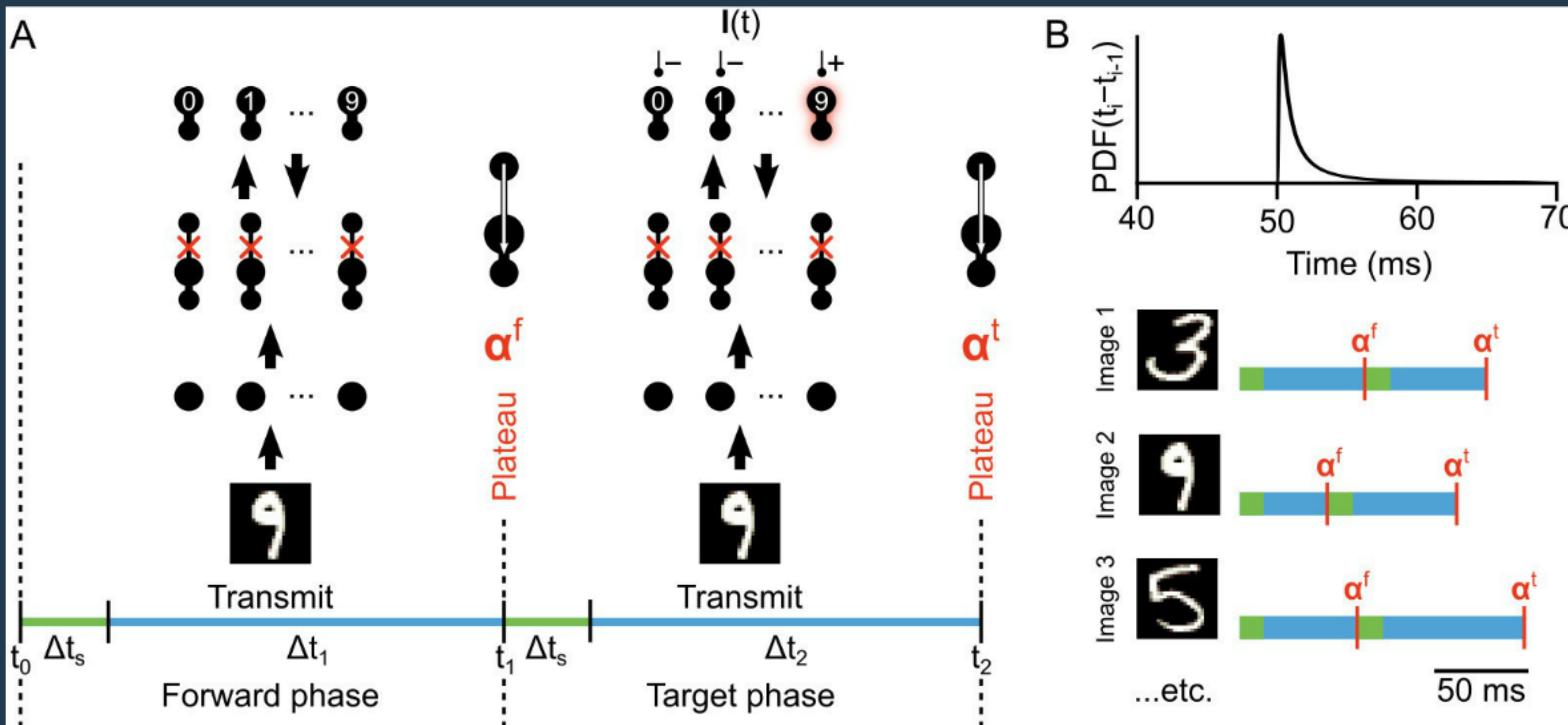
Do Our Brains Use Deep Learning to Make Sense of the World?


A
The "credit assignment" problem
Higher sensory area
Motor output area
Primary sensory area
Higher sensory area
Associative concept area
The behavioral effects of changes to these synaptic connections...
...depend on the status of these synaptic connections.
B
The backpropagation solution (AKA "weight transport")
Output
W_1
Hidden layer
ΔW_0
Input
To update these synaptic connections...
...use the current weights for these synaptic connections.

Prezi

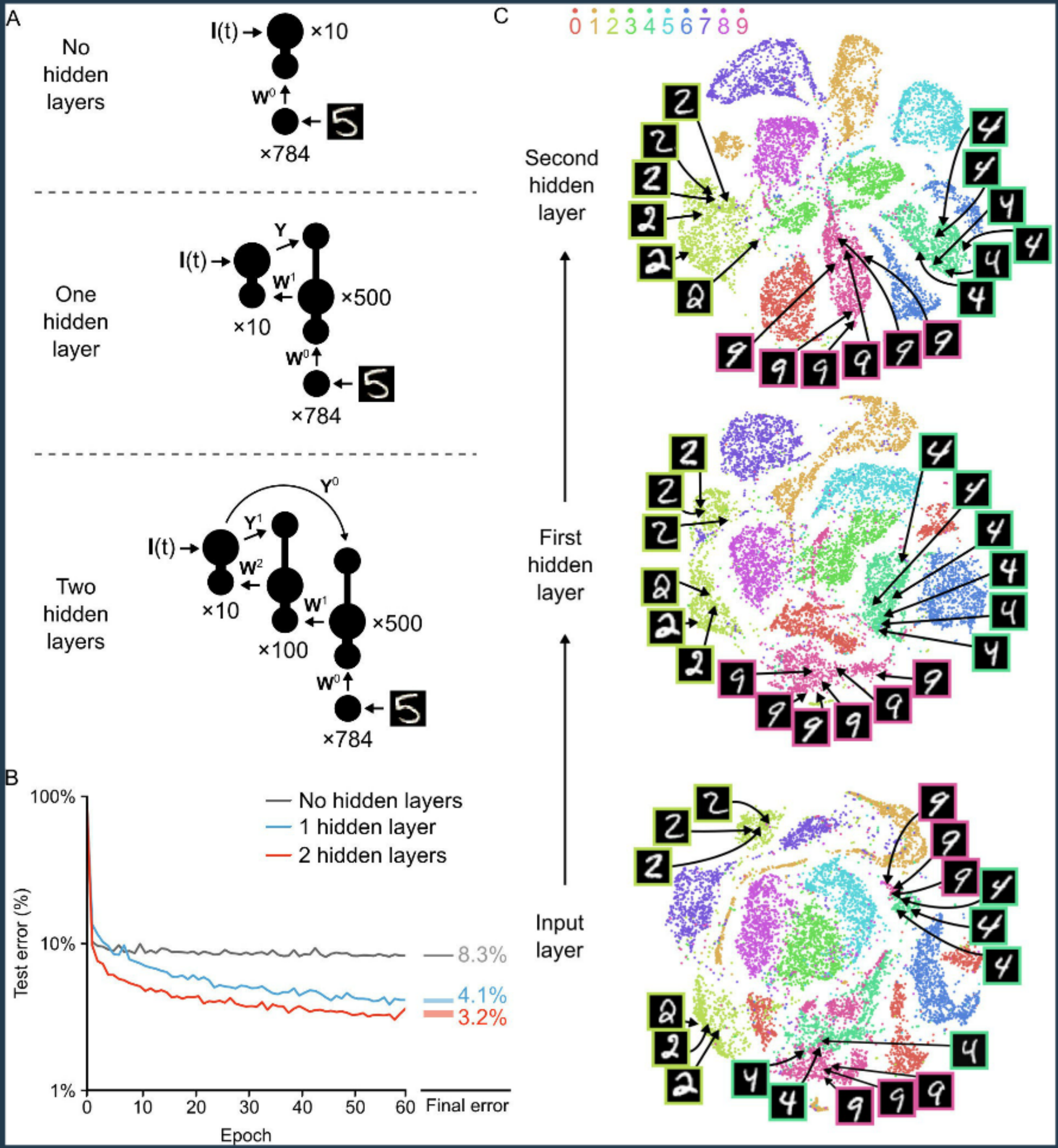


A
Separate error pathway solution
Target
Output
Error pathway
δ
Hidden layer
ΔW0
Input
B
Segregated dendrites solution
Target
Output
δ
Multi-compartment hidden layer
ΔW0
Input

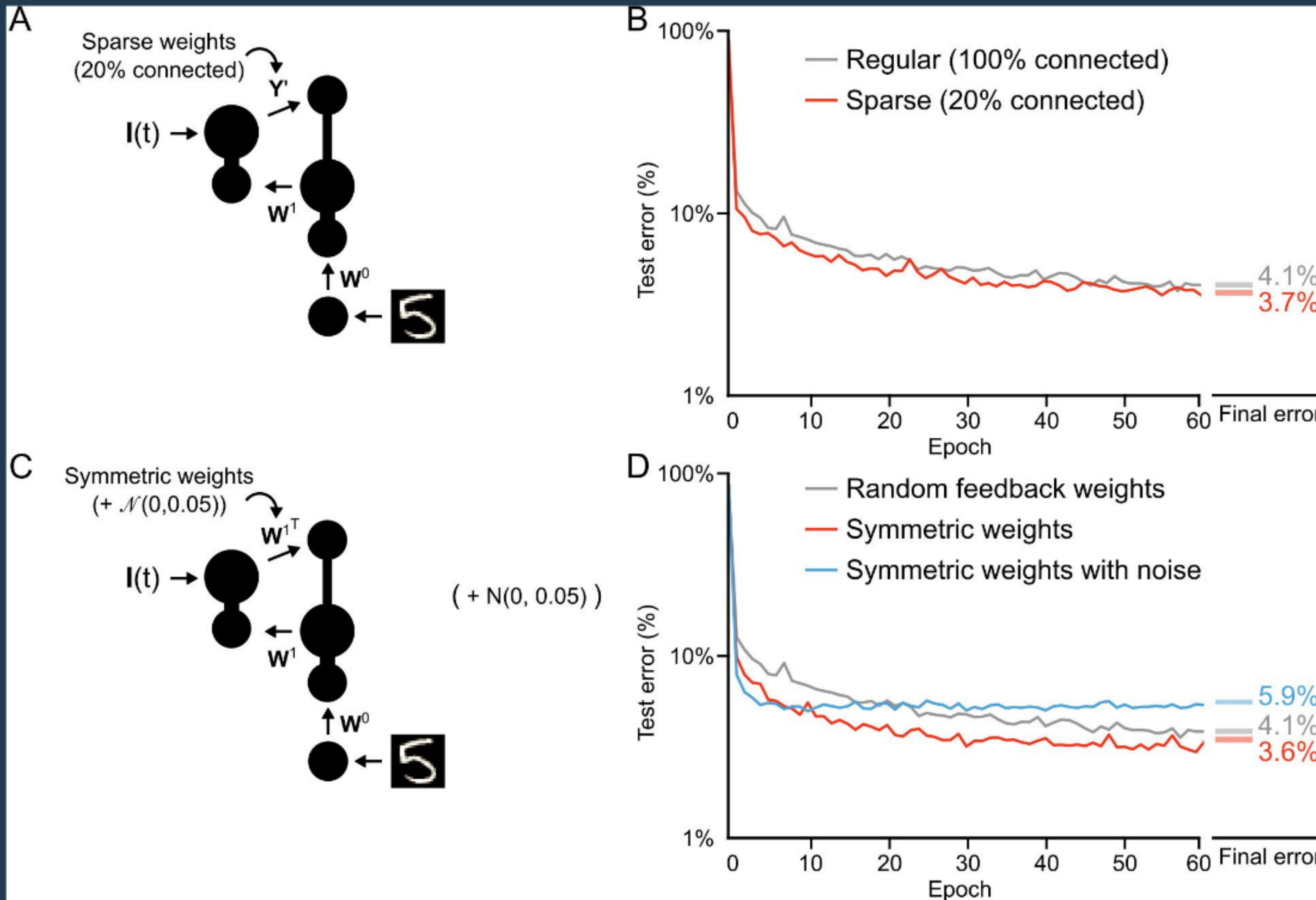


A
I(t)
−
−
+
0
1
...
9
α^f
α^t
Plateau
Plateau
Transmit
Transmit
t_0
Δt_s
Δt_1
t_1
Δt_s
Δt_2
t_2
Forward phase
Target phase
B
$PDF(t_i - t_{i-1})$
40
50
60
70
Time (ms)
Image 1
Image 2
Image 3
...etc.
50 ms

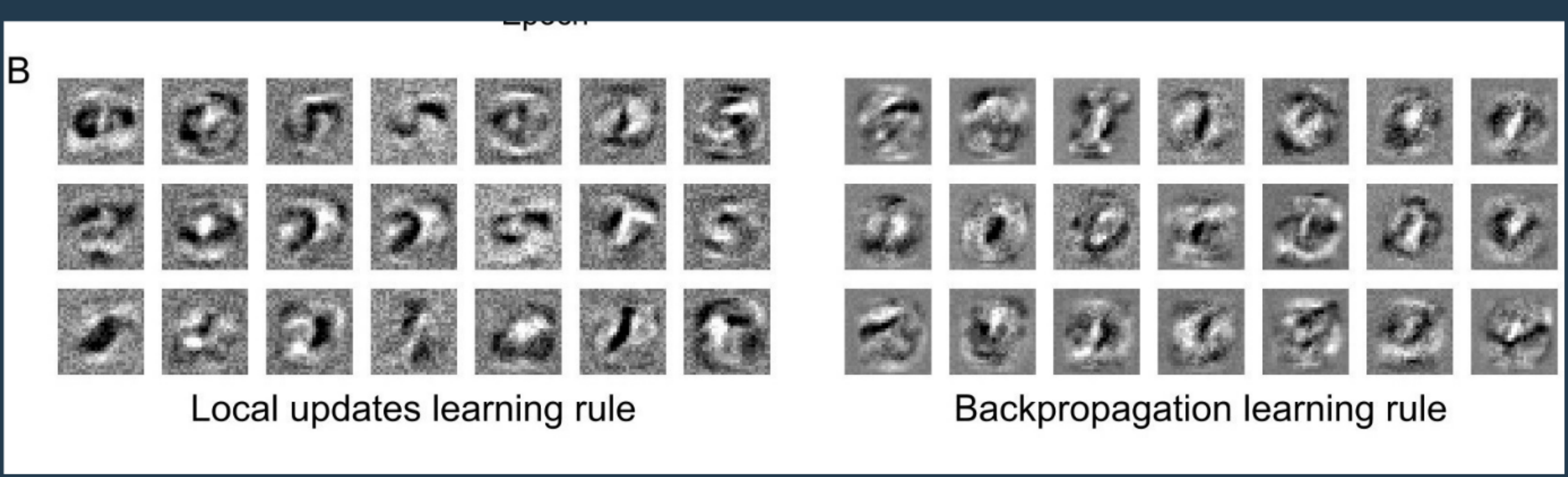
Prezi


A
No hidden layers
I(t)
×10
W0
×784
One hidden layer
Y
W1
×500
×10
W0
×784
Two hidden layers
Y0
Y1
W2
×10
W1
×500
×100
W0
×784
B
100%
10%
1%
Test error (%)
No hidden layers
1 hidden layer
2 hidden layers
8.3%
4.1%
3.2%
0
10
20
30
40
50
60
Epoch
Final error
C
0 1 2 3 4 5 6 7 8 9
Second hidden layer
First hidden layer
Input layer

Prezi


A
Sparse weights
(20% connected)
Y'
I(t)
W1
W0
5
B
100%
Regular (100% connected)
Sparse (20% connected)
10%
Test error (%)
4.1%
3.7%
1%
0 10 20 30 40 50 60
Epoch
Final error
C
Symmetric weights
(+ N(0,0.05))
W1T
I(t)
W1
W0
5
(+ N(0, 0.05))
D
100%
Random feedback weights
Symmetric weights
Symmetric weights with noise
10%
Test error (%)
5.9%
4.1%
3.6%
1%
0 10 20 30 40 50 60
Epoch
Final error

Prezi



Epoch
B
Local updates learning rule
Backpropagation learning rule

Prezi

Deep Learning Frameworks

TensorFlow

Caffe

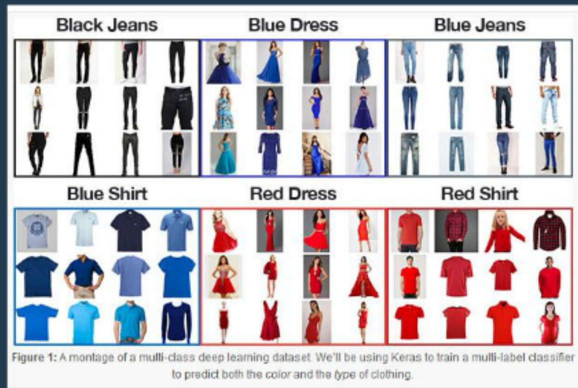
Keras

Theano

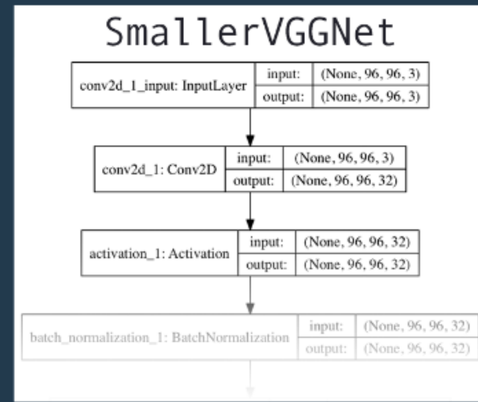
Lasagne

Practice!

- Transfer Learning (Code)
- Classify new category (Code)
- multi label classification
- Speech Recognition



pyimagesearch.com



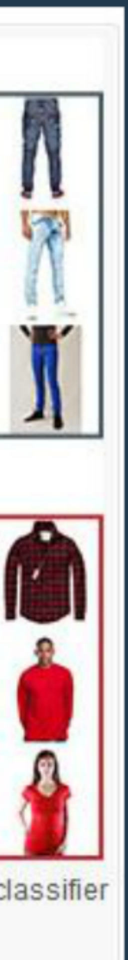
pyimagesearch.com

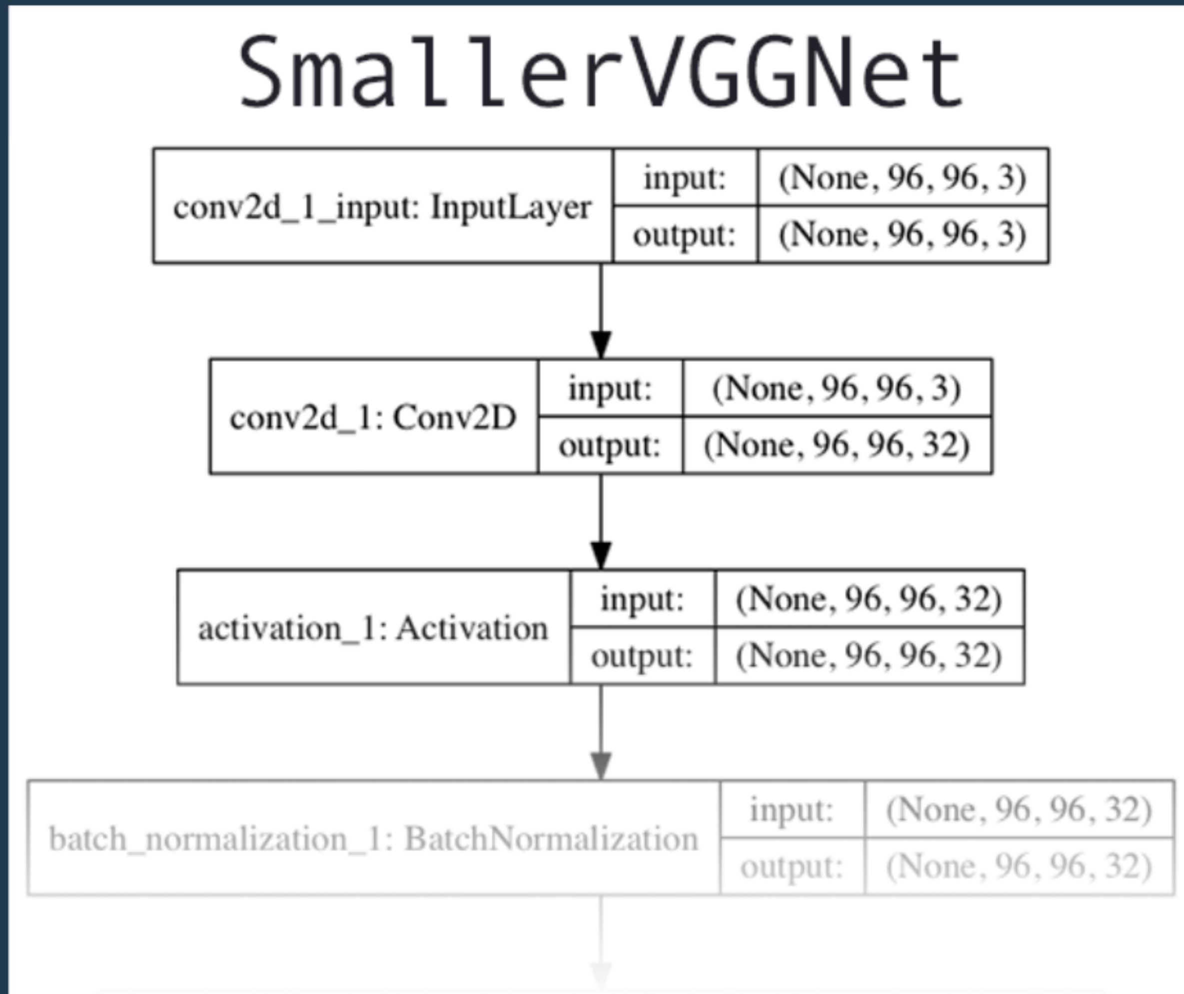
pyimagesearch.com

Figure 1: A montage of a multi-class deep learning dataset. We'll be using Keras to train a multi-label classifier to predict both the *color* and the *type* of clothing.

pyimagesearch.com

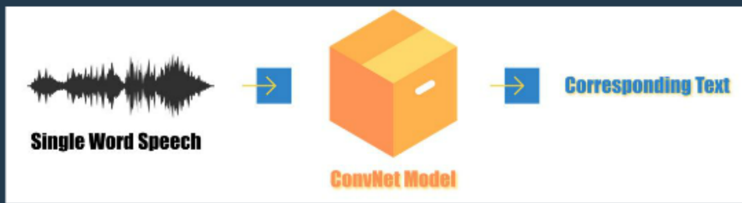
pyimagesea

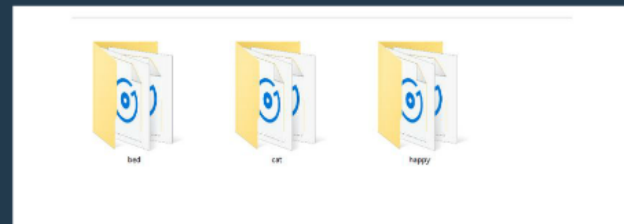


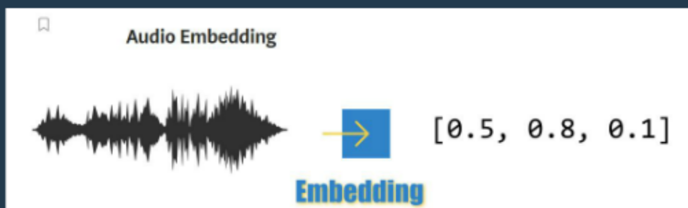


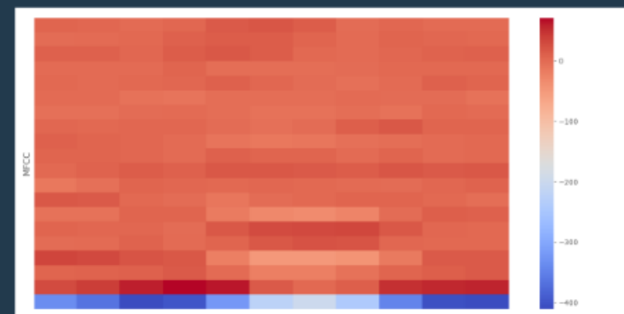
pyimagesearch.com

Prezi

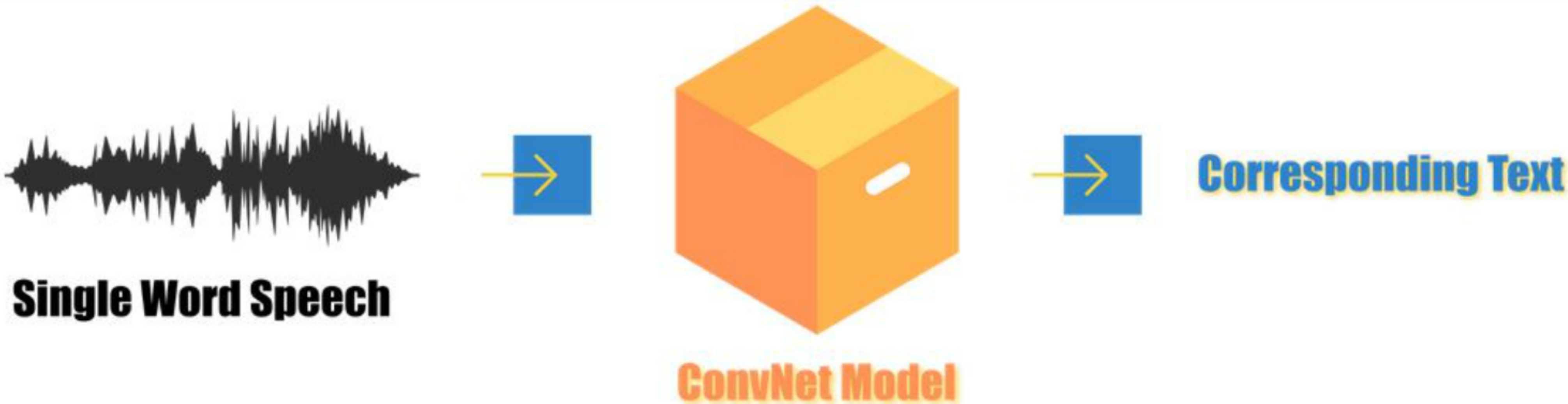




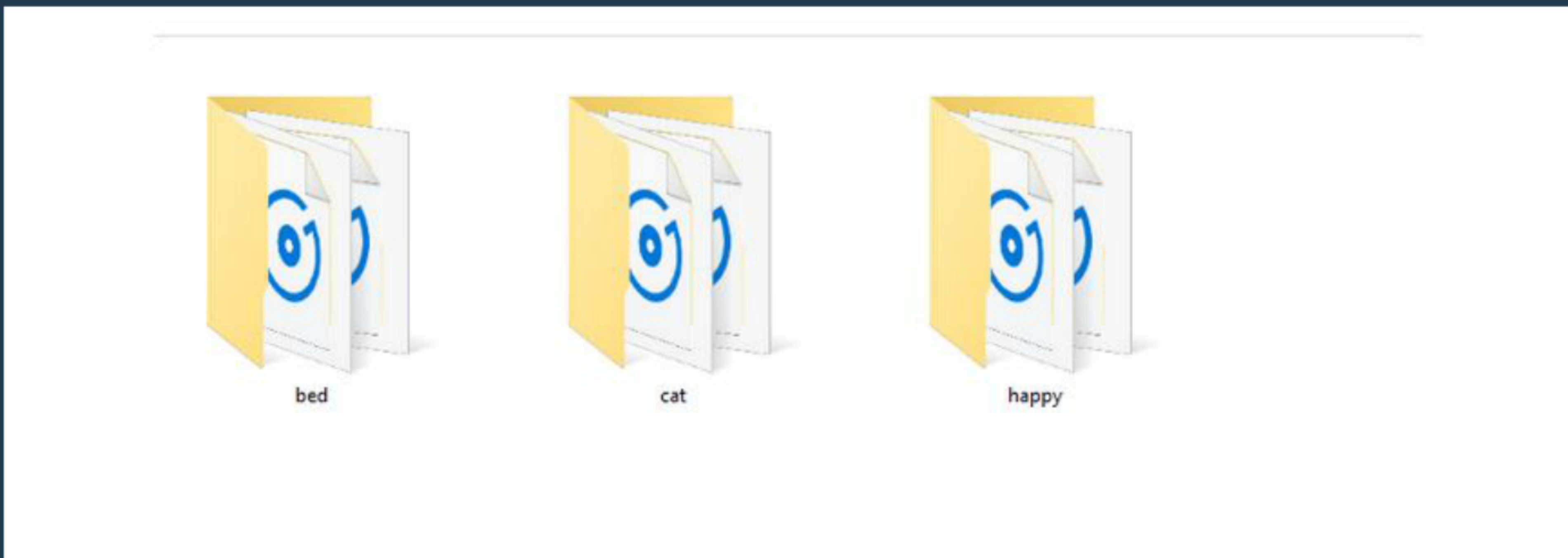




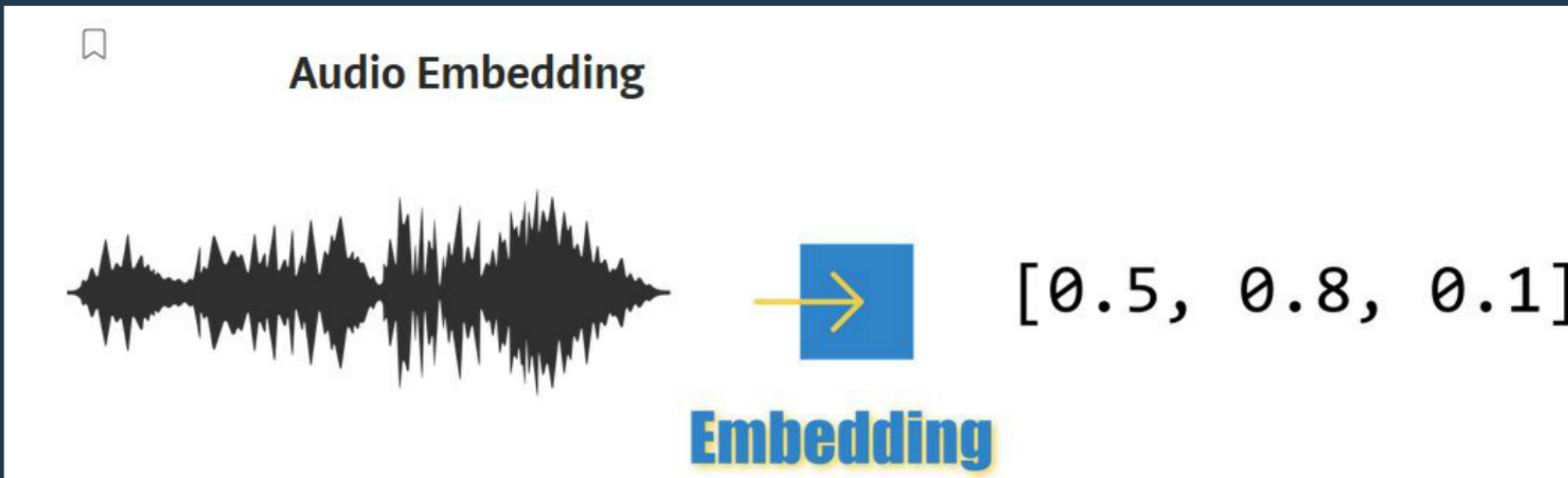
https://blog.manash.me/building-a-dead-simple-word-recognition-engine-using-convnet-in-keras-25e72c19c12b

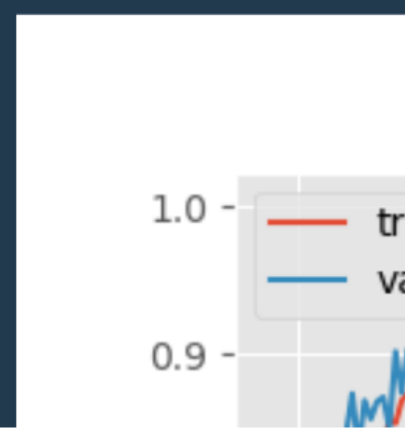


Audio Embedding


bed
cat
happy


Single Word Speech
ConvNet Model


Audio Embedding
[0.5, 0.8, 0.1]
Embedding


1.0
0.9

cat

happy

Deep Learning

What?

Why?

How?

Thank You for attention :)

Any Question?

Thank You for attention :)

Any Question?