



LAB 0

Warm up

CHECKLIST

- ✓ Repository → <https://github.com/fernанortega/AIWorkshop20>
- ✓ Azure account (Free) and Data Science VM
- ✓ Anaconda >= 5.3
- ✓ Python 3.6
- ✓ IDE → VS Code (or Atom...)
- ✓ Jupyter notebooks

AI Workshop

Initial Setup

- Follow the instruction on folder Lab0
- Choose your setup



- Execute notebook “*Lab00_SmokeTest.ipynb*” to test the environment is ready.

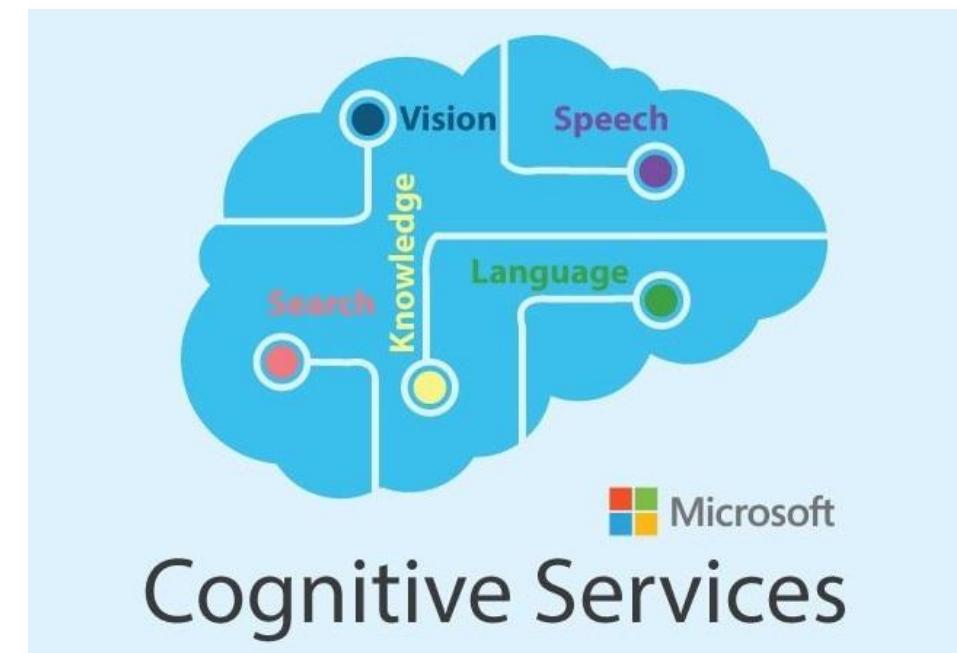


LAB 1

Cognitive Services

Cognitive Services

- Explore and visualize the dataset
- Deploy Cognitive Services in your subscription
- Ask to the cognitive services necessary in order to extract some info.
- <https://docs.microsoft.com/es-es/azure/cognitive-services/>
- Extra: use Custom Vision service
- Use notebook “*Lab01_CognitiveServices.ipynb*”



Cognitive Services



Key Phrase extraction



Location entity extraction



Sentiment analysis



Organization entity extraction



Persons entity extraction



Language detection



Face detection



Celebrity recognition



Tag extraction



Custom skills



Landmark detection



Printed text recognition

Dataset



axes



boots



carabiners



crampons



gloves



hardshell_jackets



harnesses



helmets



insulated_jackets



pulleys



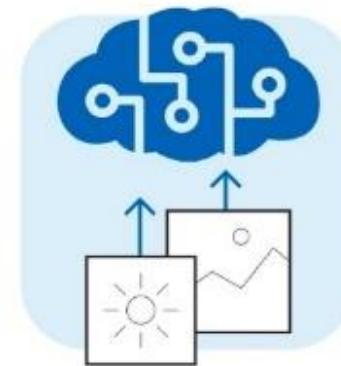
rope



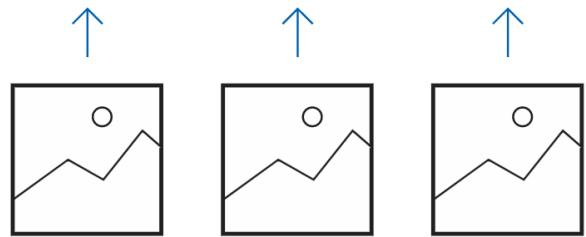
tents

Extra: Custom Vision

- Create a Custom Vision model easy to train and deploy
- Test it with some photos
- <https://docs.microsoft.com/es-es/azure/cognitive-services/custom-vision-service/>
- Use notebook “*Lab02_CustomVision.ipynb*”

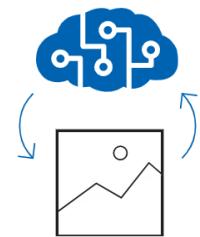


Extra: Custom Vision



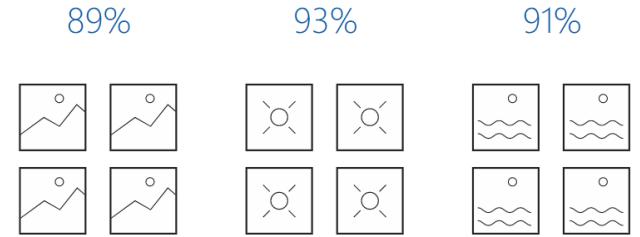
Upload Images

Bring your own labeled images, or use Custom Vision to quickly add tags to any unlabeled images.



Train

Use your labeled images to teach Custom Vision the concepts you care about.



Evaluate

Use simple REST API calls to quickly tag images with your new custom computer vision model.

[Iterations](#)[Prediction URL](#)[✓ Make default](#)[Delete](#)[Export](#)Probability Threshold: 90% [\(i\)](#)

Iteration 1

Finished training on **12/31/2017, 8:34:58 PM** using **General** domain

Iteration 2

Trained : 2 hours ago
with General domain

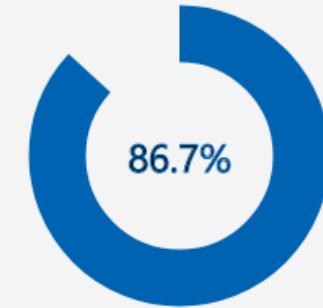
Iteration 1

Trained : 3 hours ago
with General domain

Precision [\(i\)](#)



Recall [\(i\)](#)



Performance Per Tag

Tag	Precision	Recall
Black-grass	98.0%	95.3%
Charlock	94.0%	82.1%
Cleavers	80.5%	89.2%
Common Chickweed	96.2%	70.0%



LAB 2

Data Preparation

Data preparation

- Preprocess the data
 - Scale images
 - Normalize them
- Save the resultant dataset



Pillow

<https://pillow.readthedocs.io/en/stable/>

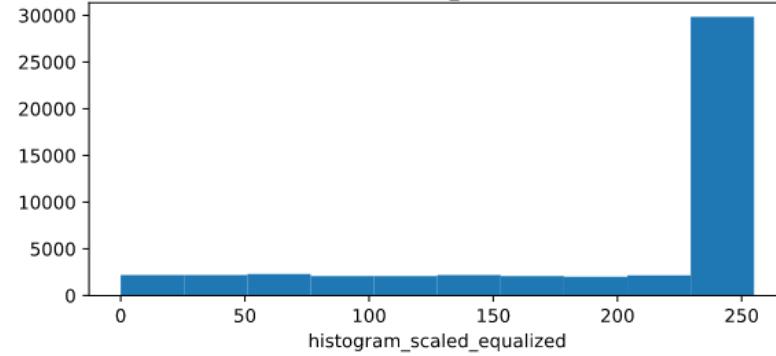
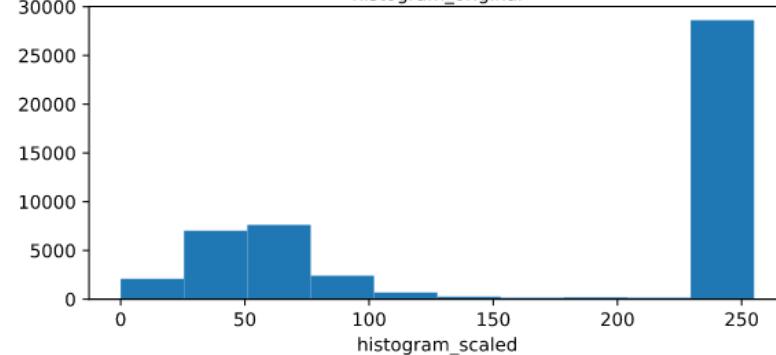
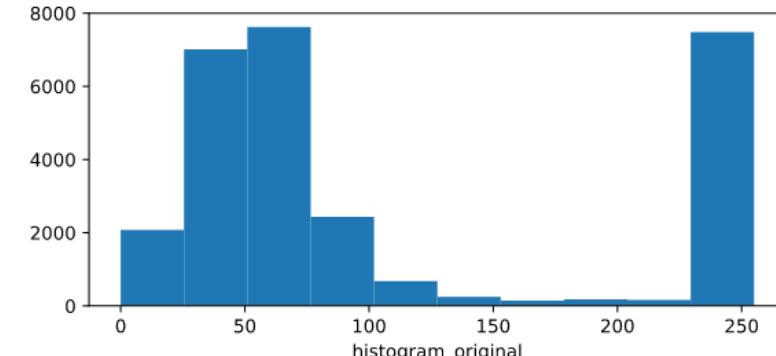


<https://numpy.org/>

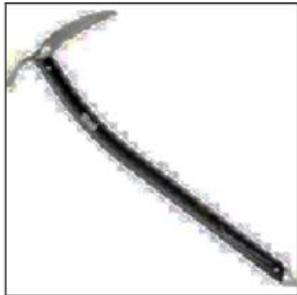
Scale



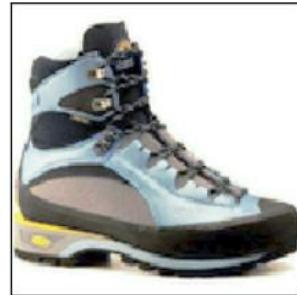
Normalize



Processed



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



carabiners



LAB 3

Train Models

Train Models

- Use a Machine Learning algorithm
- Use notebook “Lab04_MachineLearning101.ipynb”
- Use scikit-learn and any of its non deterministic algorithms.
- <https://scikit-learn.org/stable/>
- Use Azure Machine learning Workspace to train it and record metrics
- Save the model and register it.



Azure Machine Learning

Machine learning on Azure

Sophisticated pretrained models

To simplify solution development

Popular frameworks

To build advanced deep learning solutions

Productive services

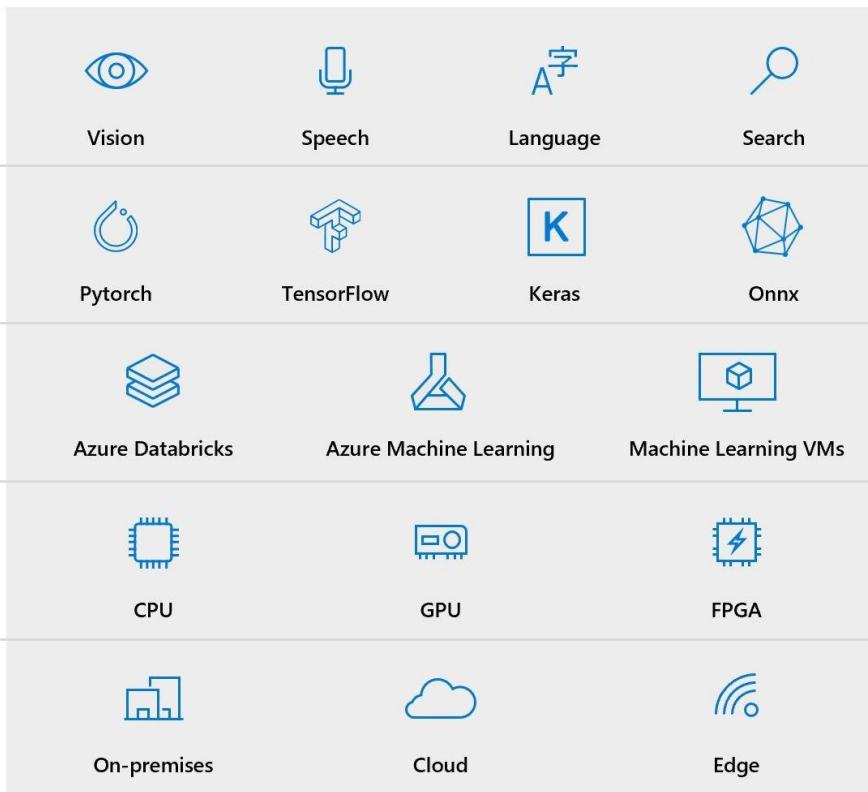
To empower data science and development teams

Powerful infrastructure

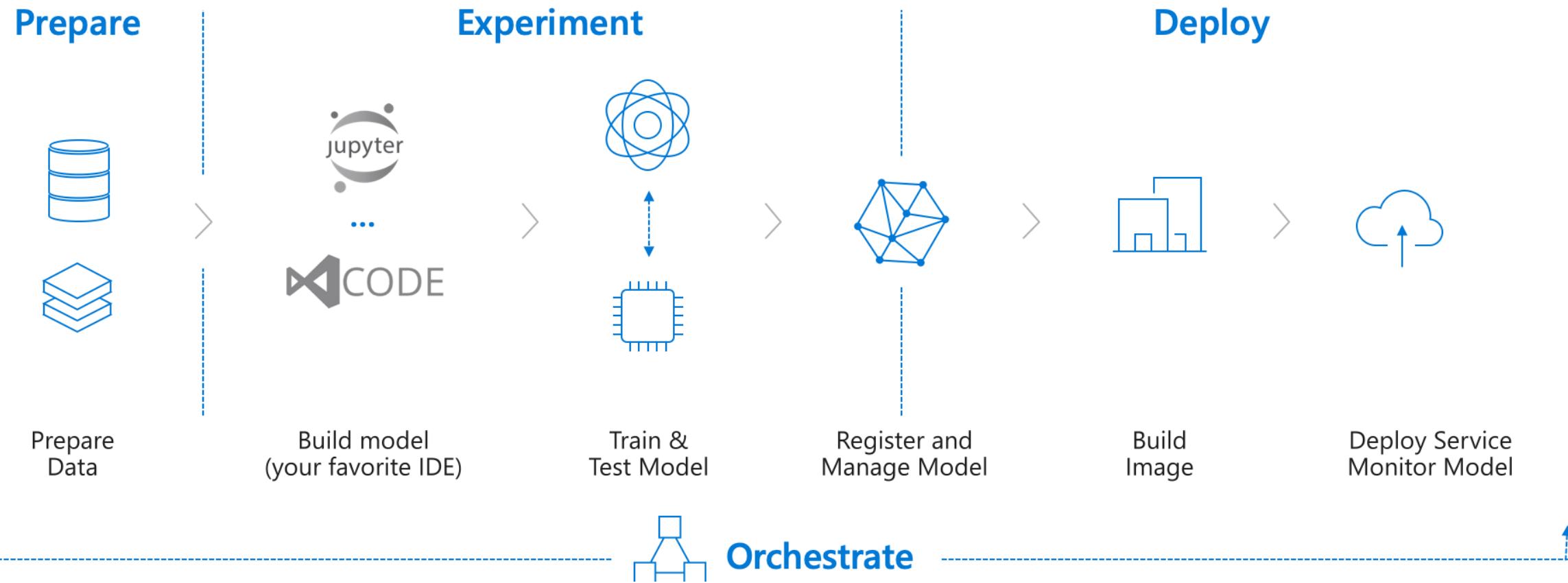
To accelerate deep learning

Flexible deployment

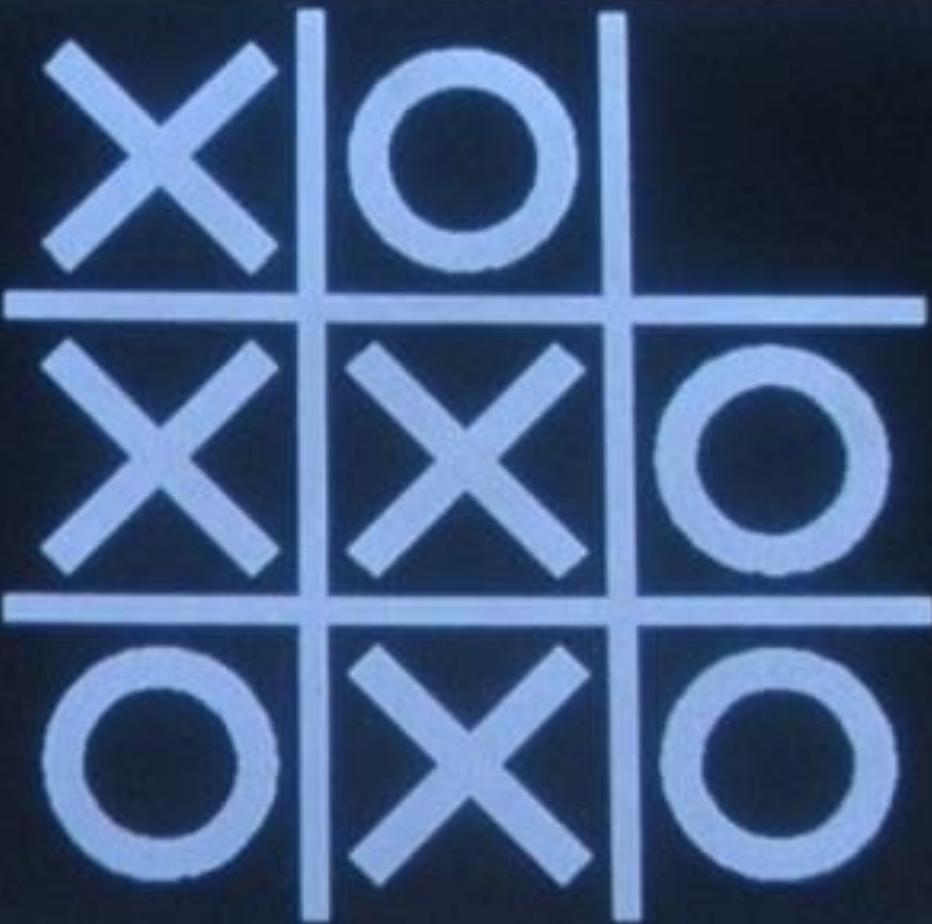
To deploy, manage models on intelligent cloud & edge



Azure Machine Learning



SHALL WE PLAY A LITTLE GAME?



YLING
4
COM STS
8365

GPM STS
3603

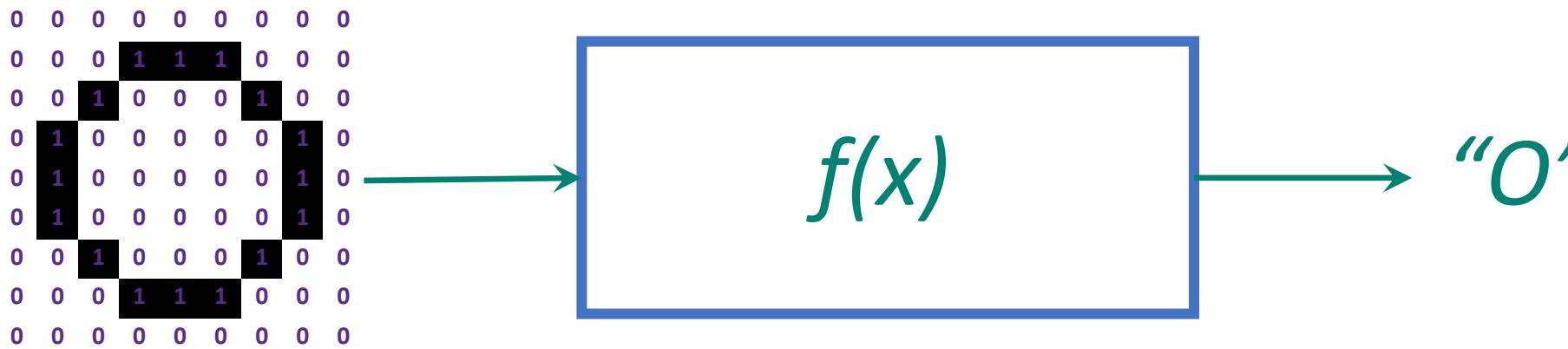
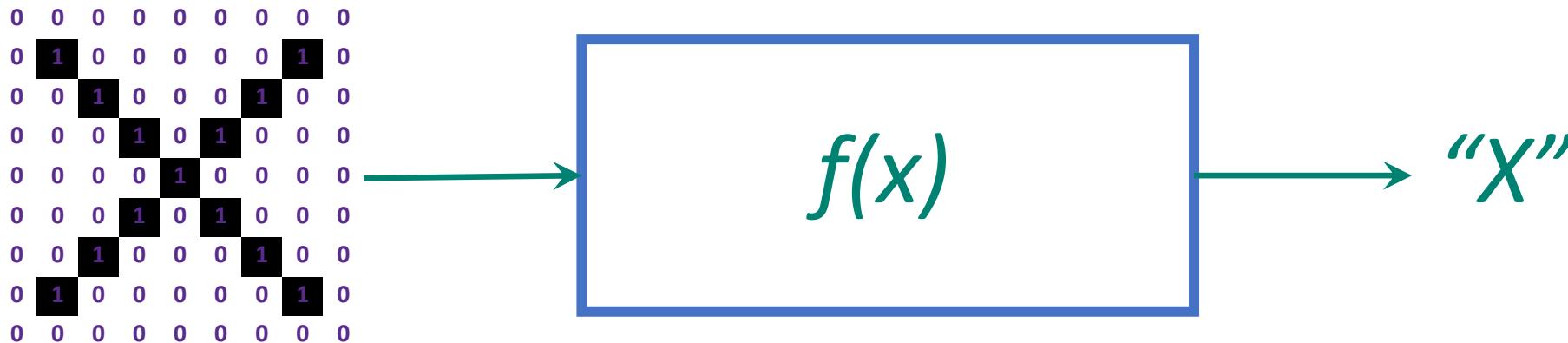
RPL STS
9071

EOT STS
4531

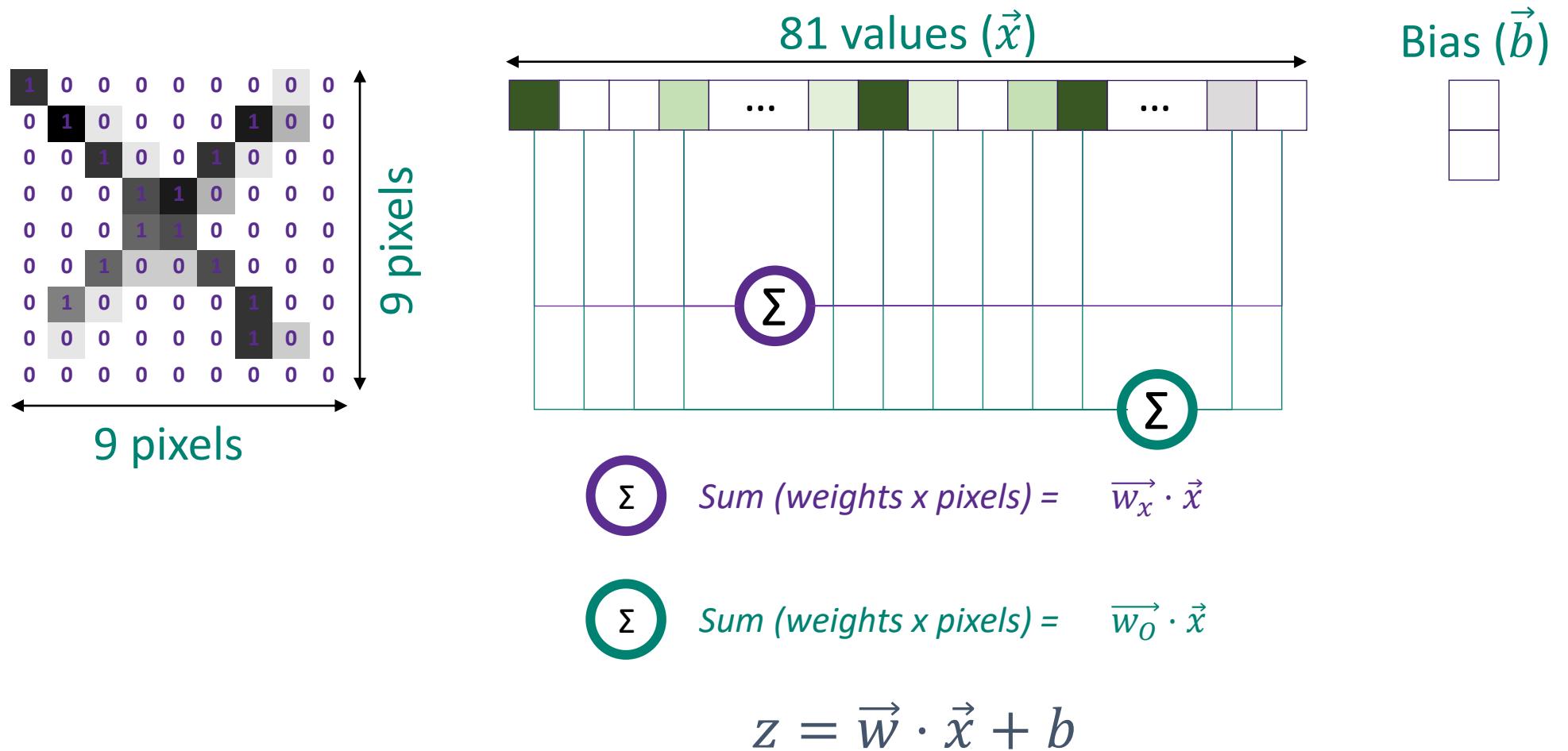
PAC STS
8838

ADL STS
6632

A SIMPLE EXAMPLE

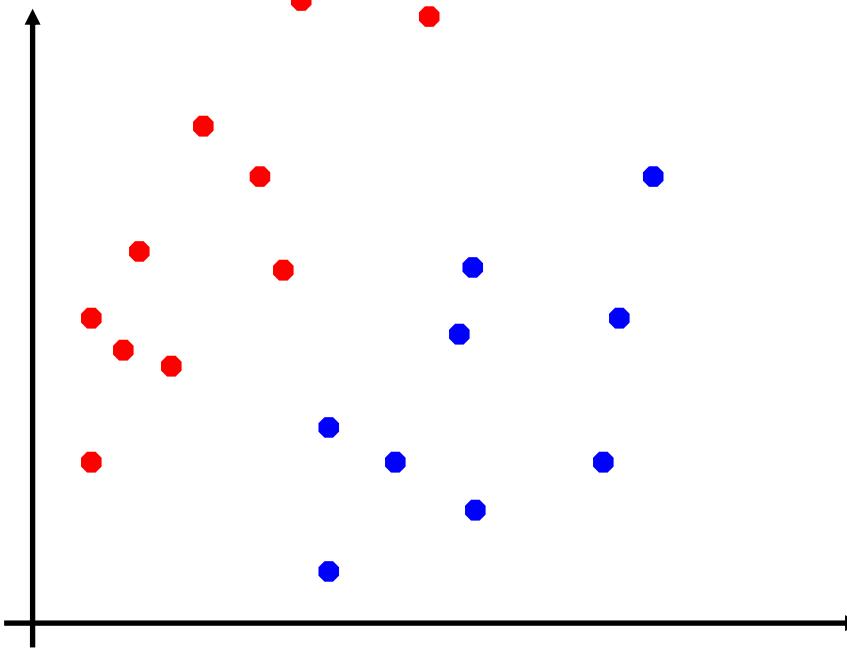


A SIMPLE EXAMPLE



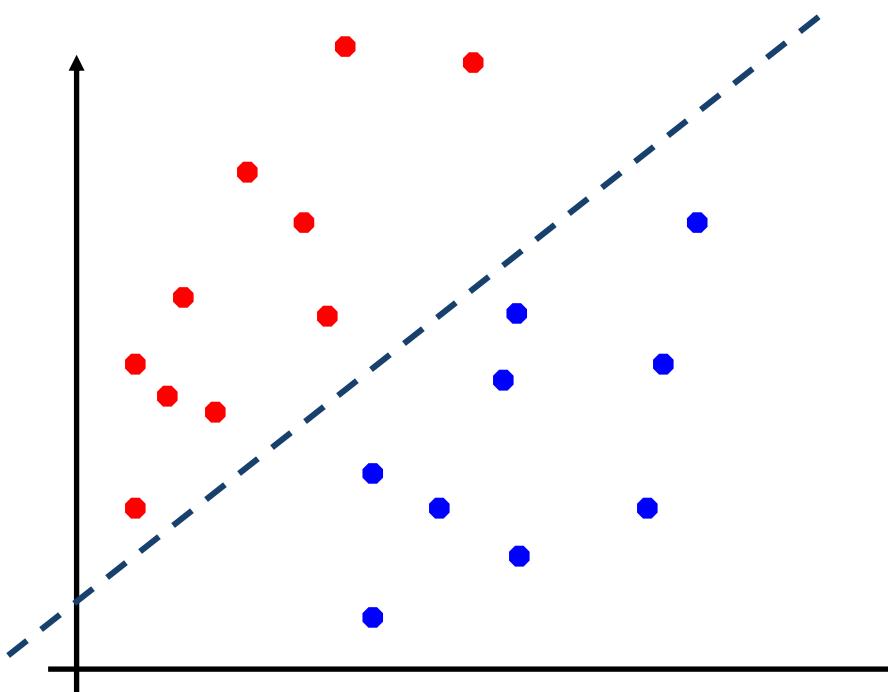
NOW THAT YOU MENTION IT...

$$z = w \cdot \vec{x} + b$$



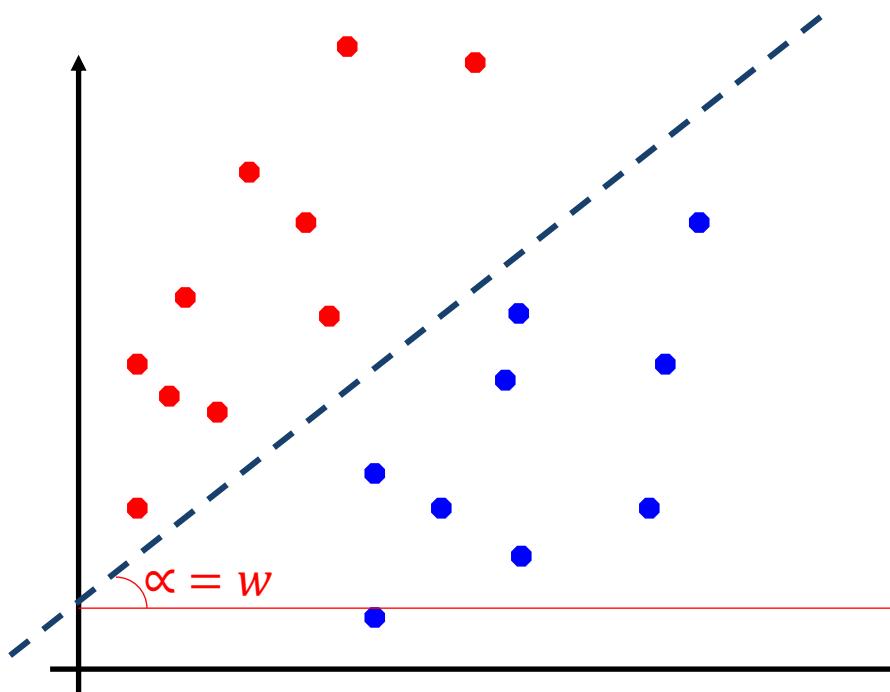
NOW THAT YOU MENTION IT...

$$z = \mathbf{w} \cdot \vec{x} + b$$

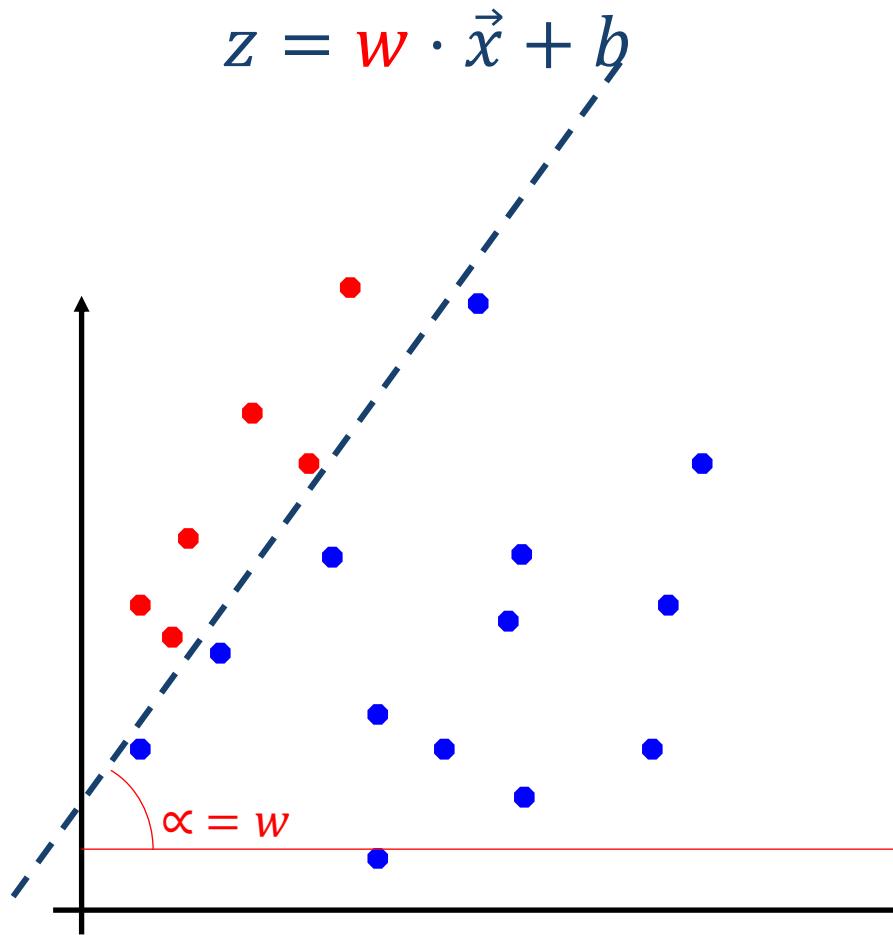


NOW THAT YOU MENTION IT...

$$z = \mathbf{w} \cdot \vec{x} + b$$

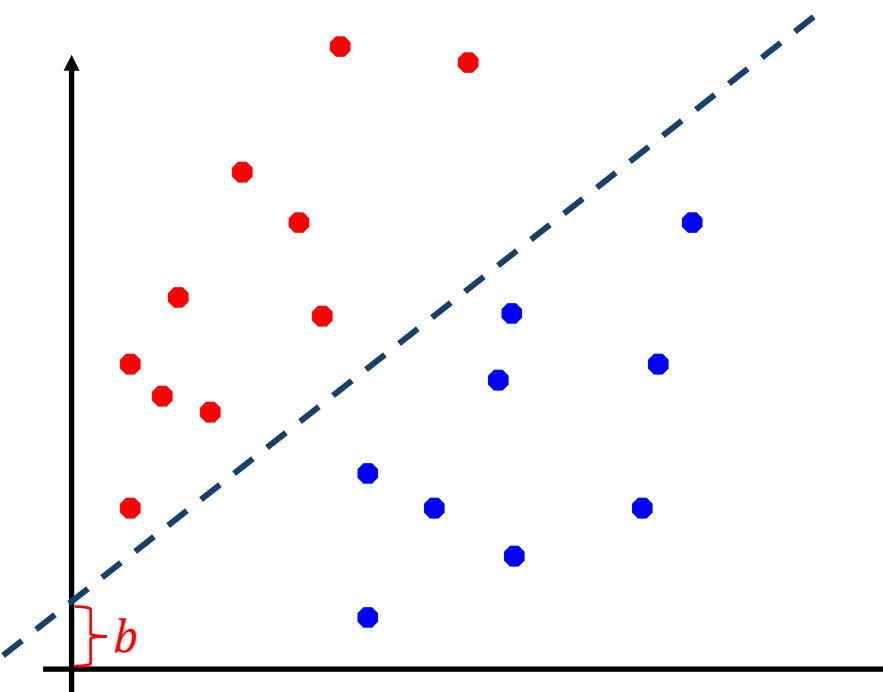


NOW THAT YOU MENTION IT...



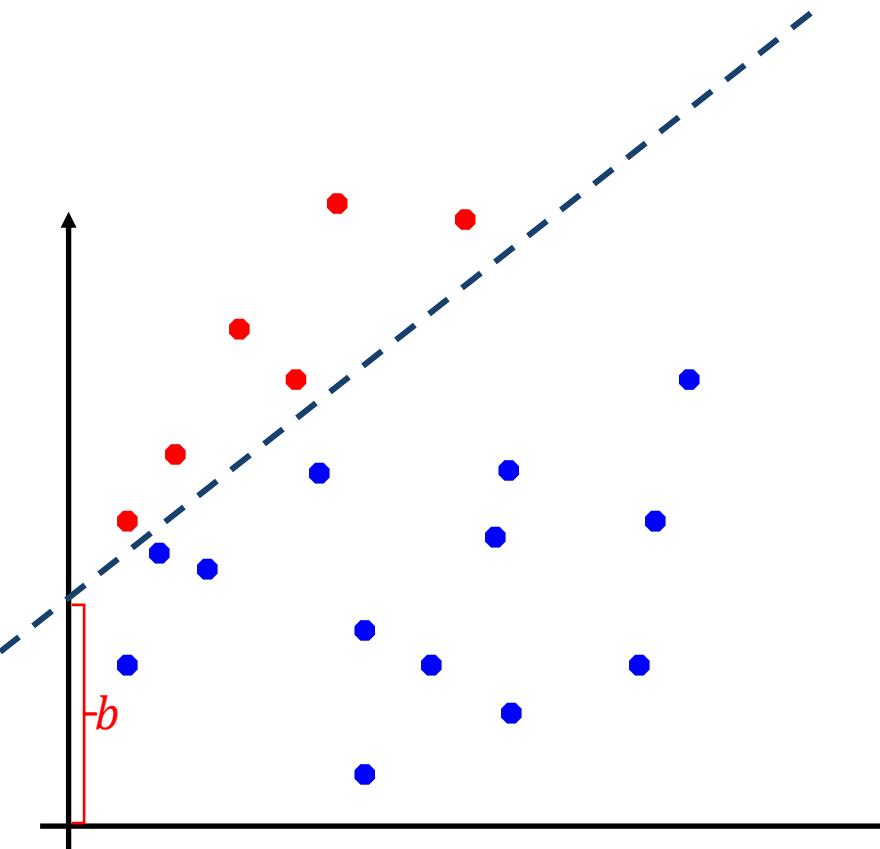
NOW THAT YOU MENTION IT...

$$z = w \cdot \vec{x} + b$$

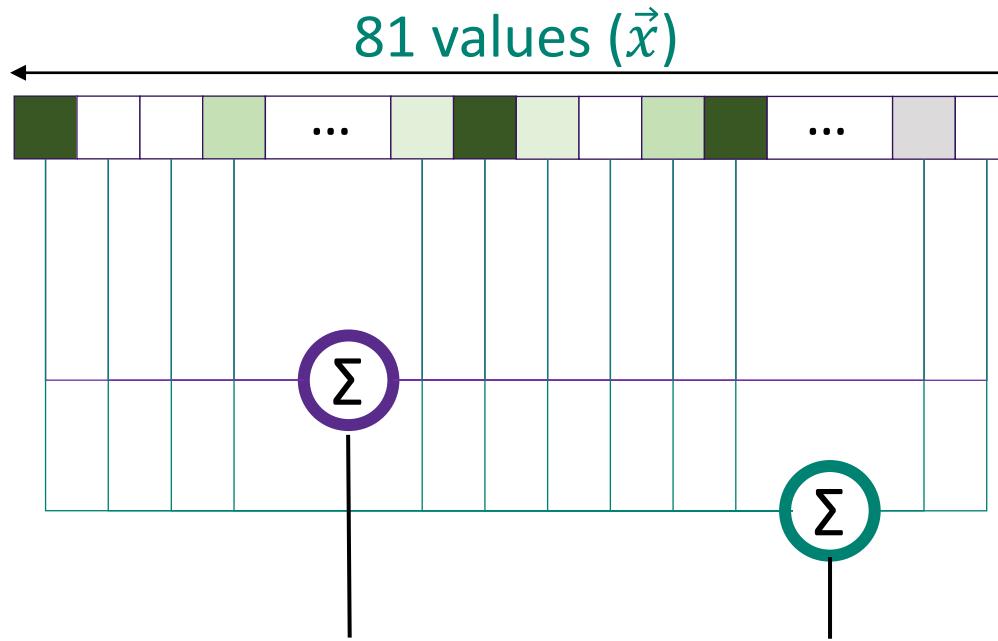
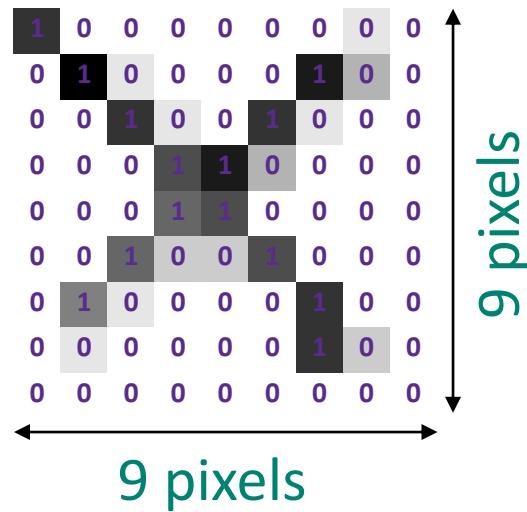


NOW THAT YOU MENTION IT...

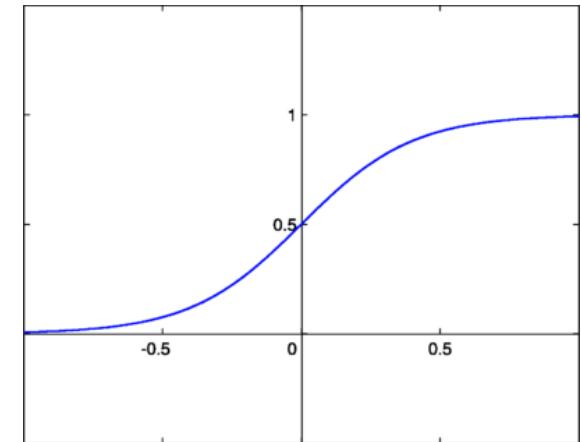
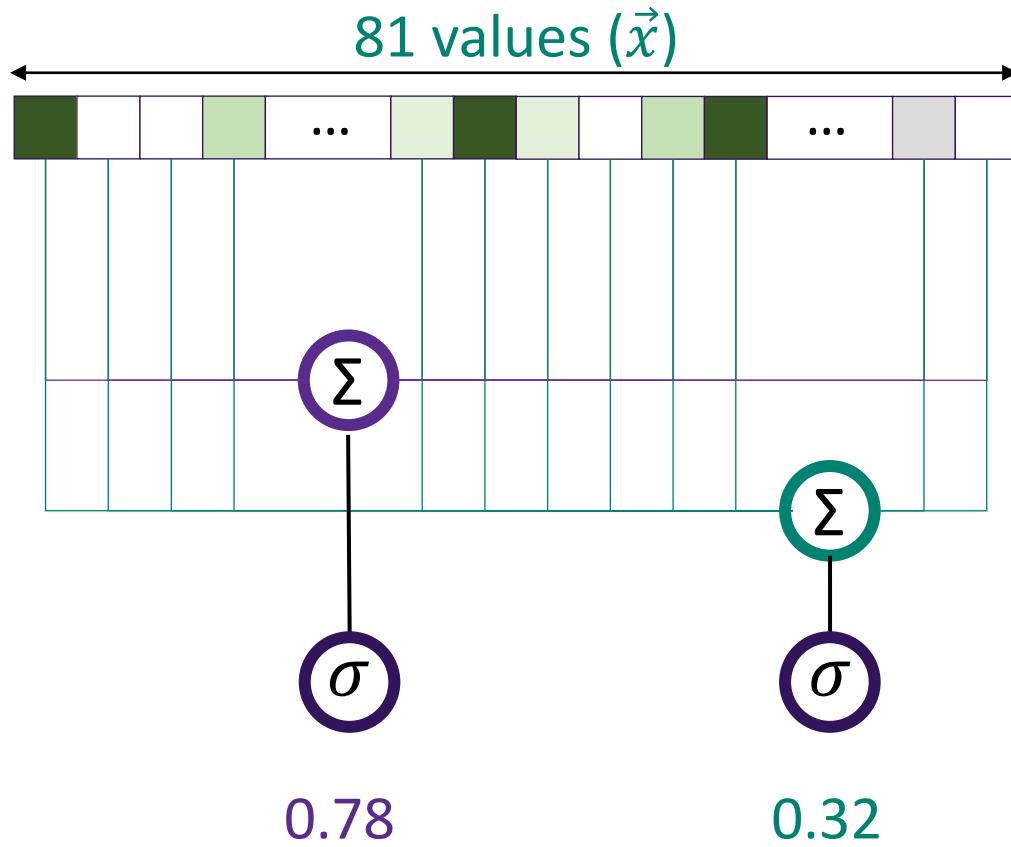
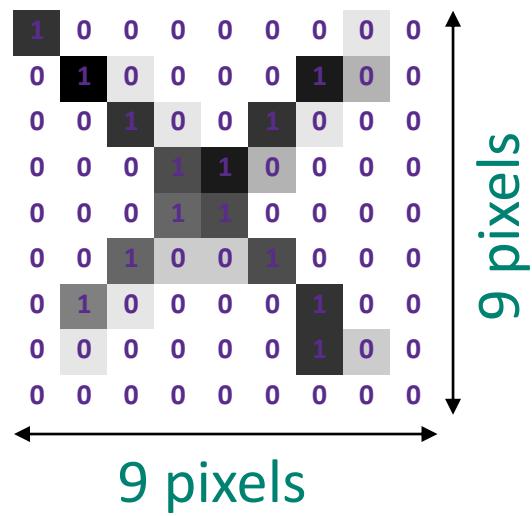
$$z = w \cdot \vec{x} + b$$



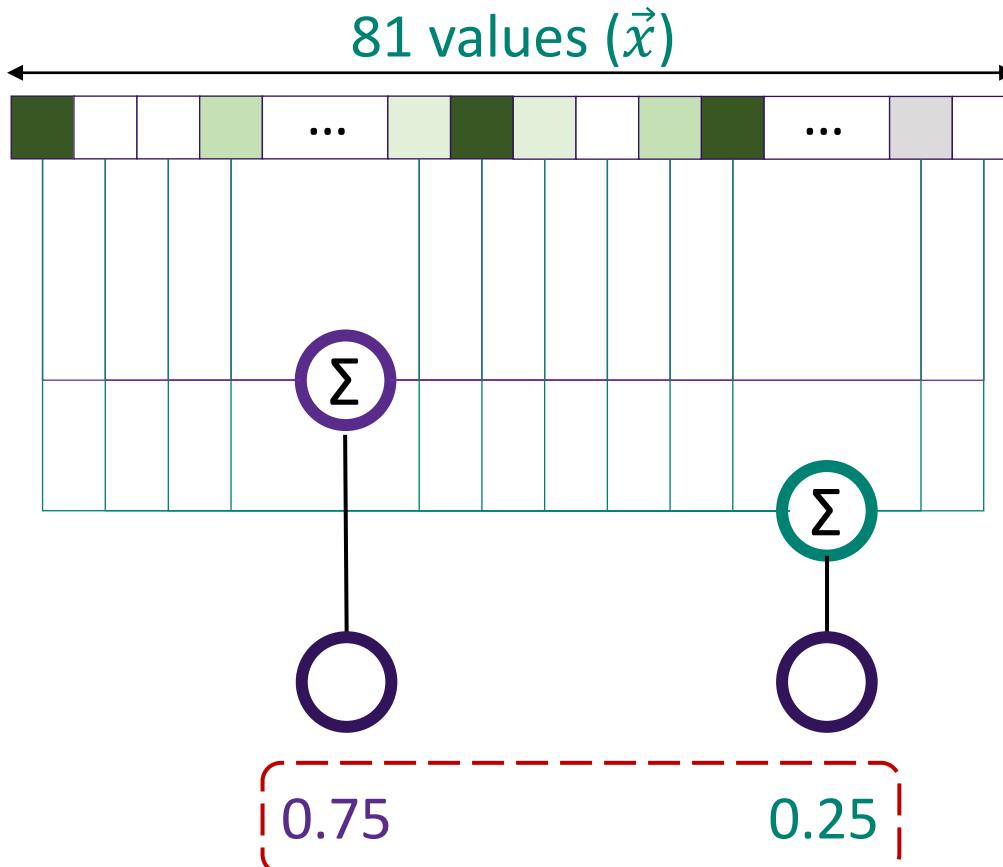
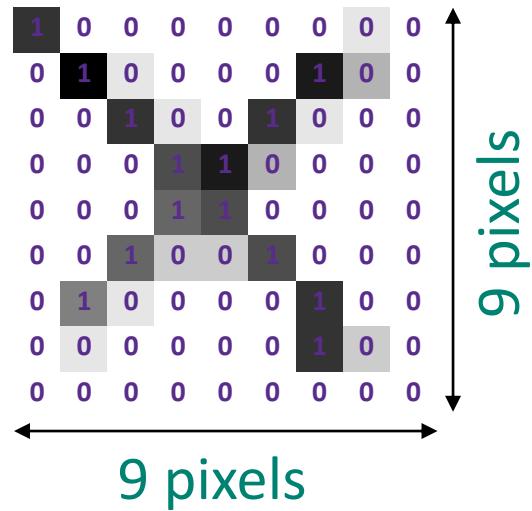
SIGMOID



SIGMOID

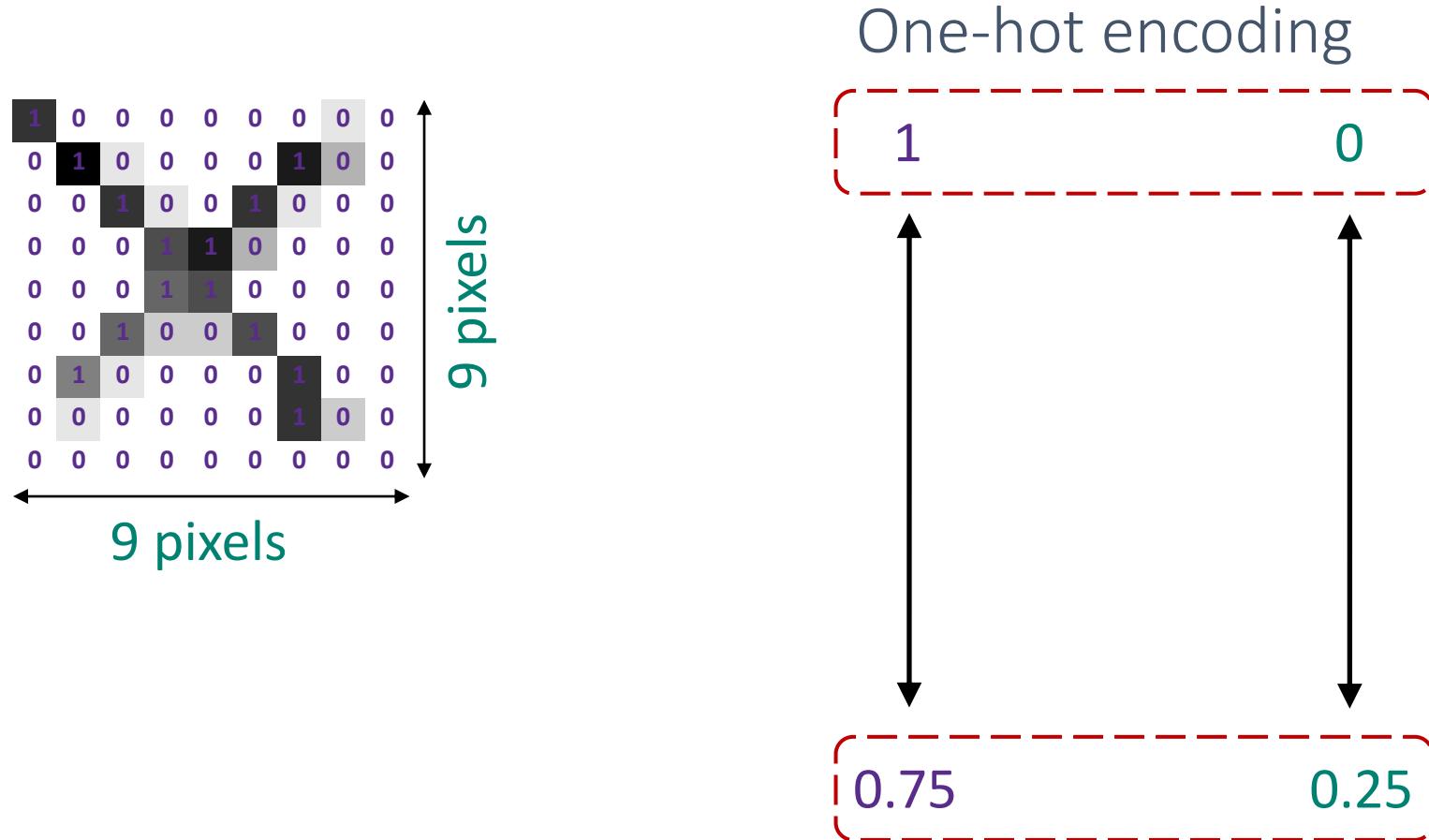


SOFTMAX

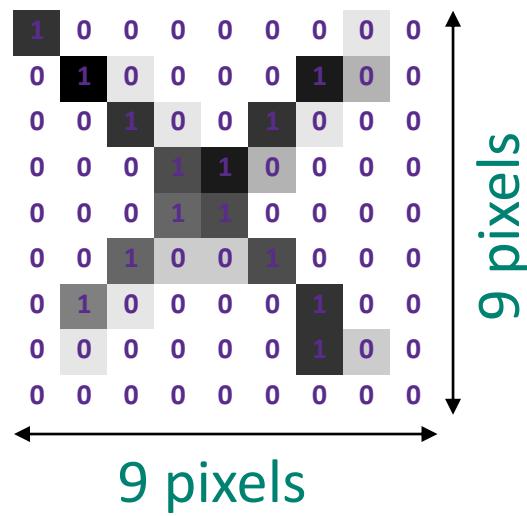


These are actual probabilities

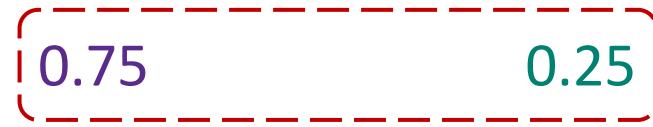
LOSS FUNCTION



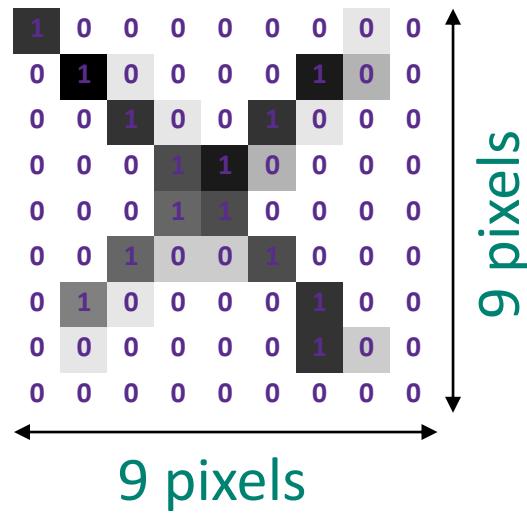
LOSS FUNCTION



$$\text{error} = (y_j - p_j)$$



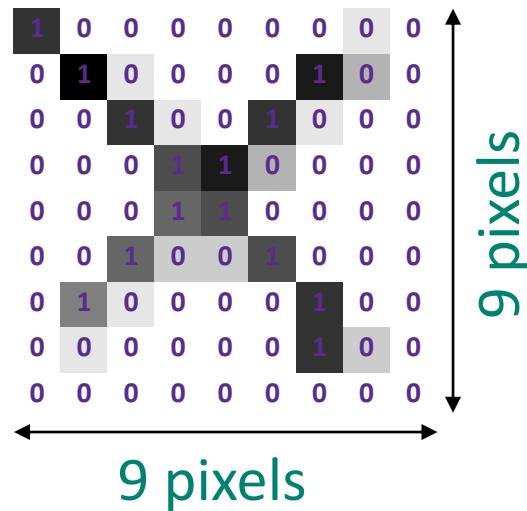
LOSS FUNCTION



$$\text{error} = \frac{1}{n} \sum_{i=1}^n (y_j - p_j)^2$$



LOSS FUNCTION

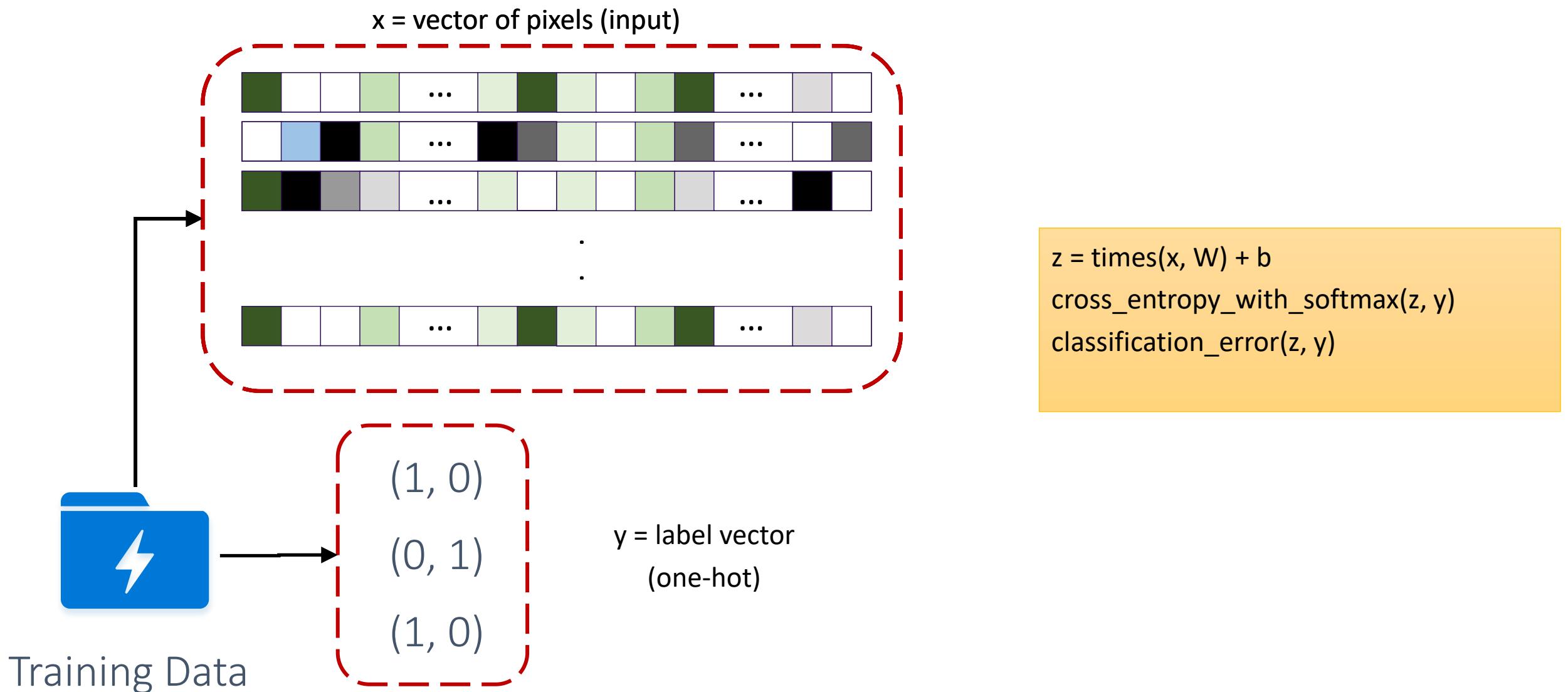


$$\text{Cross Entropy Error} = - \sum_{c=1}^M y_{0,c} \log(p_{0,c})$$

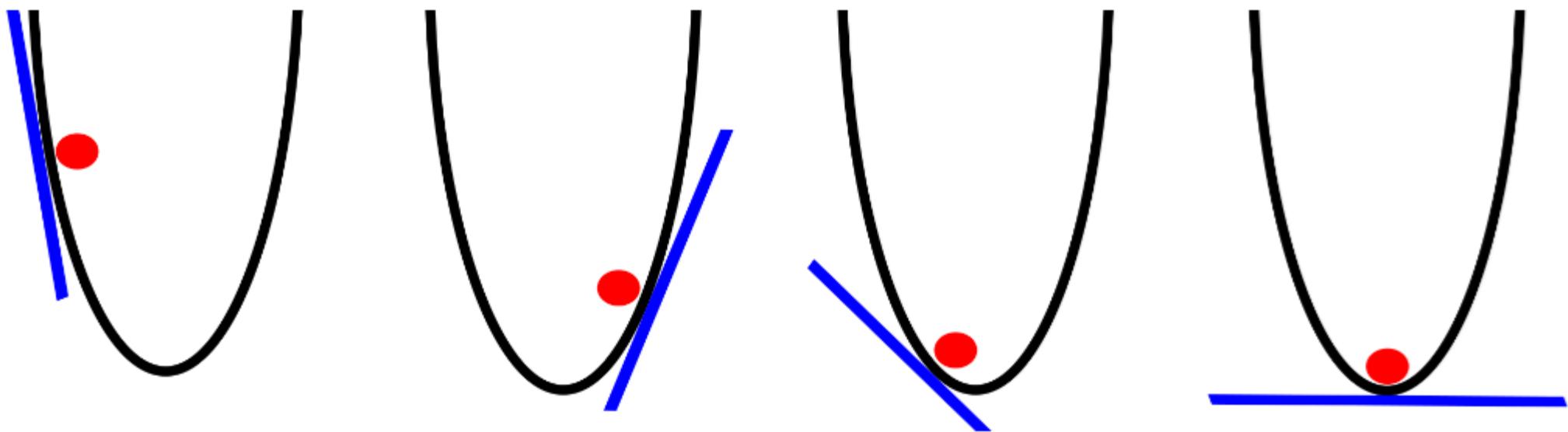
$$\text{Binary: } -(y * \log(p) + (1-y) * \log(1-p))$$



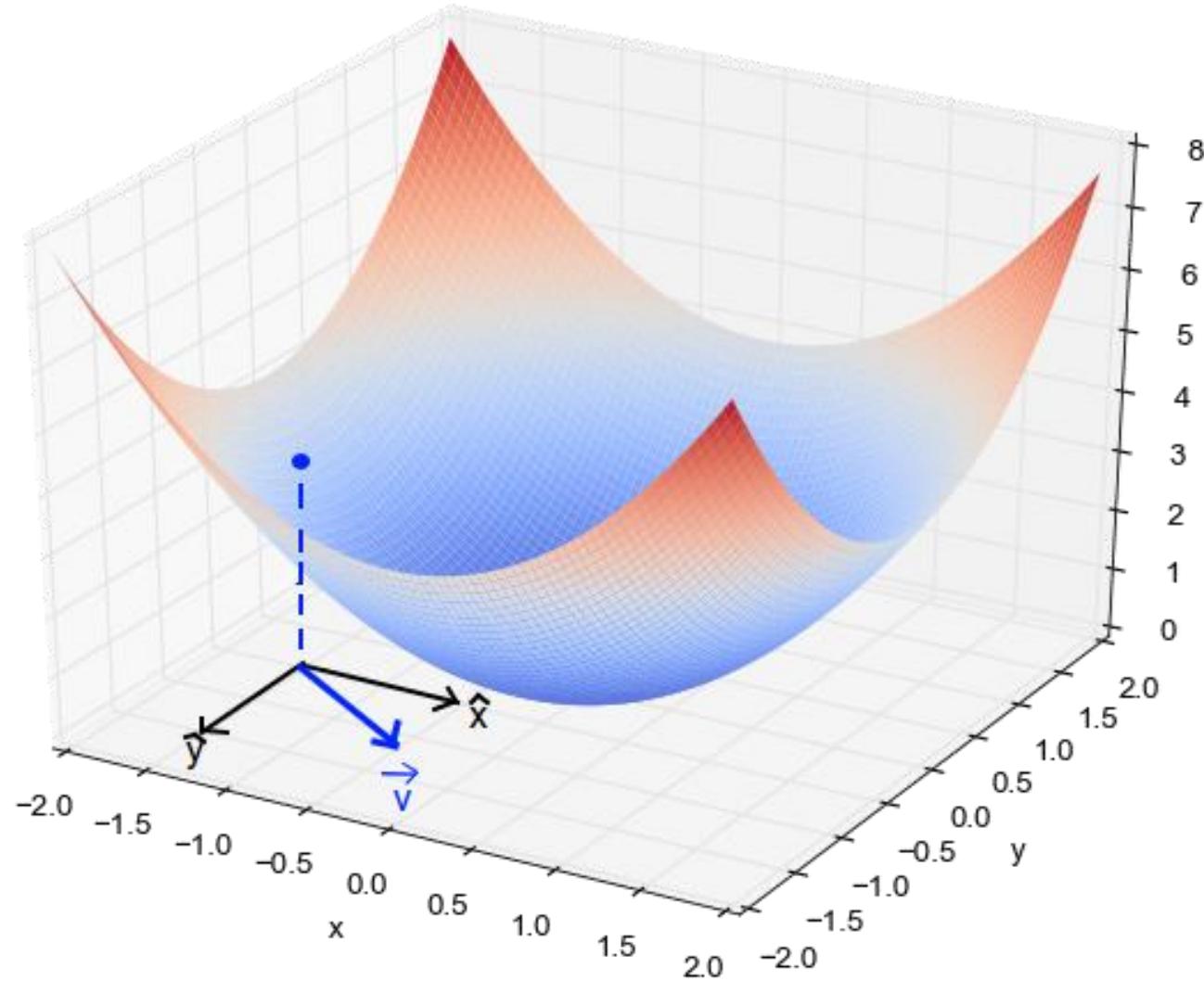
TRAINING



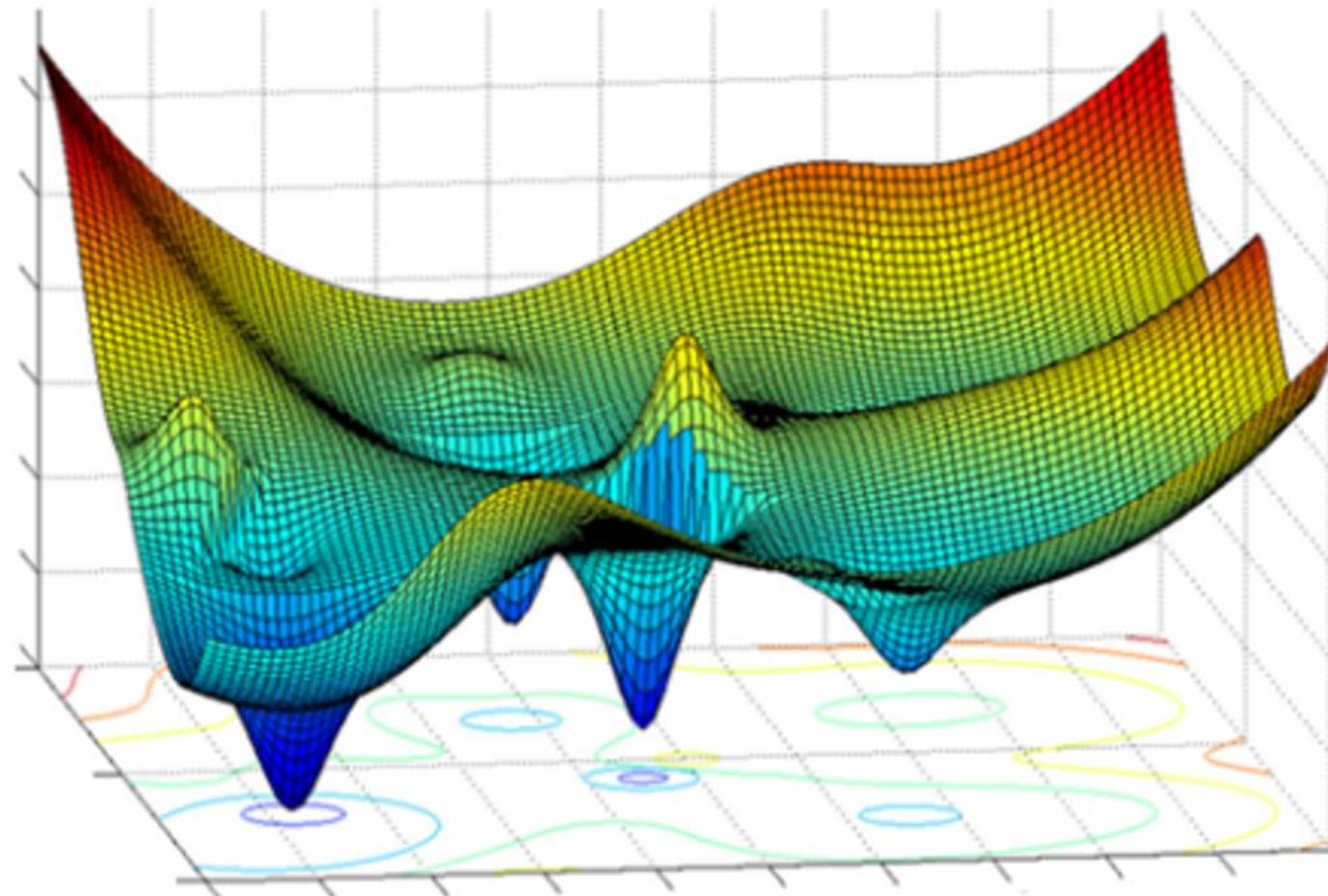
GRADIENT DESCENT

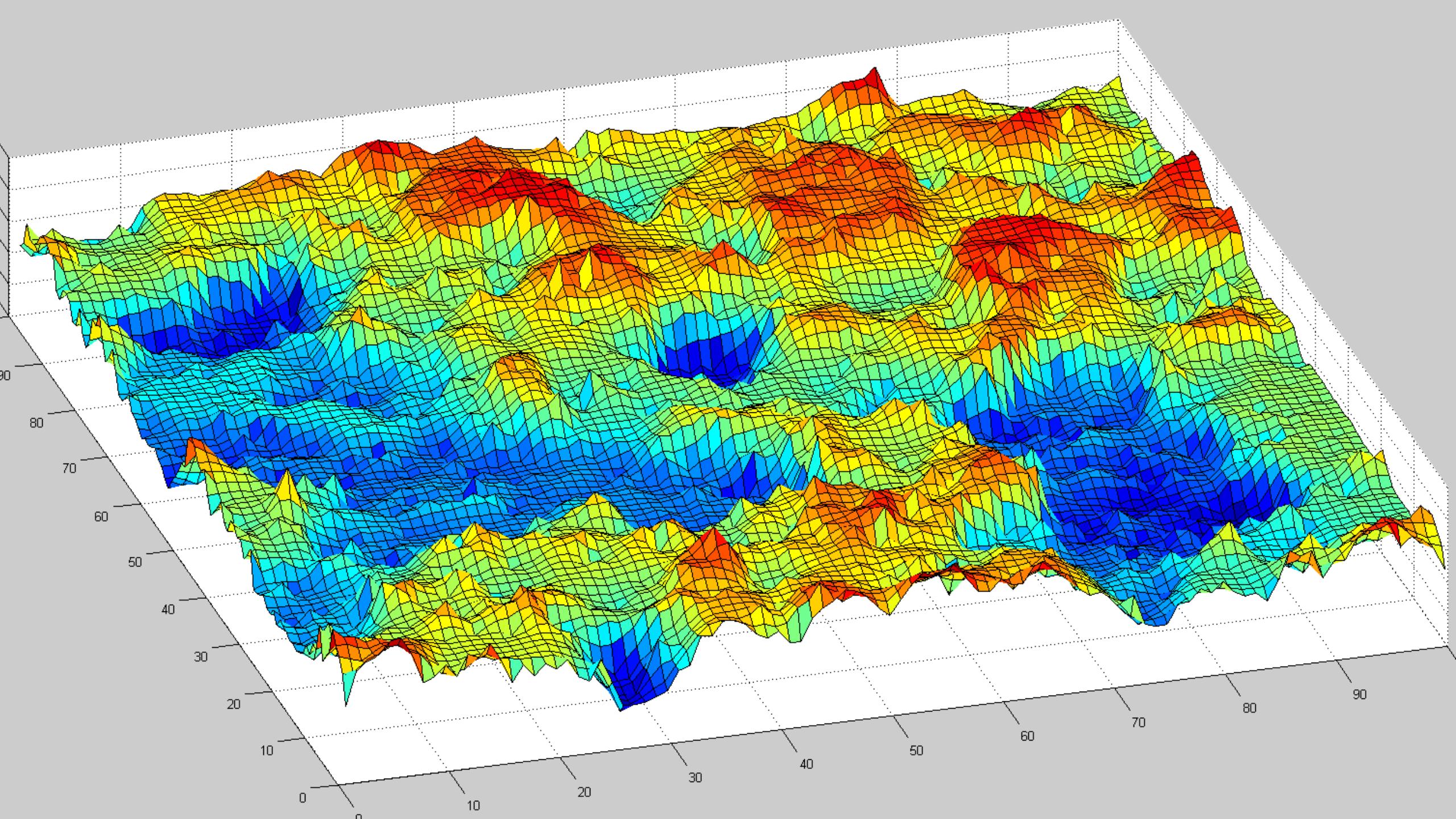


GRADIENT DESCENT



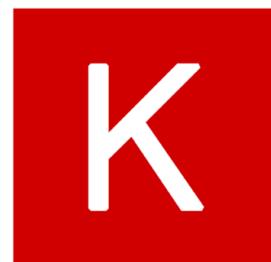
GRADIENT DESCENT





Deep Learning

- Create a Deep Learning Model.
- Use notebook “Lab05_DeepLearning.ipynb”
- Try to create a Convolutional Neural Network to classify image per category.
- <https://www.tensorflow.org/>
- Use Azure Machine learning Workspace to train it and record metrics.
- Save the model and register it.

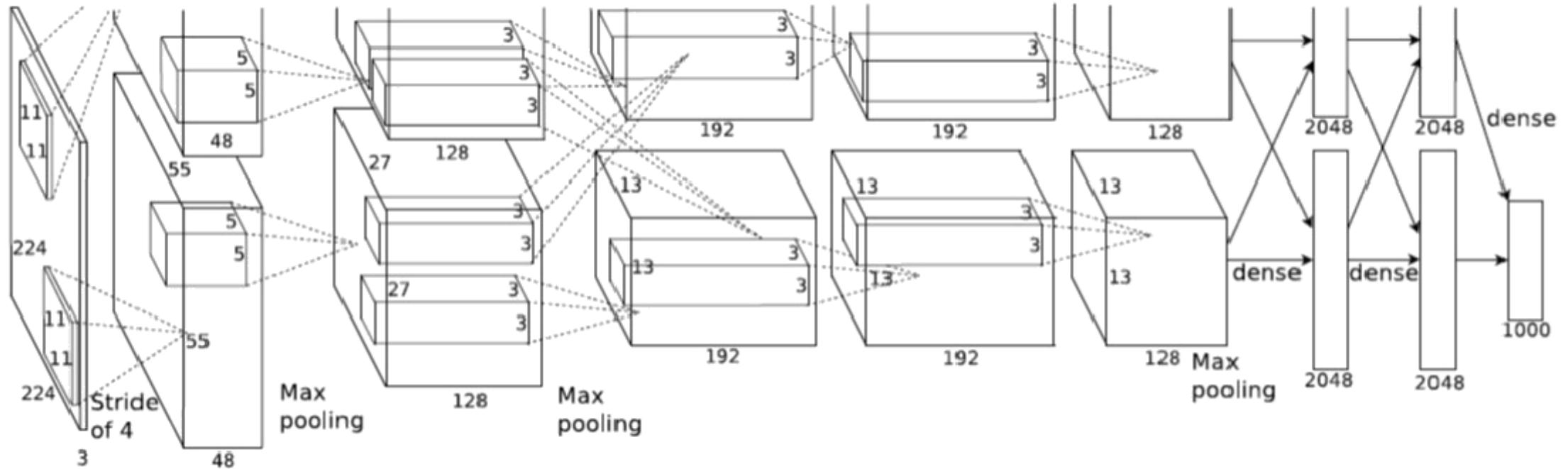


Keras

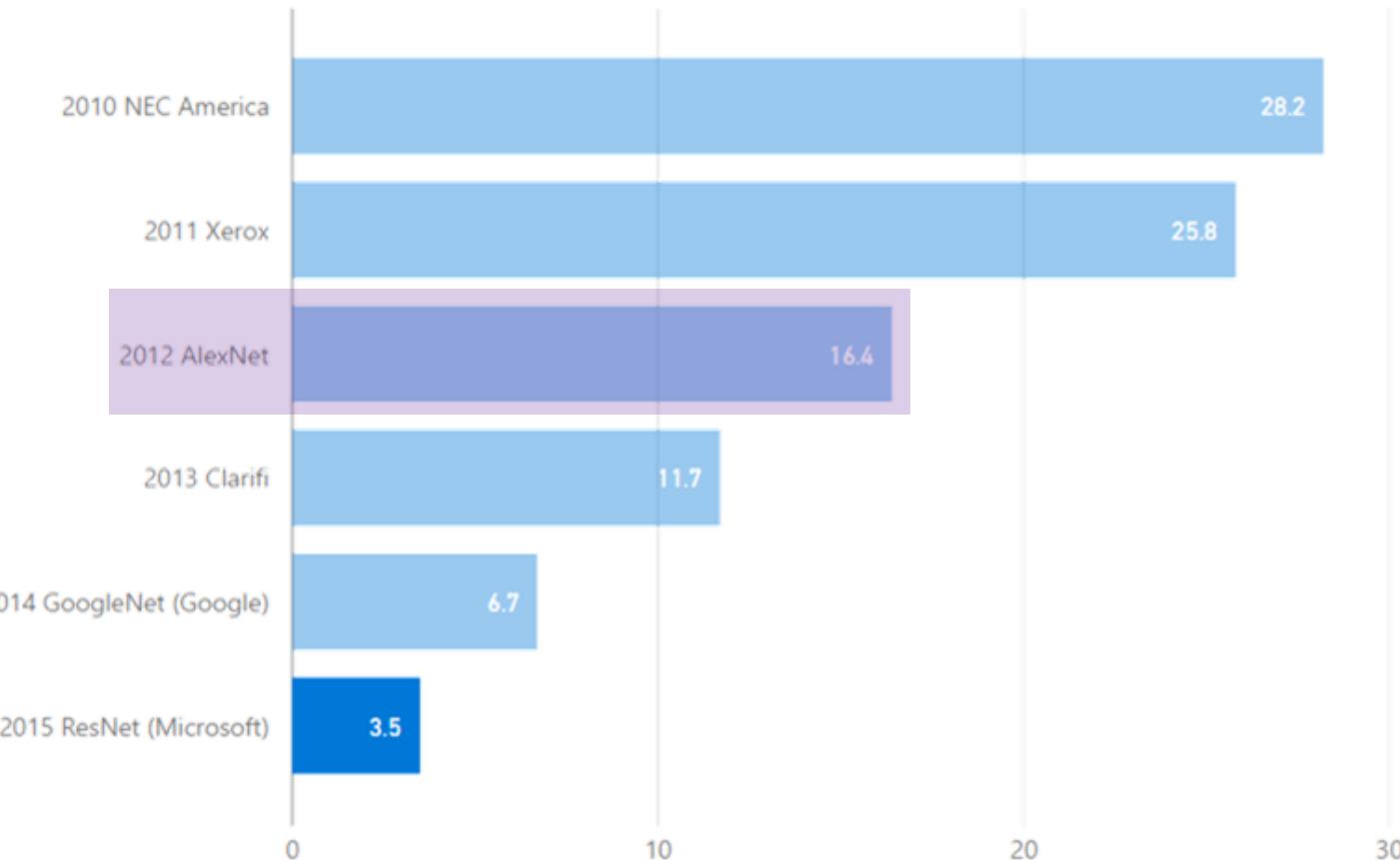


Convolutional Networks

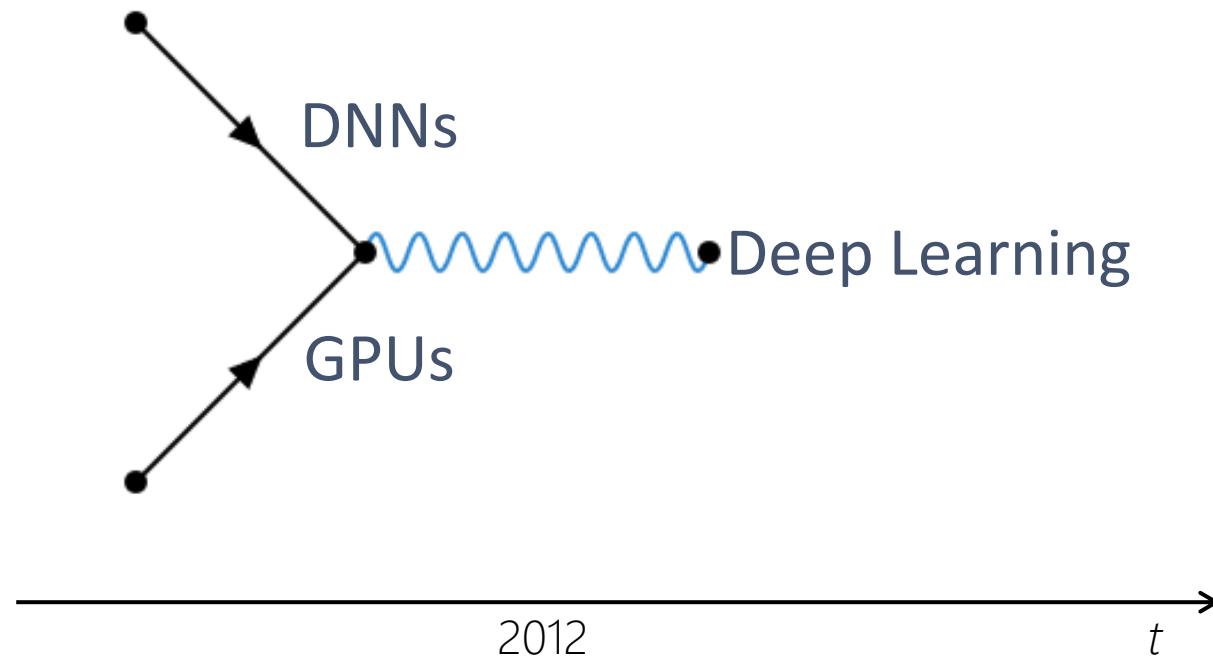
REMEMBER THIS THING?



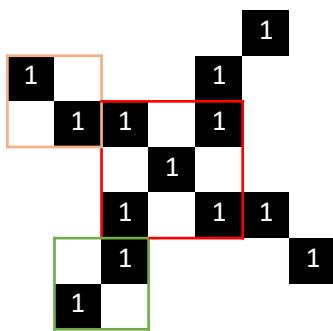
IMAGENET CHALLENGE (CLASSIFICATION)



WHAT HAPPENED IN 2012?



FILTERING



FILTERING

...looking for common characteristics

- Filter 1 Filter 2 Filter 3

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

1	-1	1
-1	1	-1
1	-1	1

1	1	1
1	1	1
1	1	1

-1	1	-1
1	1	1
-1	1	-1

1	1	-1
1	1	1
-1	1	1

1.0

0.11

0.55

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

• Filter

1	-1	-1
-1	1	-1
-1	-1	1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

CONVOLUTION

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1



1	-1	-1
-1	1	-1
-1	-1	1

=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

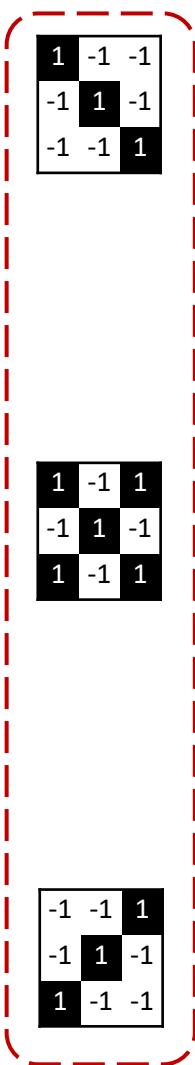


-1	-1	1
-1	1	-1
1	-1	-1

=

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	1	-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	1	-1



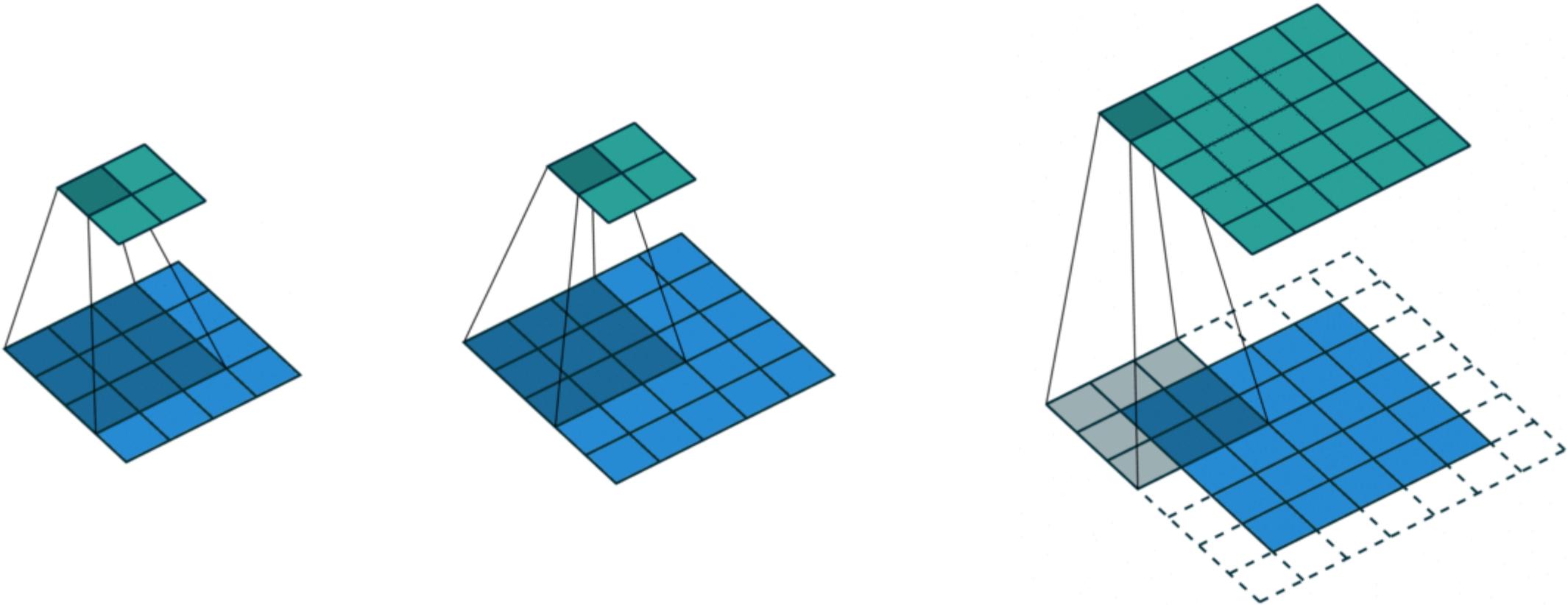
=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

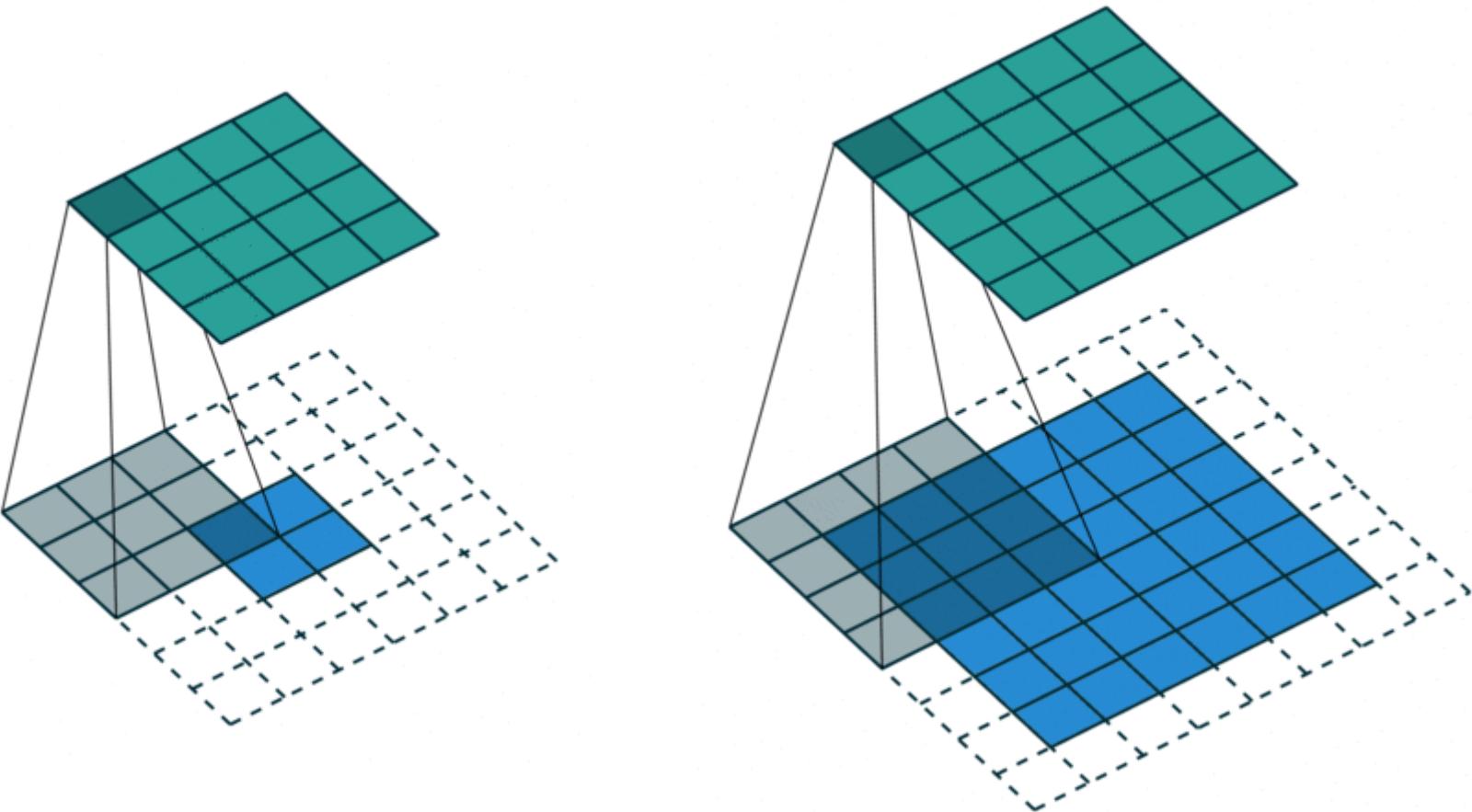
0.33	-0.55	-0.11	-0.11	0.11	-0.55	0.33
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.11	0.33	-0.77	1.00	-0.77	0.33	-0.11
0.11	-0.55	0.55	-0.77	0.55	-0.55	0.11
-0.55	0.55	-0.55	0.33	-0.55	0.55	-0.55
0.33	-0.55	0.11	-0.11	0.11	-0.55	0.33

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

STRIDES AND PADDING



DECONVOLUTION – TRANSPOSED CONVOLUTION



source: https://github.com/vdumoulin/conv_arithmetic

POOLING

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

POOLING

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

AVERAGE POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.39	0.22	0.17	0.28	0.11	0.06
0.22	0.45	0.22	0.00	0.00	0.11
0.17	0.22	0.22	0.00	0.00	0.28
0.28	0.00	0.00	0.22	0.22	0.17
0.11	0.00	0.00	0.22	0.45	0.22
0.06	0.11	0.28	0.17	0.22	0.39

MAX POOLING – STRIDE 1

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	1.00	0.33	0.55	0.55	0.33
1.00	1.00	1.00	0.33	0.11	0.55
0.33	1.00	1.00	0.55	0.33	0.55
0.55	0.33	0.55	1.00	1.00	0.33
0.55	0.11	0.33	1.00	1.00	1.00
0.33	0.55	0.55	0.33	1.00	1.00

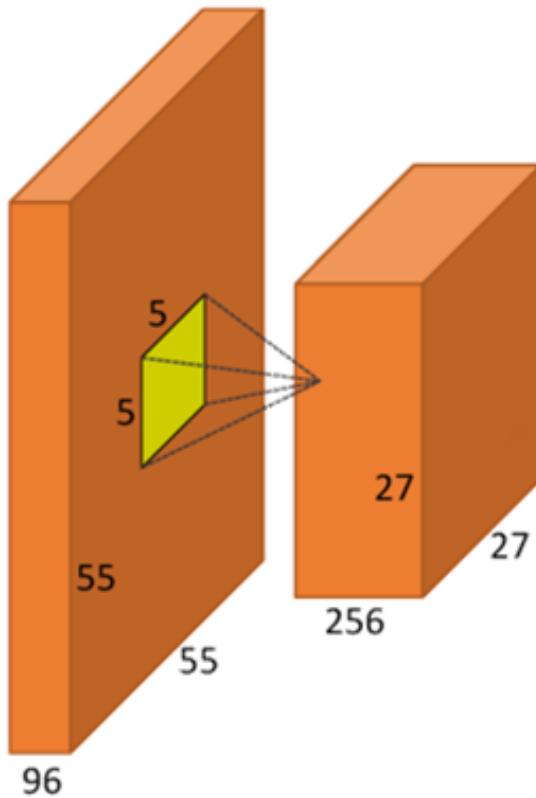
AVERAGE POOLING – STRIDE 2

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

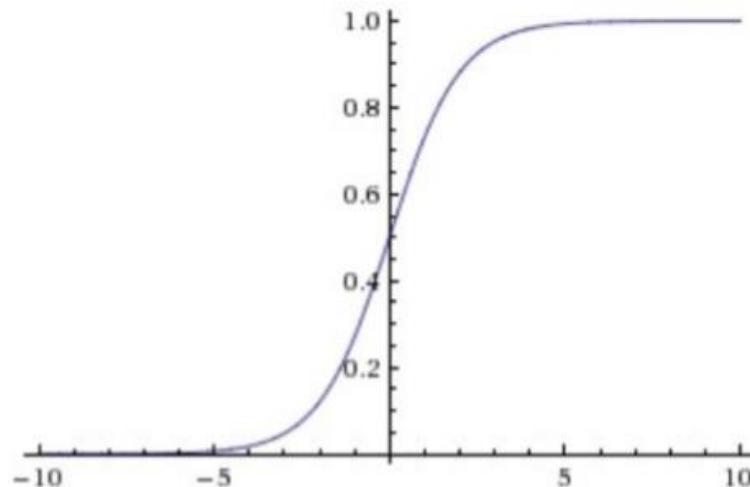
WHY POOLING?

ACTIVATION FUNCTIONS?



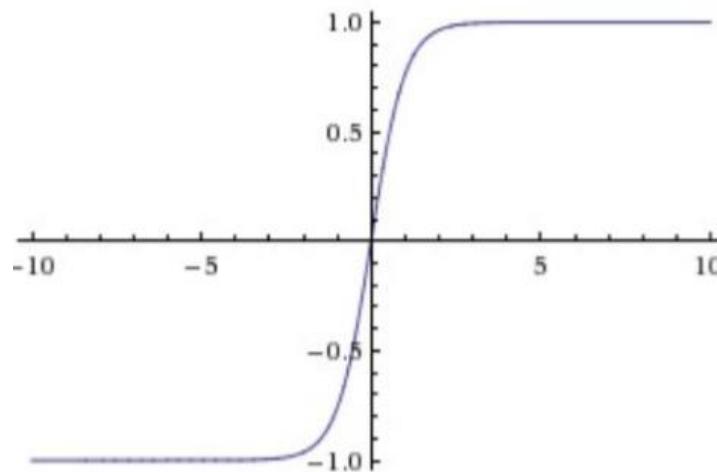
ACTIVATION FUNCTIONS?

LOGISTIC FUNCTION (“*sigmoid*”):



- Simple
- Drawbacks:
 - Saturates
 - Non-centered in zero
- We don't use it anymore, but we thank her for the services provided to the cause :)

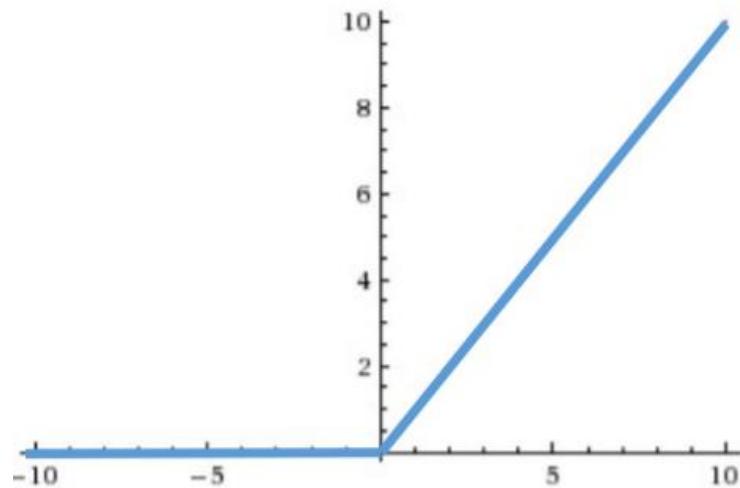
ACTIVATION FUNCTIONS?



HYPERBOLIC TANGENT (\tanh):

- Solves the issue with not being centered in zero
- Still ‘killing gradients’ due to saturating behaviour
- It is always better than the logistic function

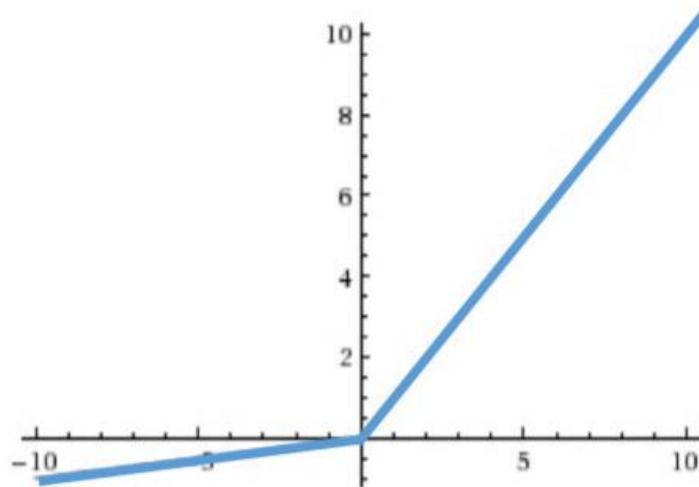
ACTIVATION FUNCTIONS?



RECTIFIED LINEAR UNITS (ReLU):

- Optimization for the SGD convergence
- More simple from a computational point of view
- Fragile

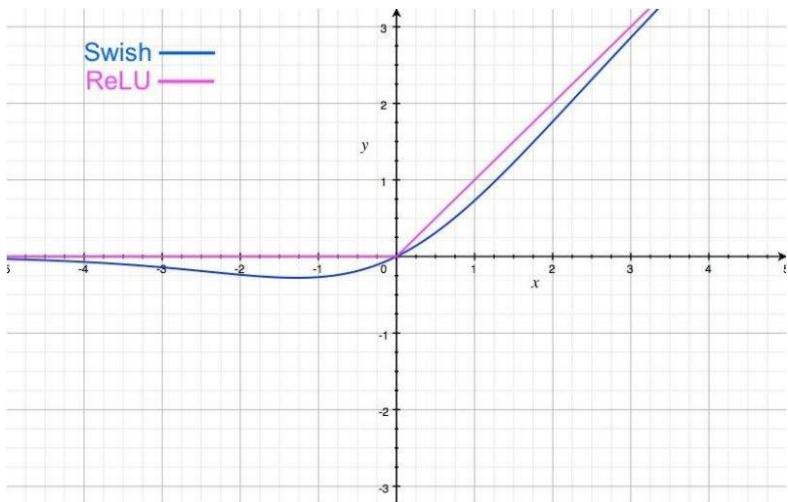
ACTIVATION FUNCTIONS?



LEAKY ReLU:

- More stable than traditional ReLU
- Particular case of MaxOut

ACTIVATION FUNCTIONS?



SWISH:

- Self-Gating activation function
- Better performance than ReLU

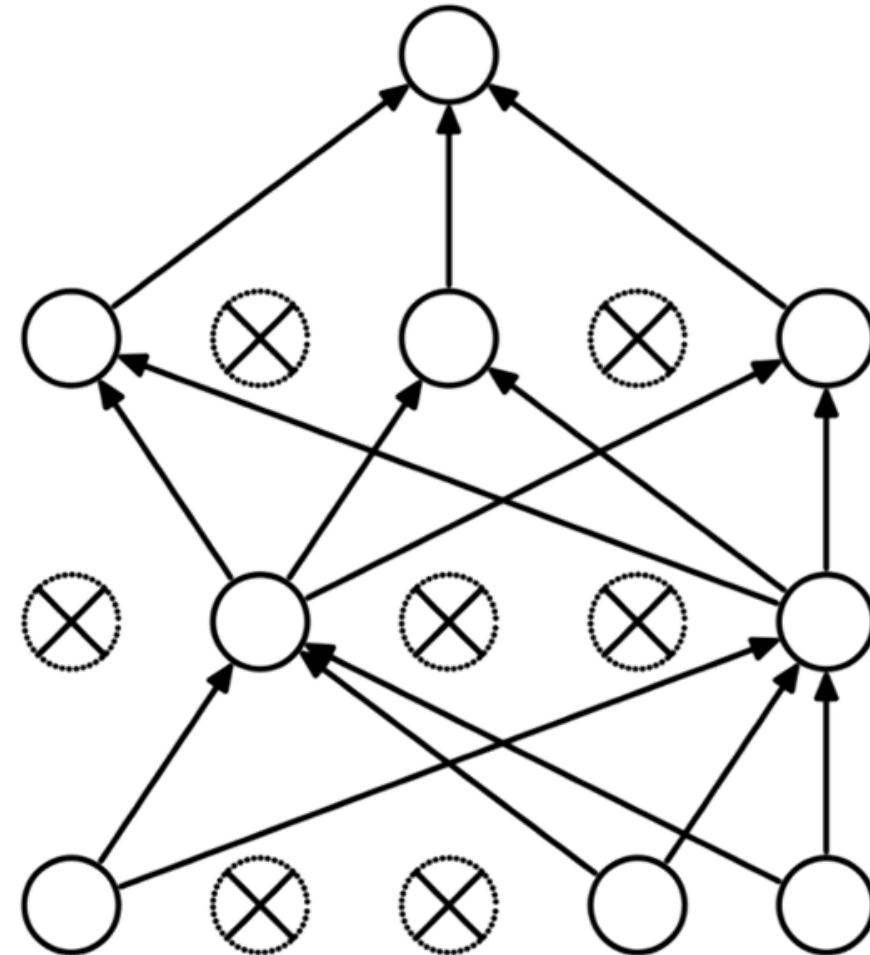
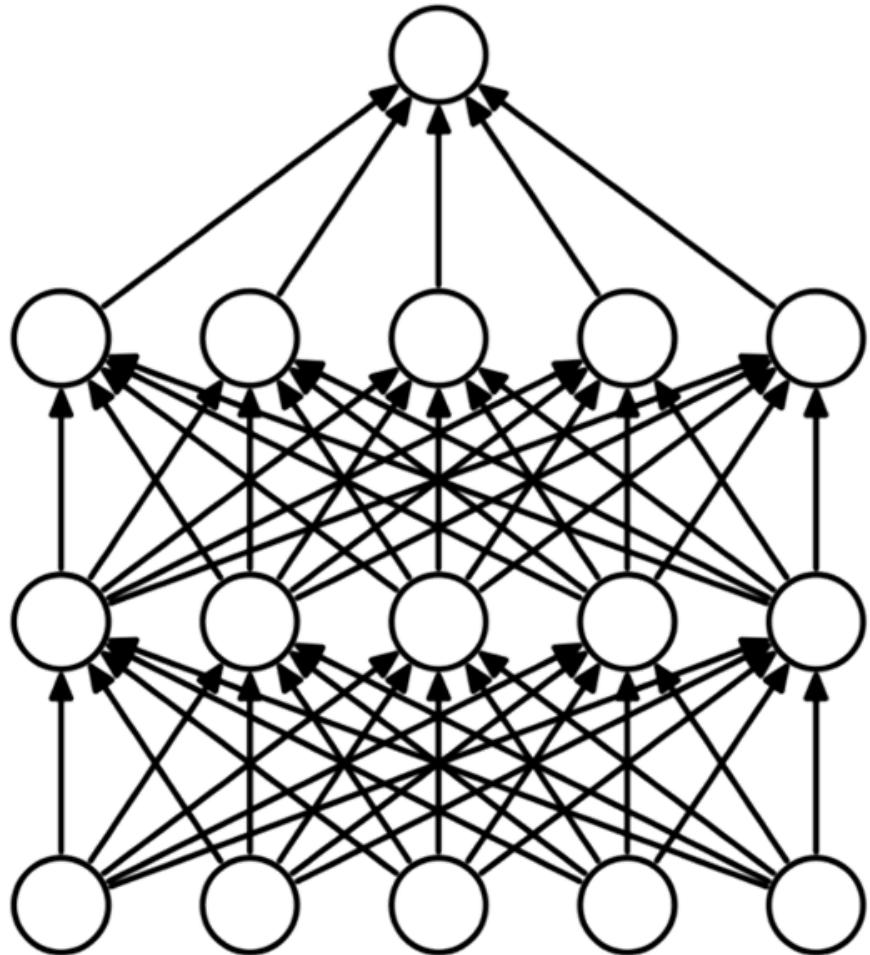
RECTIFIED LINEAR UNITS (ReLU)

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

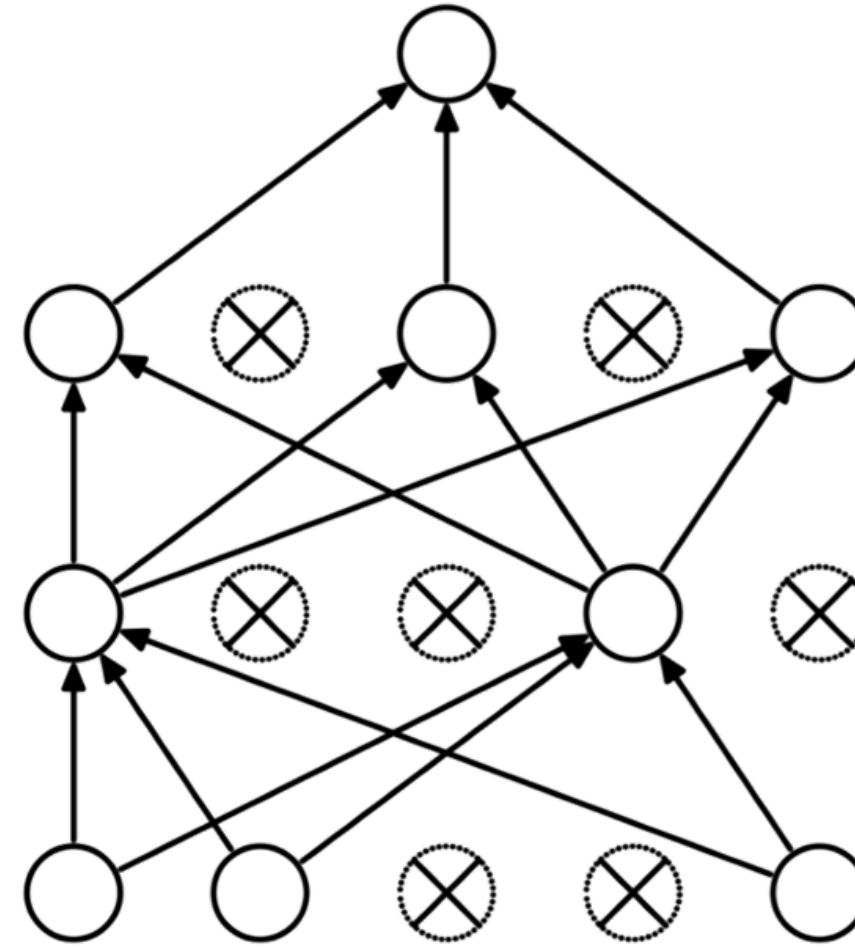
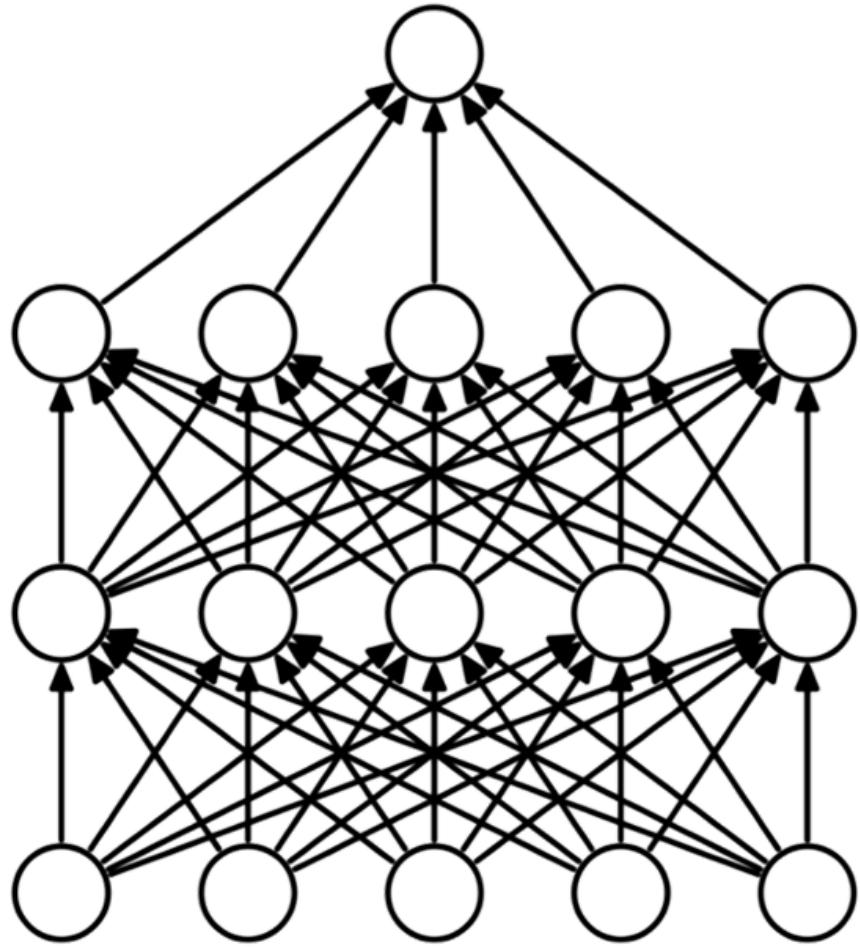


0.77	0.00	0.11	0.33	0.55	0.00	0.33
0.00	1.00	0.00	0.33	0.00	0.11	0.00
0.11	0.00	1.00	0.00	0.11	0.00	0.55
0.33	0.33	0.00	0.55	0.00	0.33	0.33
0.55	0.00	0.11	0.00	1.00	0.00	0.11
0.00	0.11	0.00	0.33	0.00	1.00	0.00
0.33	0.00	0.55	0.33	0.11	0.00	0.77

DROPOUT



DROPOUT



WHAT IS MISSING?

- Filter 1 Filter 2 Filter 3

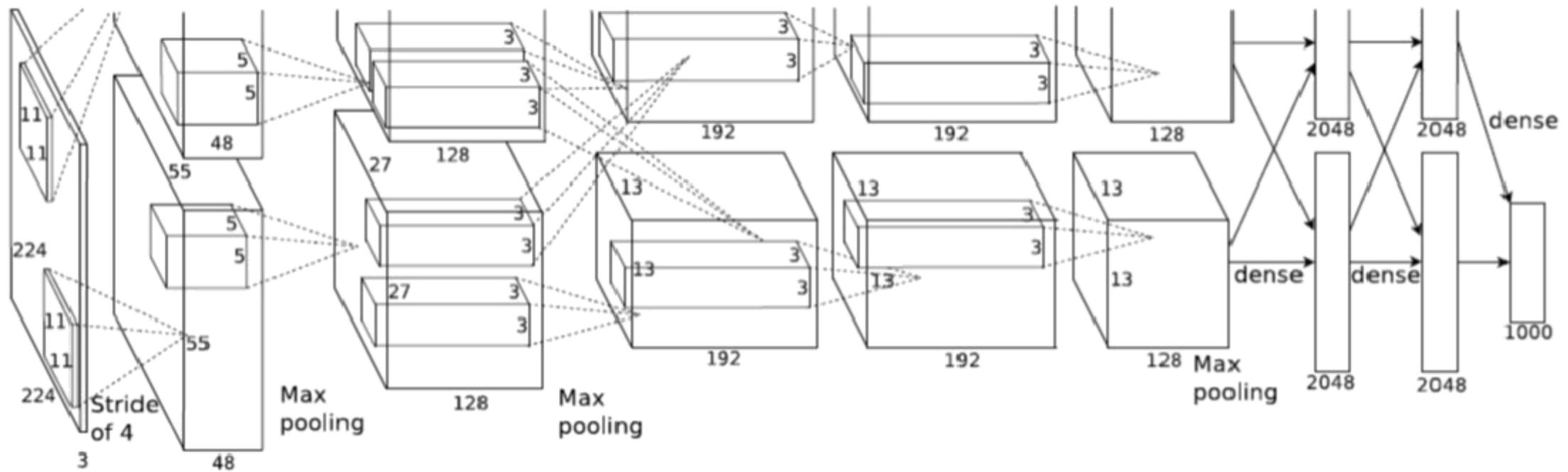
$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & \boxed{1} & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & \boxed{1} & -1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & \boxed{1} & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

How have these filters been
selected/created?

AlexNet



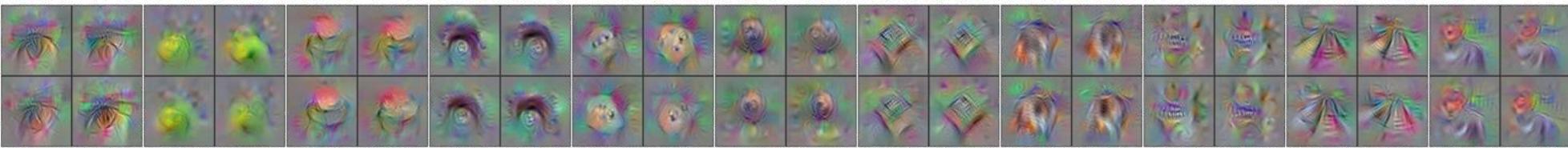
HOW A CNN SEES

Layer 2



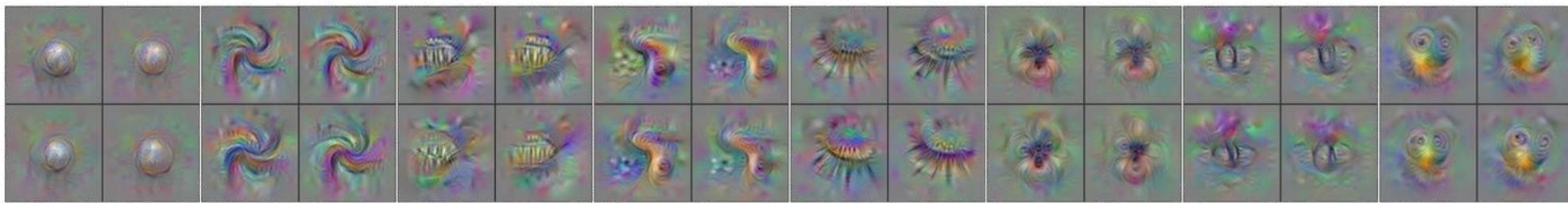
HOW A CNN SEES

Layer 3



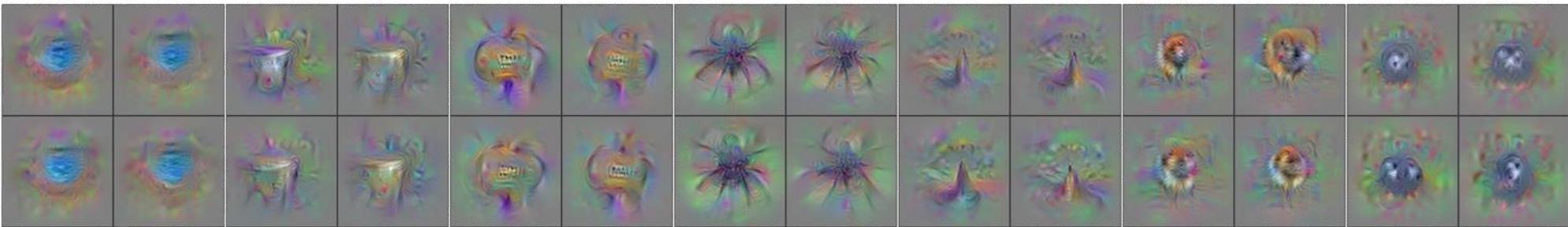
HOW A CNN SEES

Layer 4



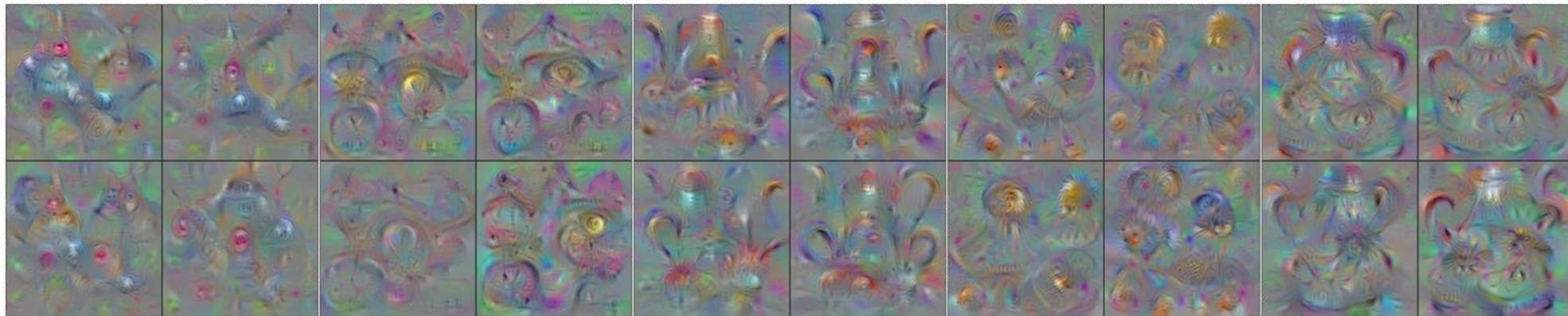
HOW A CNN SEES

Layer 5



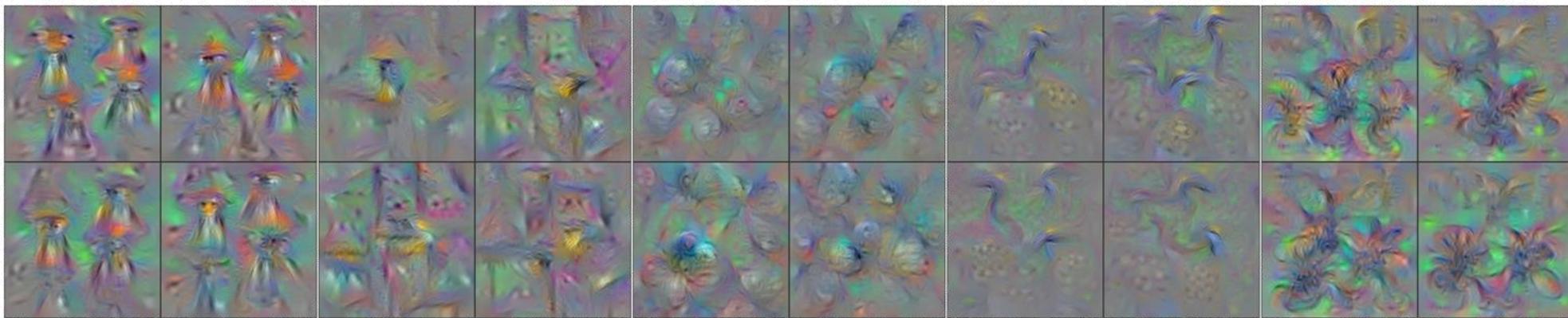
HOW A CNN SEES

Layer 6



HOW A CNN SEES

Layer 7



HOW A CNN SEES

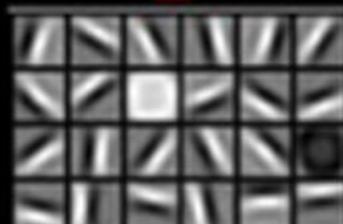
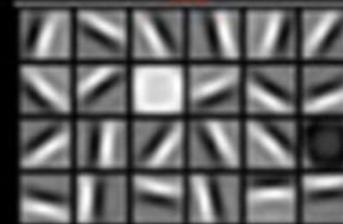
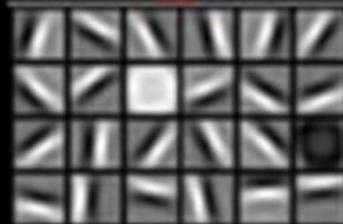
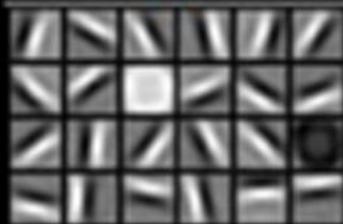
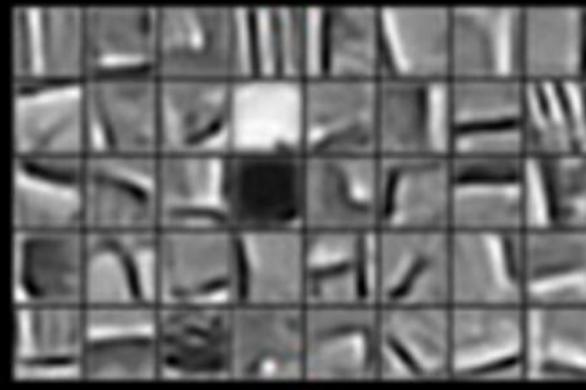
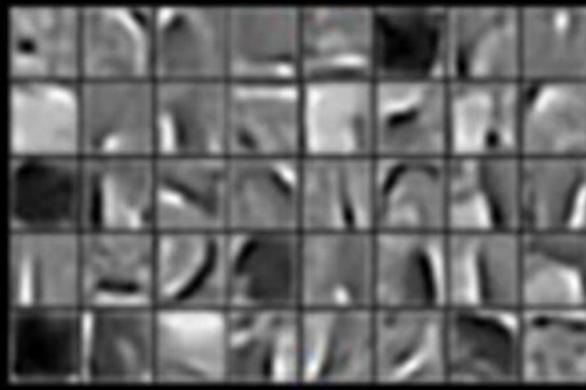
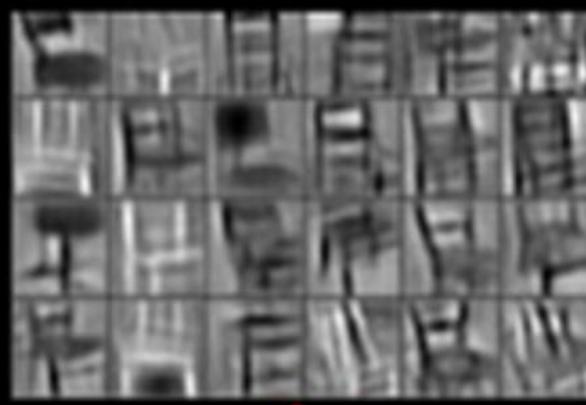


Faces

Cars

Elephants

Chairs





LAB 4

Deploy



Deploy

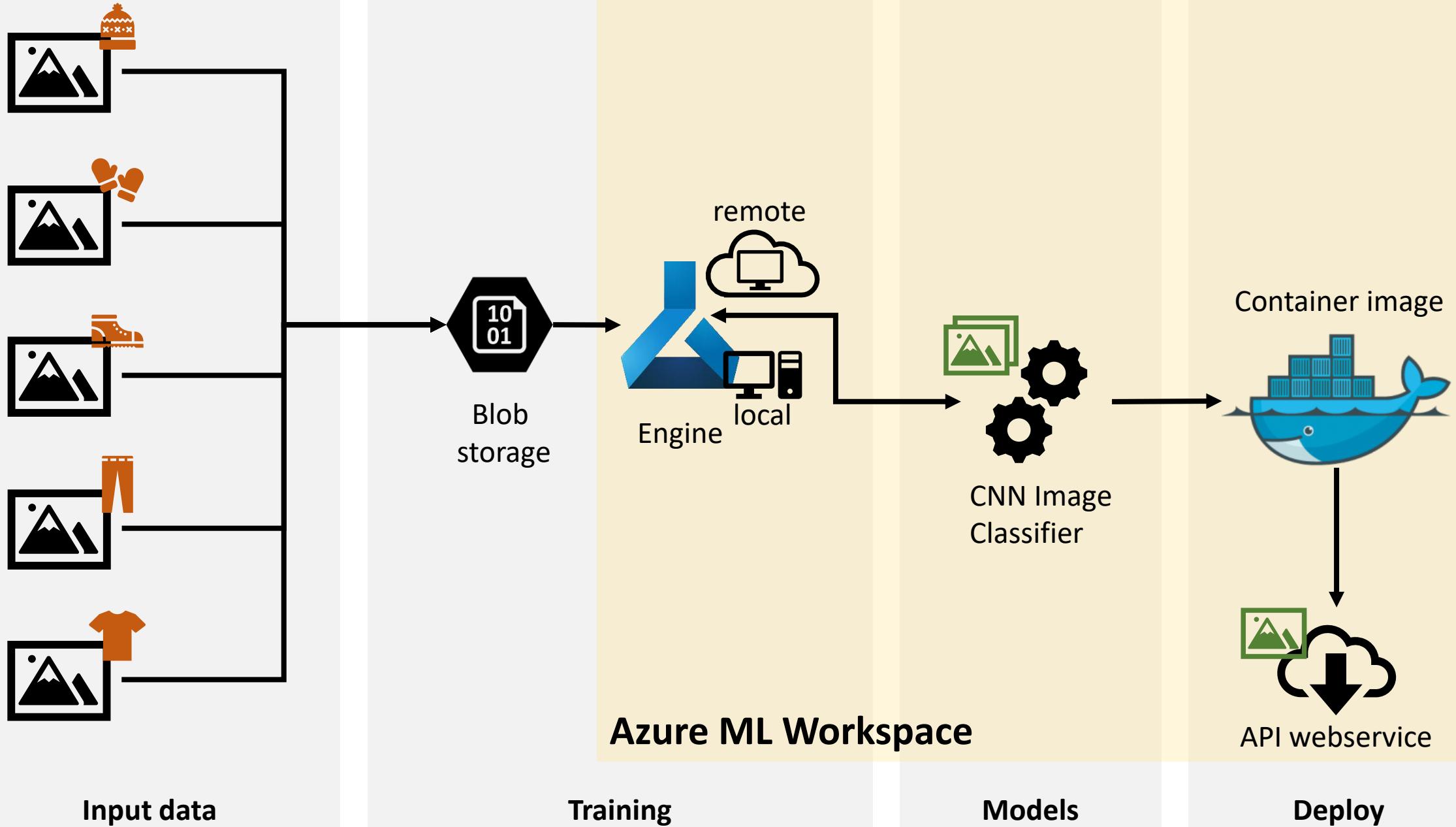
Use notebook “*Lab06_deploy.ipynb*”

Create a docker image

- Install dependencies (conda & pip)
- Score.py
 - Init() → Download the model
 - Run(raw_data) → Json request in the service

Create the webservice to deploy the image

Test the webservice



Some Advanced Ideas

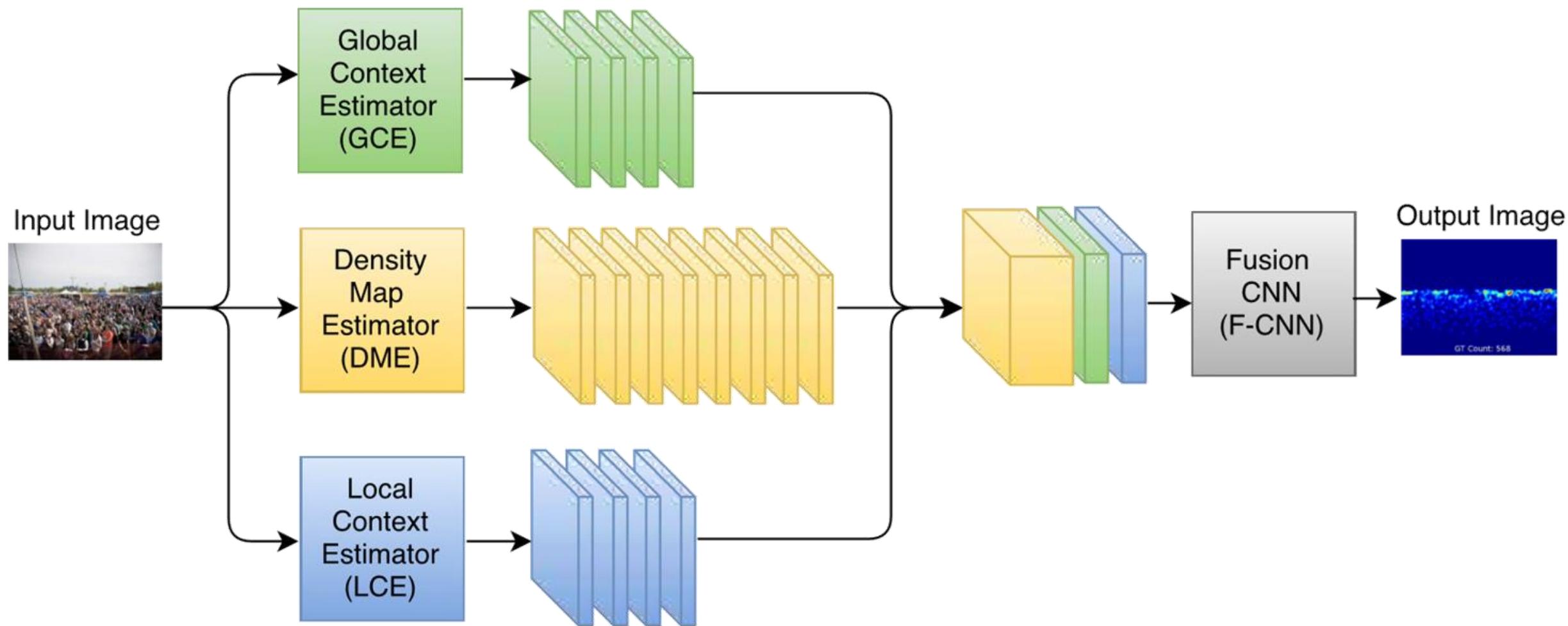
Counting people in a multitude

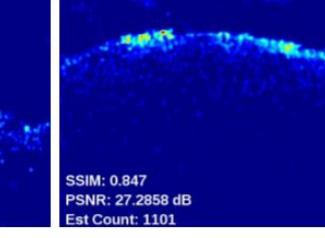
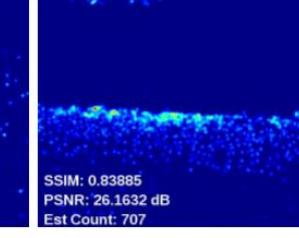
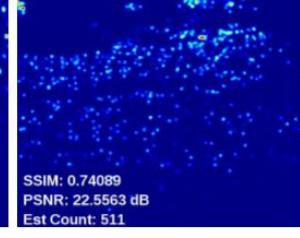
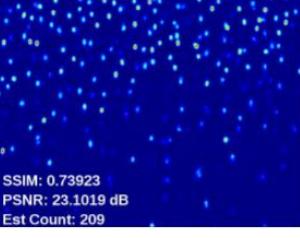
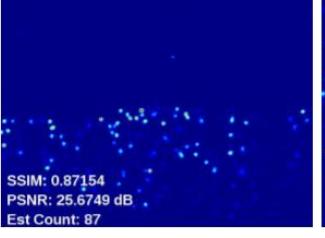
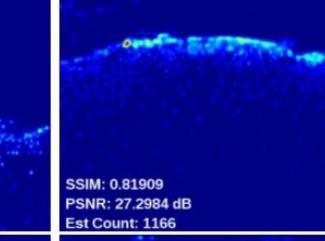
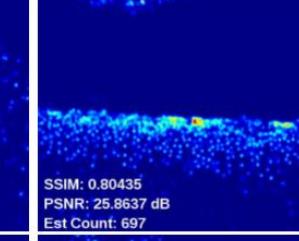
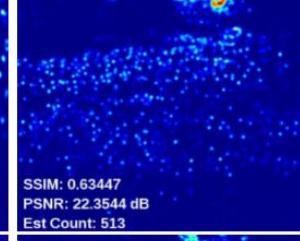
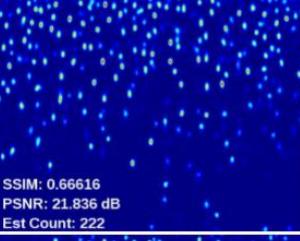
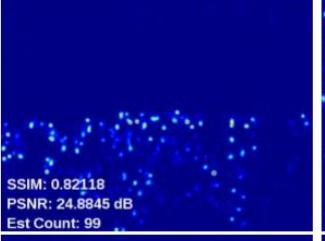
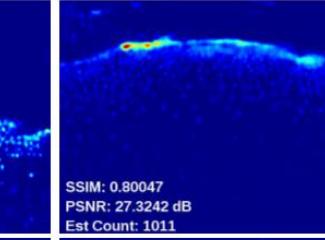
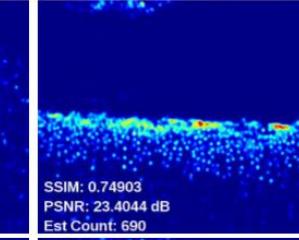
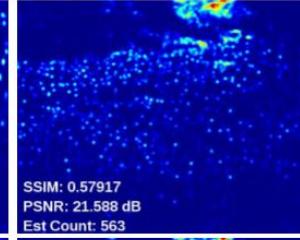
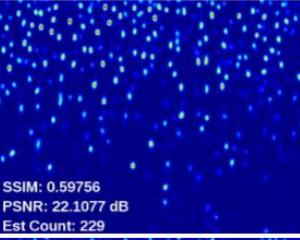
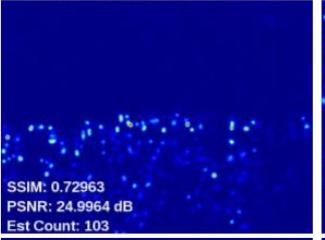
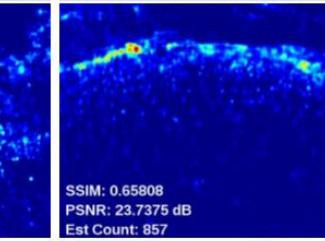
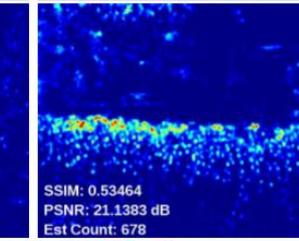
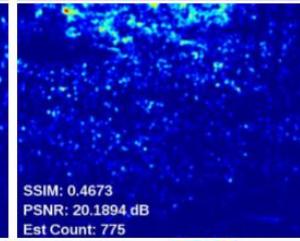
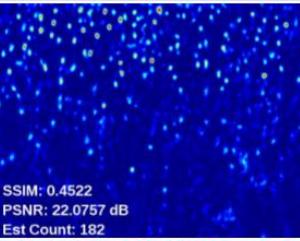
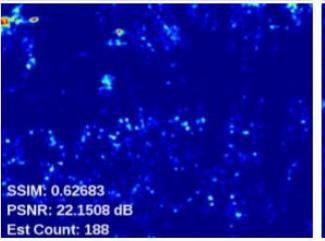
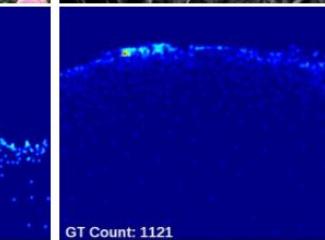
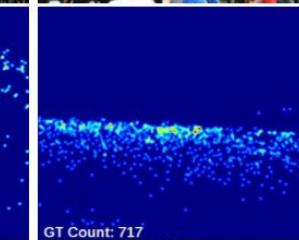
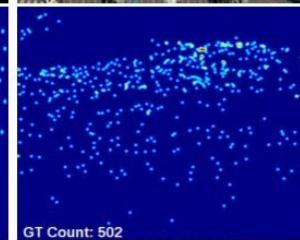
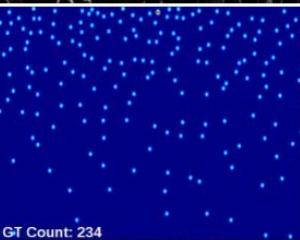
- CONTEXTUAL PYRAMID CNN (CP-CNN)

A new method named Contextual Pyramid CNN (CP-CNN) is proposed here to generate density maps and influx estimations, by explicitly incorporating global and local context information. Composed of four modules: Global Context Estimator (GCE), Local Context Estimator (LCE), Density Map Estimator (DME) and a Fusion-CNN (F-CNN) convolutional network.

Vishwanath A. Sindagi, Vishal M. Patel; The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1861-1870









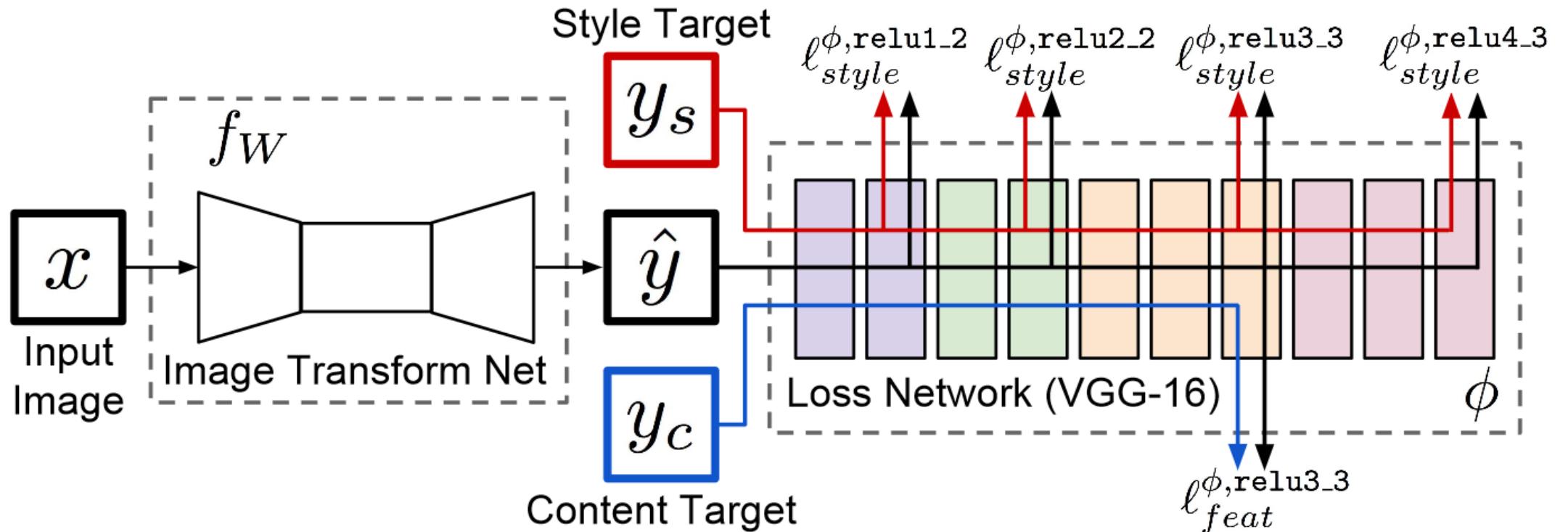
Transferring style between images

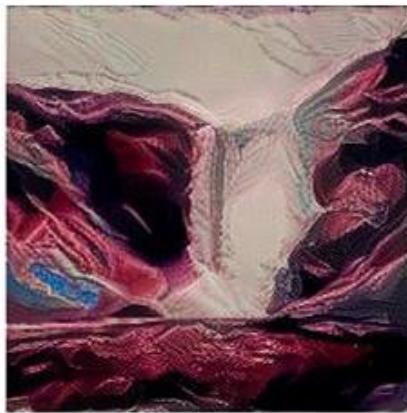
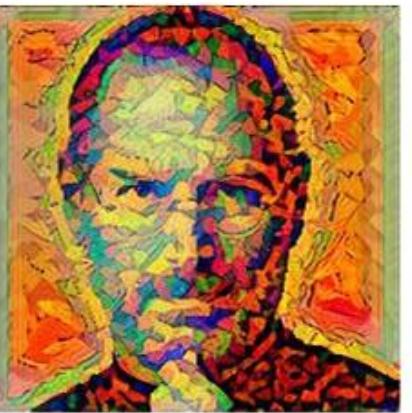
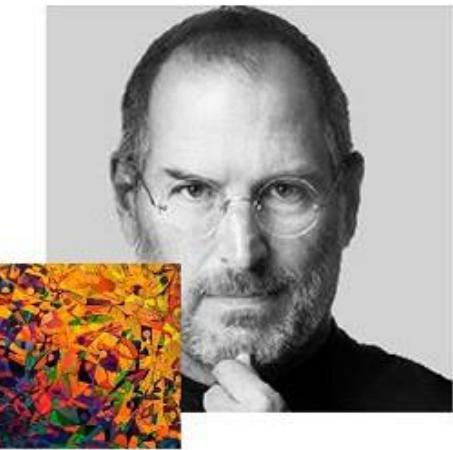
- CONVOLUTIONAL NEURAL NETWORKS

By using a perceptual loss functions based on high-level features extracted from pretrained networks, networks for image transformation tasks can be trained, and by fine tuning the loss function different features can be kept for the source image and the style image.

Justin Johnson, Alexandre Alahi, Li Fei-Fei; Perceptual Losses for Real-Time Style Transfer and Super-Resolution, 2016









Using GANs to drive design decisions

- GENERATIVE ADVERSARIAL NETWORKS

By taking advantage of Generational Adversarial Networks, synthetic images based on the training data can be generated. Including an external array of features, the generated images can be tailored to a specific set of requirements.

Jaime Deverall, Jiwoo Lee, Miguel Ayala; Using Generative Adversarial Networks to Design Shoes

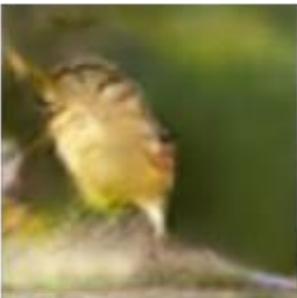


Text
description

This bird is blue with white and has a very short beak



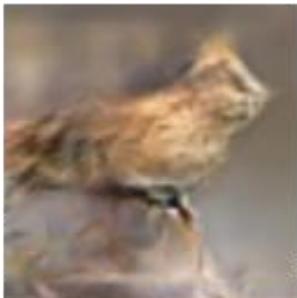
This bird has wings that are brown and has a yellow belly



A white bird with a black crown and yellow beak



This bird is white, black, and brown in color, with a brown beak



The bird has small beak, with reddish brown crown and gray belly



This is a small, black bird with a white breast and white on the wingbars.



This bird is white black and yellow in color, with a short black beak



Stage-I
images

Stage-II
images

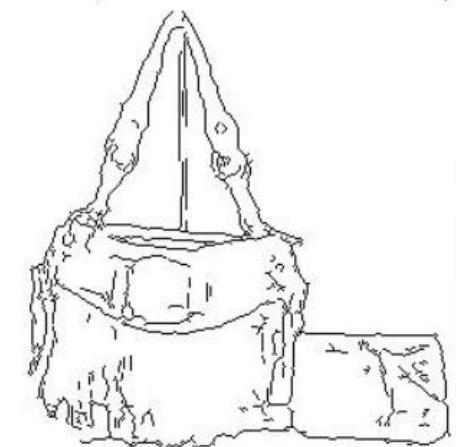
Input



Ground truth



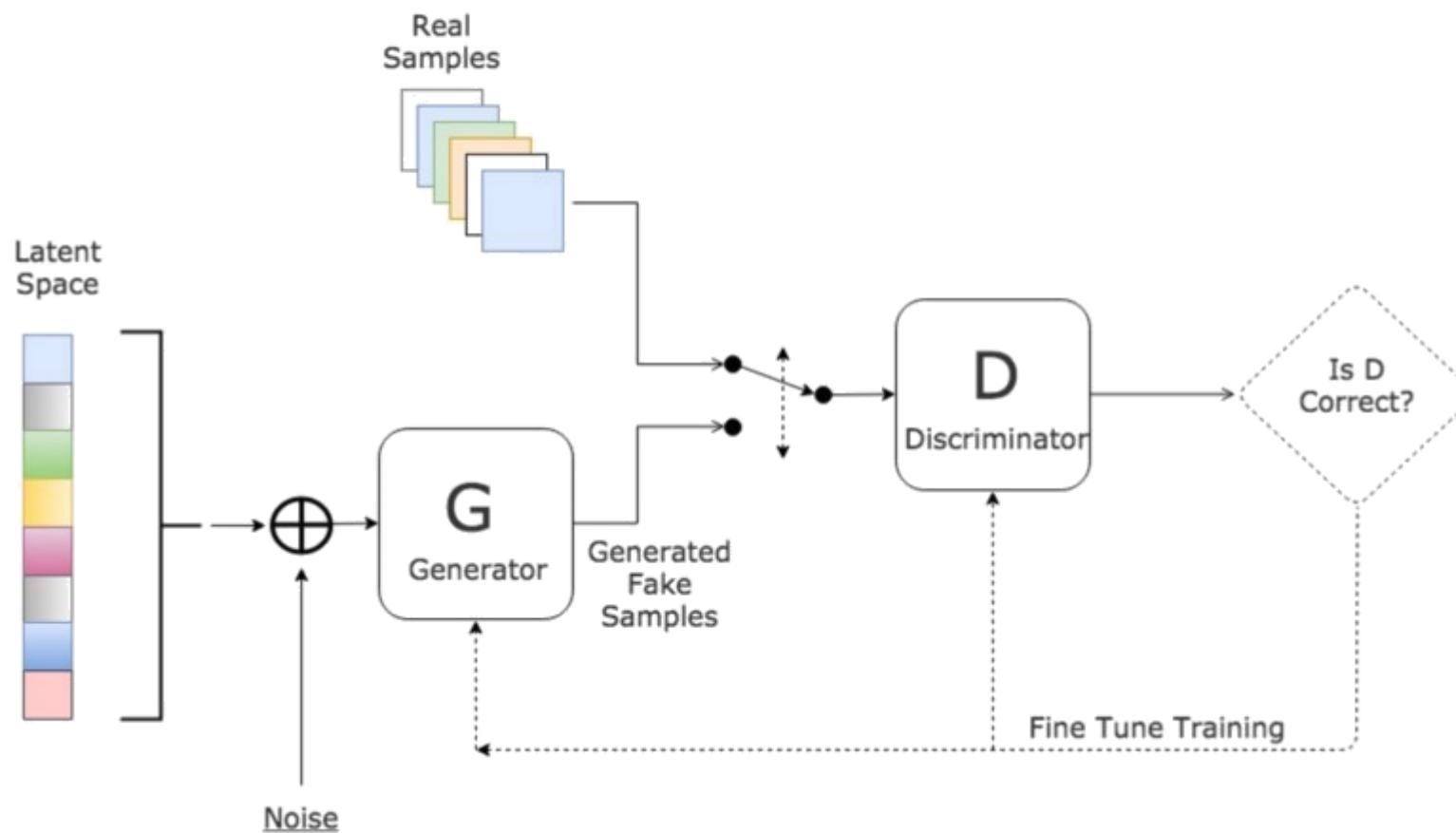
Output





HOW DOES THIS WORK?

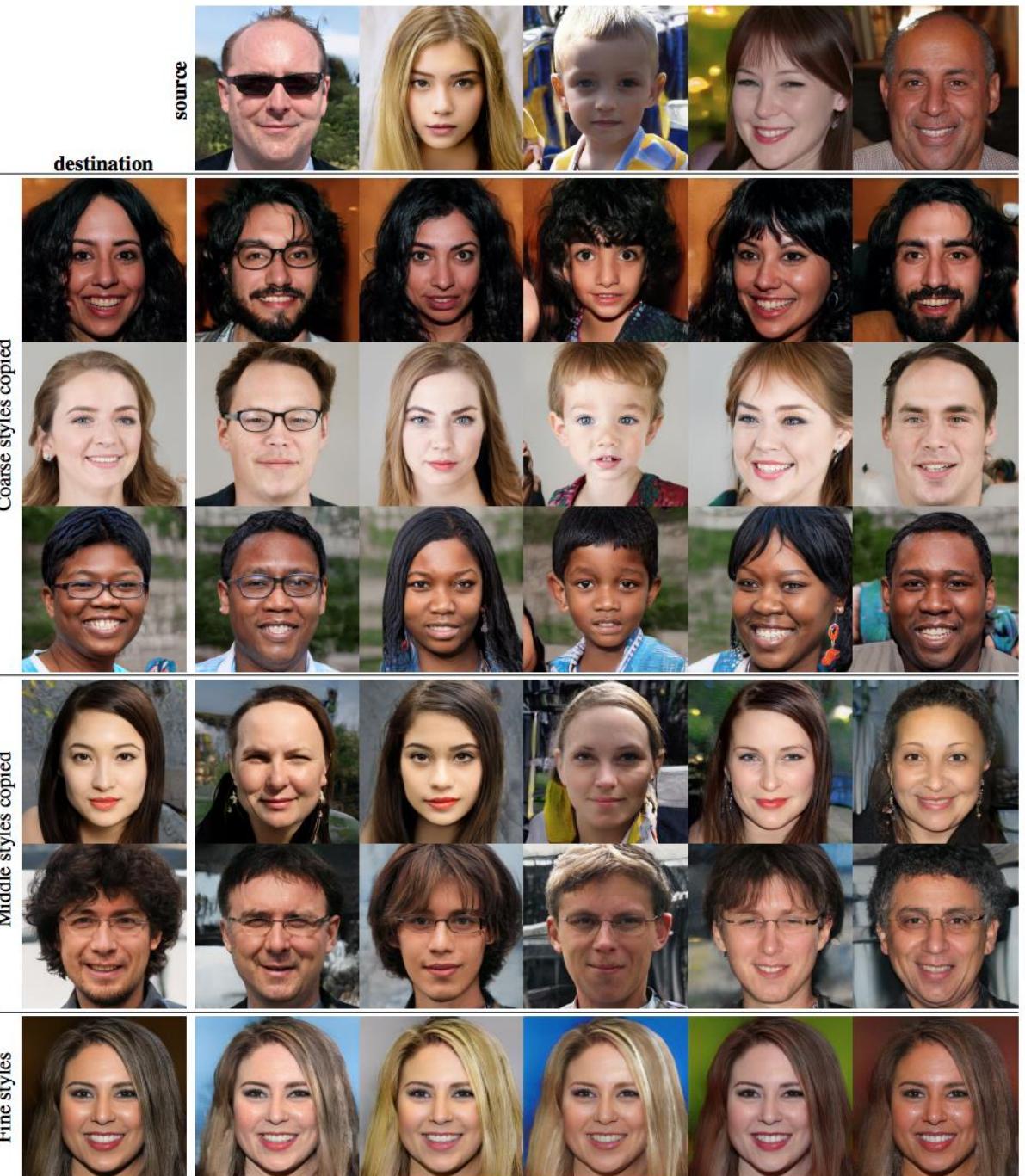
Generative Adversarial Network



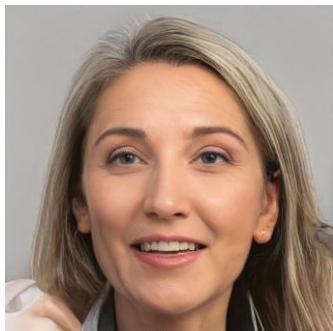
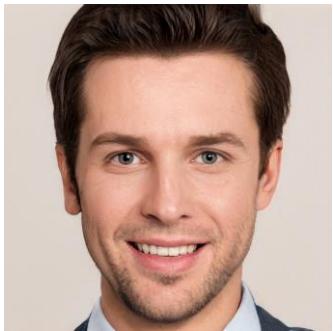
NVIDIA GauGAN Beta Demo

StyleGAN = GAN + Style Transfer

by NVlabs



Even create your own people



What about generating text?

GPT-2

- OpenAI project
- Based on Transformer approach
- Training on larger datasets:
 - books, webs...
- Increases:
 - Reading comprehension
 - Translation
 - Summarization
 - QA
- Entire code not publish → Model with 1.5b words

After leaving my friends at the bar, I went to the cinema and sat by the screen. I watched the entire 90' film. Then I walked home, which I had intended for a few months. In the process of doing so, I decided that the film was not quite as moving as I had initially imagined, and this made me feel more uncomfortable.

It was only a few minutes after the flick that my girlfriend noticed me looking at things in my pockets. I said I was just looking for something to read, and she said, "It's so bad, you could die!"

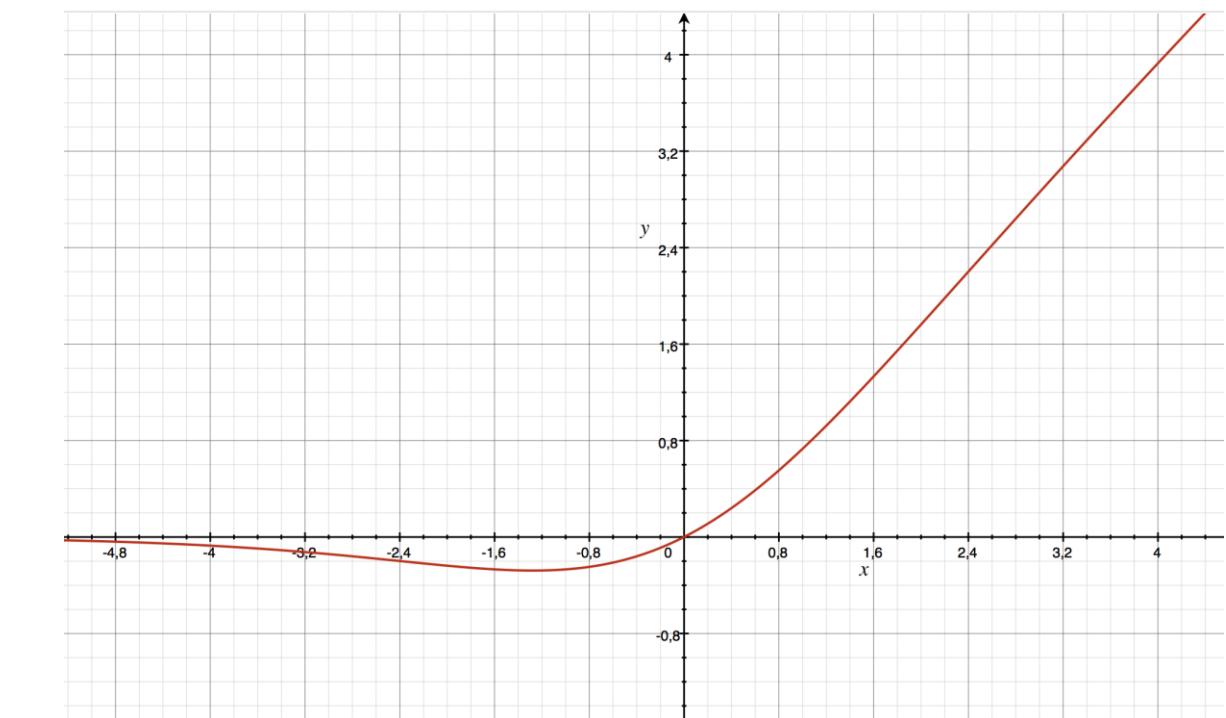
"What are you talking about," I said.

"Oh, what are you talking about? I'm going to die, what are you talking about?" she asked me.

I had been listening to music in silence and then suddenly I was looking around the room, trying to get this feeling in my body out of my head. By the time I finished reading, it was too late. I had already had a vision I have heard described with various theories. At the time, I thought it was something mystical, something out

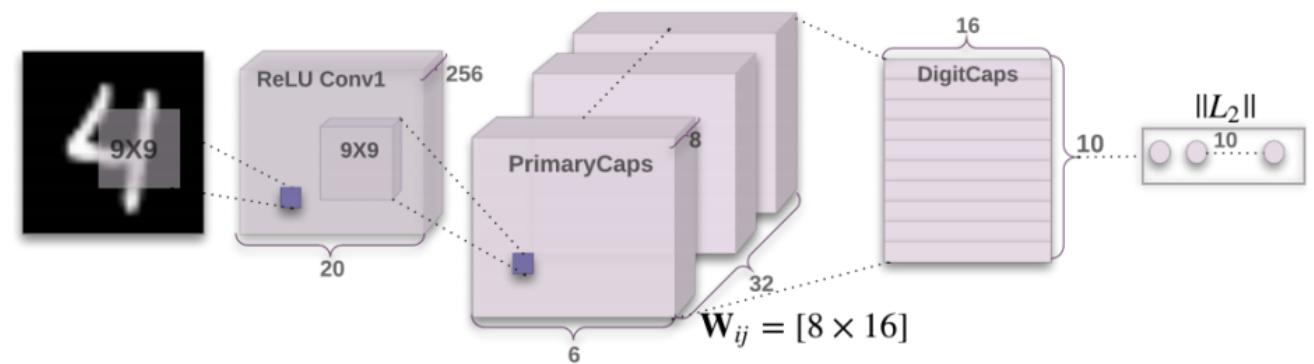


MOVING FAST!





MOVING FAST!



SOME INTERESTING PAPERS

- “ImageNet Classification with Deep Convolutional Neural Networks”
 - [Krizhevsky, Sutskever, Hinton]
 - <http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>
- “Visualizing and Understanding Convolutional Networks”
 - [Zeiler, Fergus]
 - arXiv: 1311.2901v3
- “Optimization Methods for Large-Scale Machine Learning”
 - [Bottou, Curtis, Nocedal]
 - arXiv: 1606.04838v2

SOME INTERESTING PAPERS

- “Swish: A Self-Gated Activation Function”
- [Ramachandran, Zoph, Quoc V.]
- arXiv: 1710.05941v2
- “Dynamic Routing Between Capsules”
- [Sabour, Frosst, Hinton]
- arXiv: 1710.09829