LAB 0

Warm up

CHECKLIST

Azure	Free account DSVM
Anaconda	>= 5.3 Python 3.6
IDE	VS Code (or Atom, …) Jupyter
Repository	https://github.com/matallanas/AIWorkshop20
Slack	https://ugr-ai.slack.com

AI UGR Workshop

Initial Setup

- Follow the instruction on folder Lab0
- Choose your setup

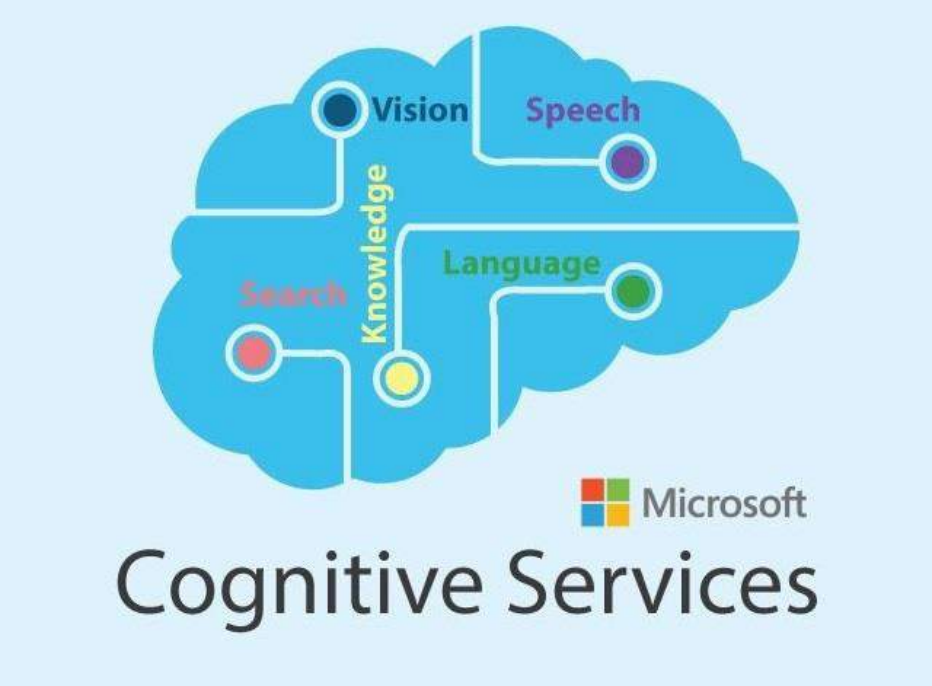
- Execute notebook "*Lab00_SmokeTest.ipynb*" *to test the environment is ready.*

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/
- 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

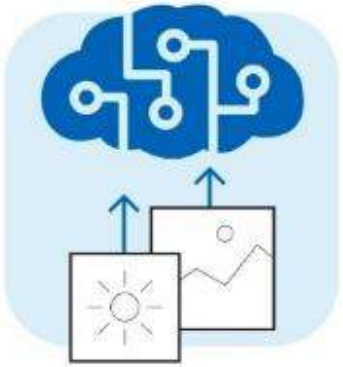

tents

LAB 2

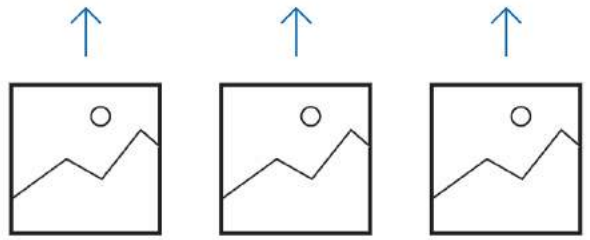
Custom Vision

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"*

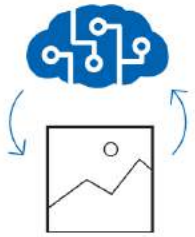


Custom Vision



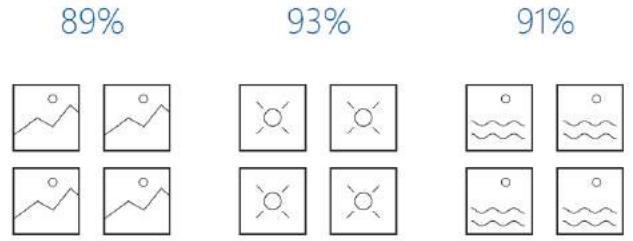
Upload Images

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%

Iteration 2

Trained : 2 hours ago
with General domain

Iteration 1

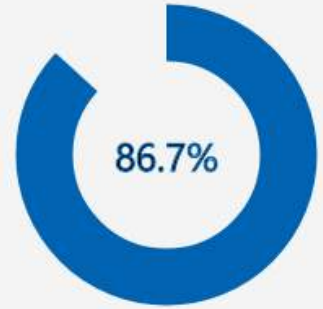
Trained : 3 hours ago
with General domain

Iteration 1

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

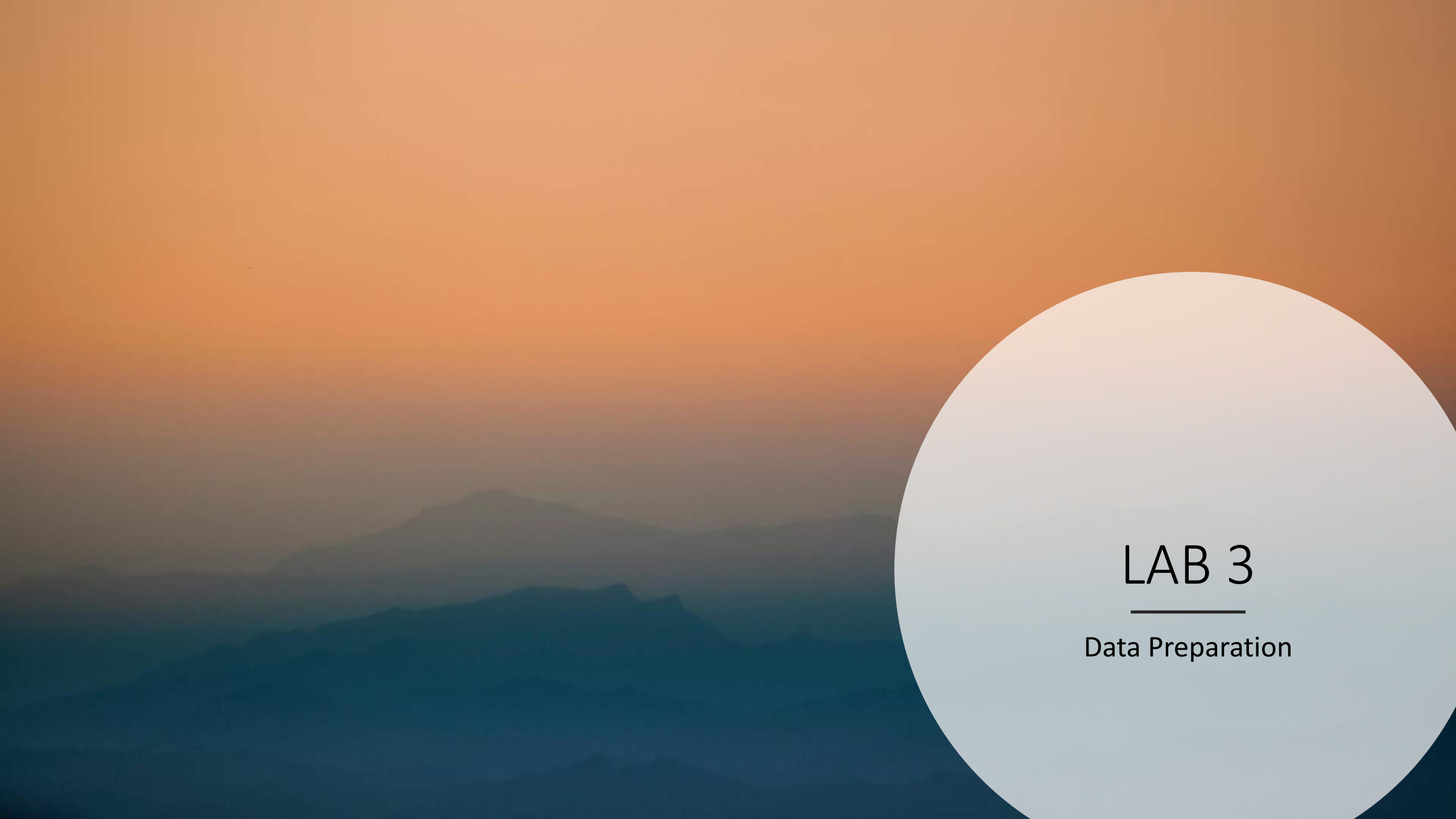
Precision

Recall



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 3

Data Preparation

Data preparation

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

Pillow

NumPy

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

https://numpy.org/

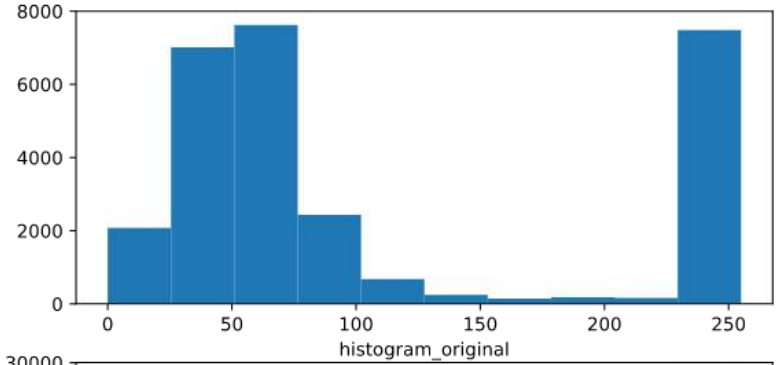
Scale


0
50
100
150
200
250
300
350
0
100
200
original

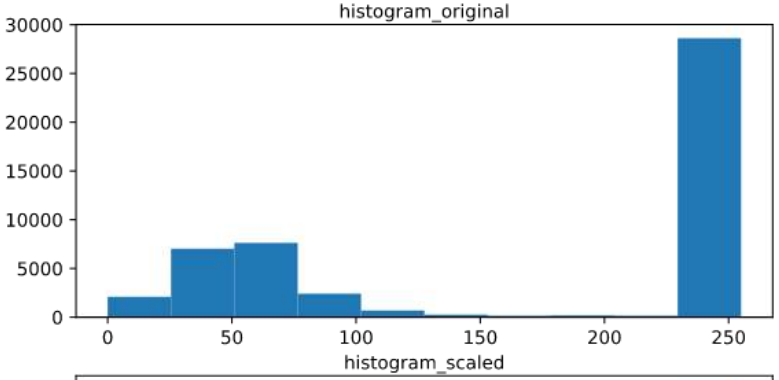

0
20
40
60
80
100
120
0
50
100
scaled

Normalize

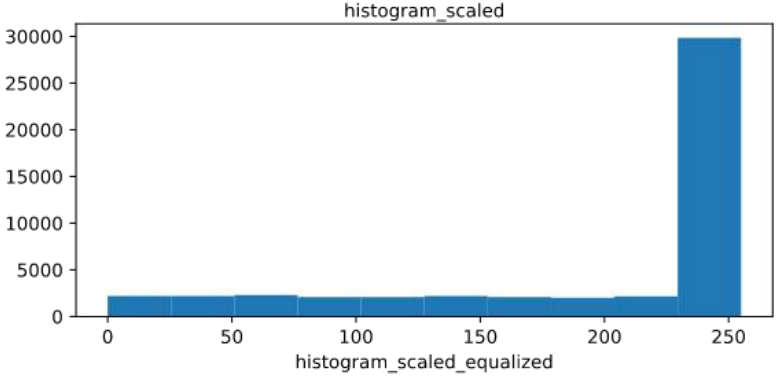

0
20
40
60
80
100
120
0
25
50
original


8000
6000
4000
2000
0
0
50
100
150
200
250
histogram_original


0
20
40
60
80
100
120
0
50
100
scaled


30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
histogram_scaled


0
20
40
60
80
100
120
0
50
100
scaled_equalized


30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
histogram_scaled_equalized

Processed


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners


carabiners

LAB 4

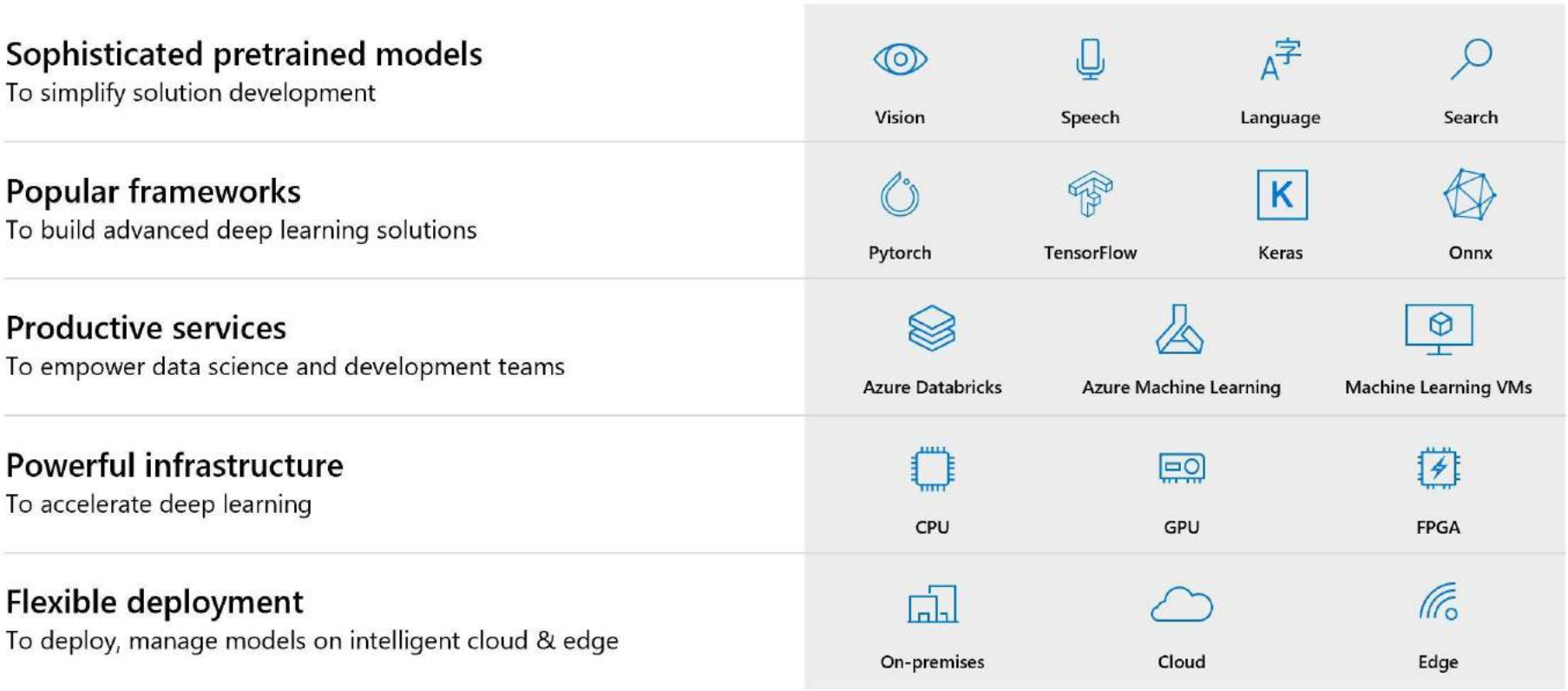
Machine Learning 101

Machine Learning

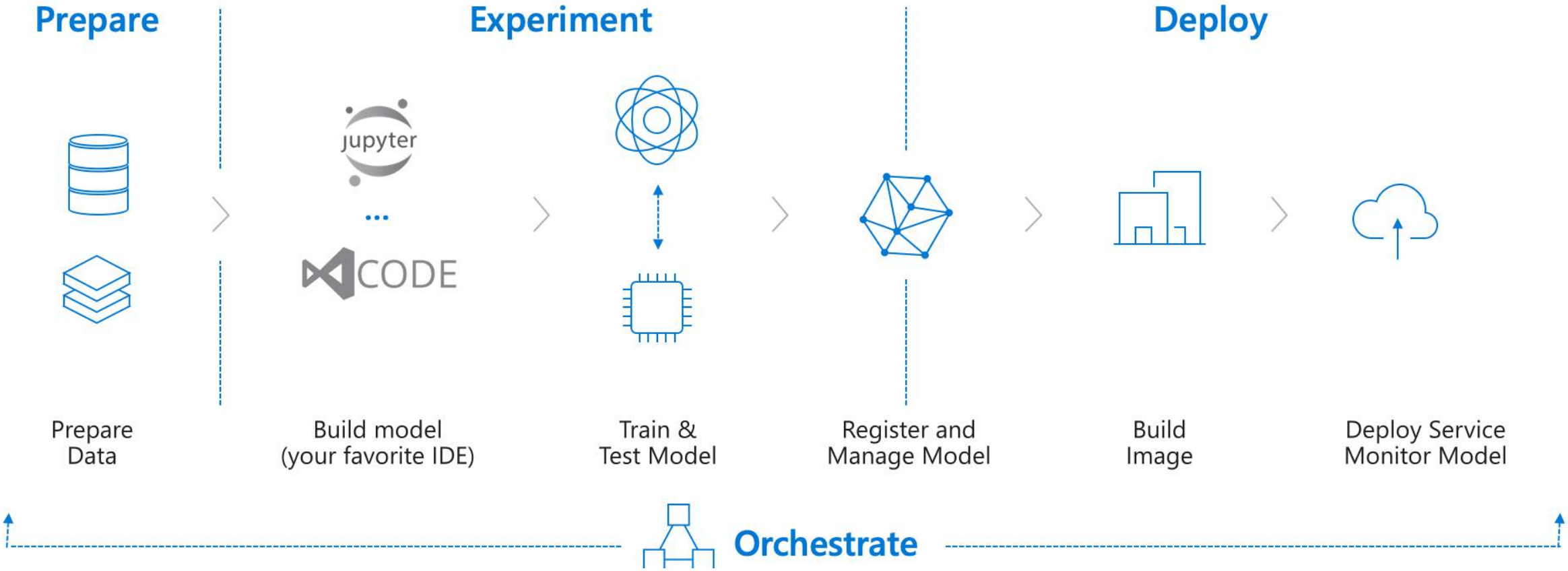
- Use a Machine Learning algorithm
- Use notebook “Lab04_MachineLearning101.ipynb”
- Use scikit-learn and any of its non deterministic algorithms.
- https://scikit-learn.org/stable/
- Use Azure Machine learning Workspace to train it and record metrics
- Save the model and register it.

Azure Machine Learning

Machine learning on Azure

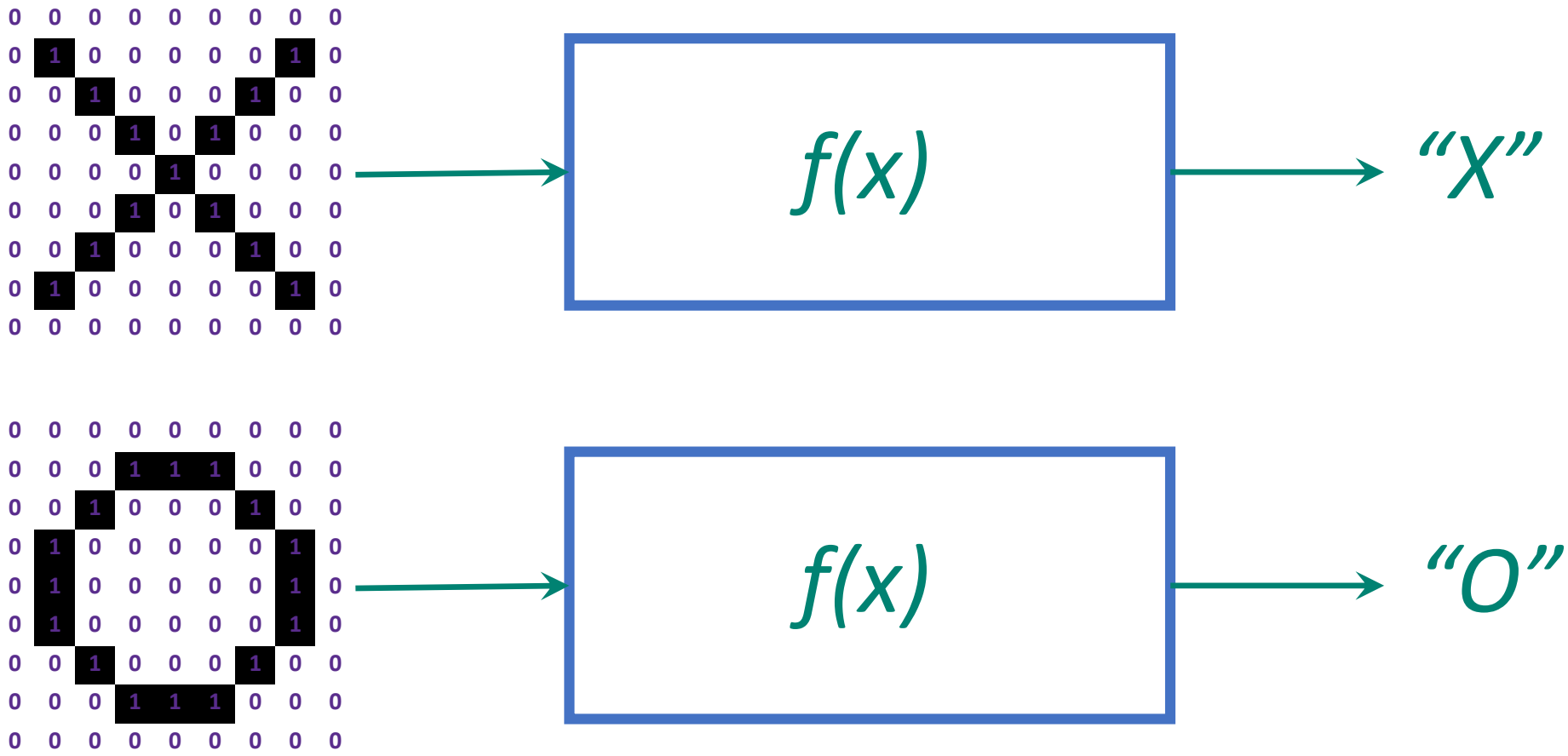


Azure Machine Learning

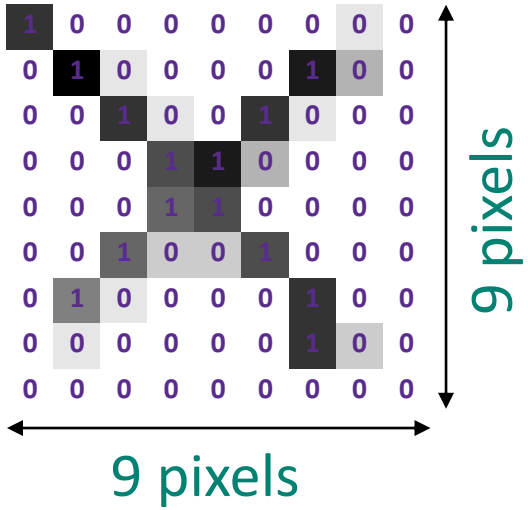


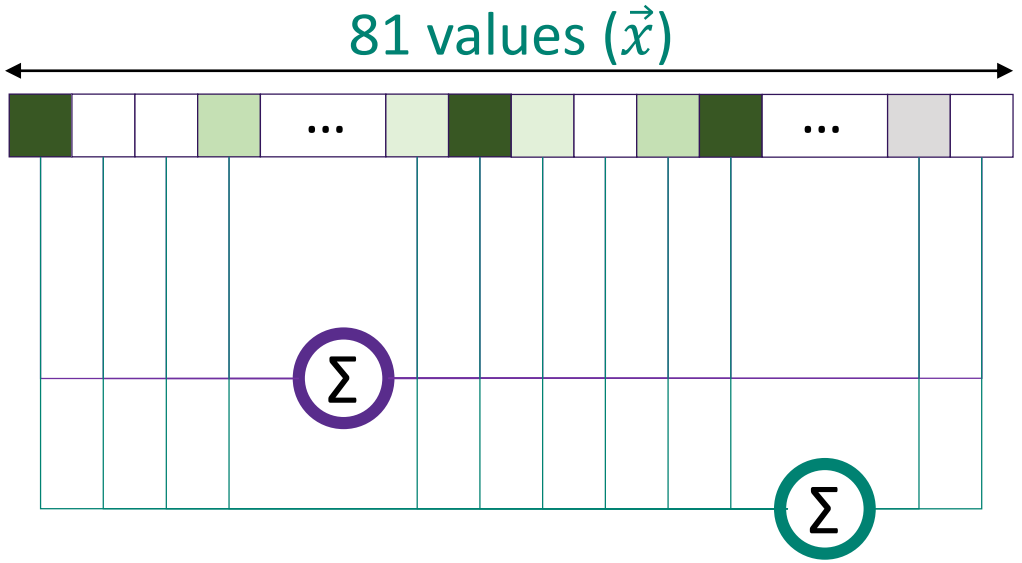
SHALL WE PLAY A LITTLE GAME?

A SIMPLE EXAMPLE



A SIMPLE EXAMPLE





Bias ($\vec{b}$)

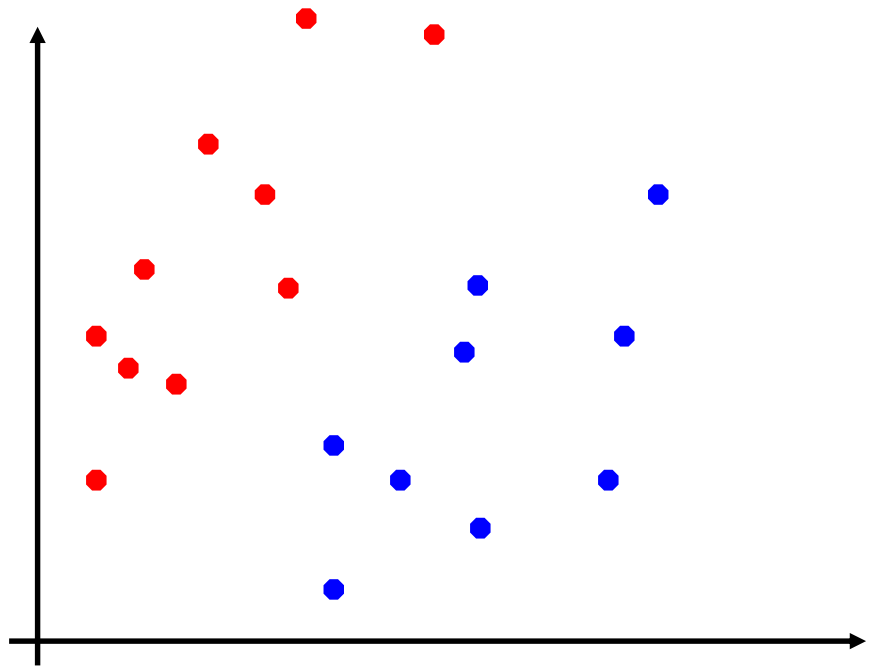
Σ *Sum (weights x pixels) =* $\overrightarrow{w_x} \cdot \vec{x}$

Σ *Sum (weights x pixels) =* $\overrightarrow{w_0} \cdot \vec{x}$

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

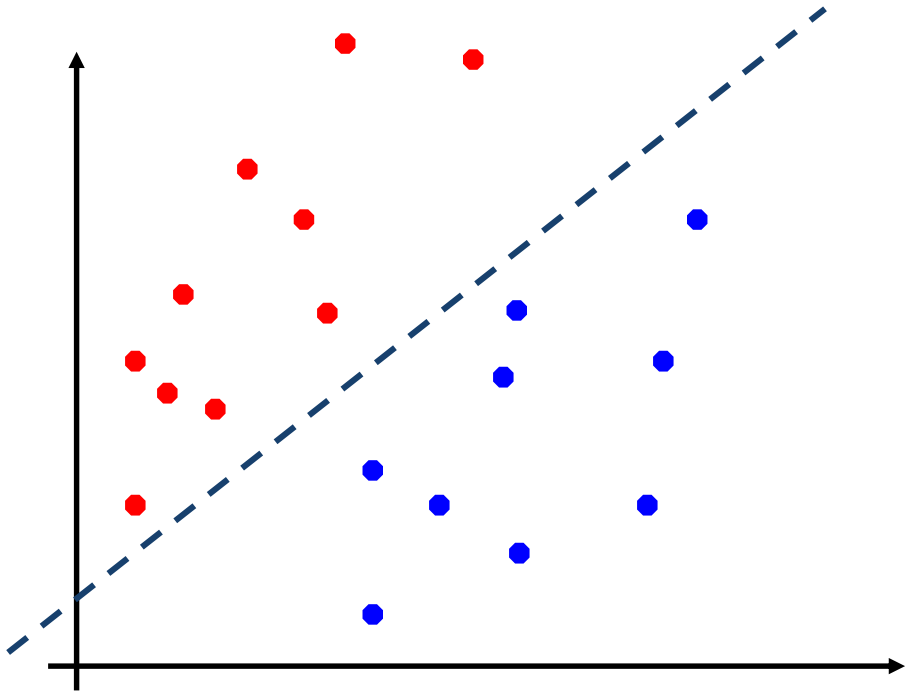
NOW THAT YOU MENTION IT...

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



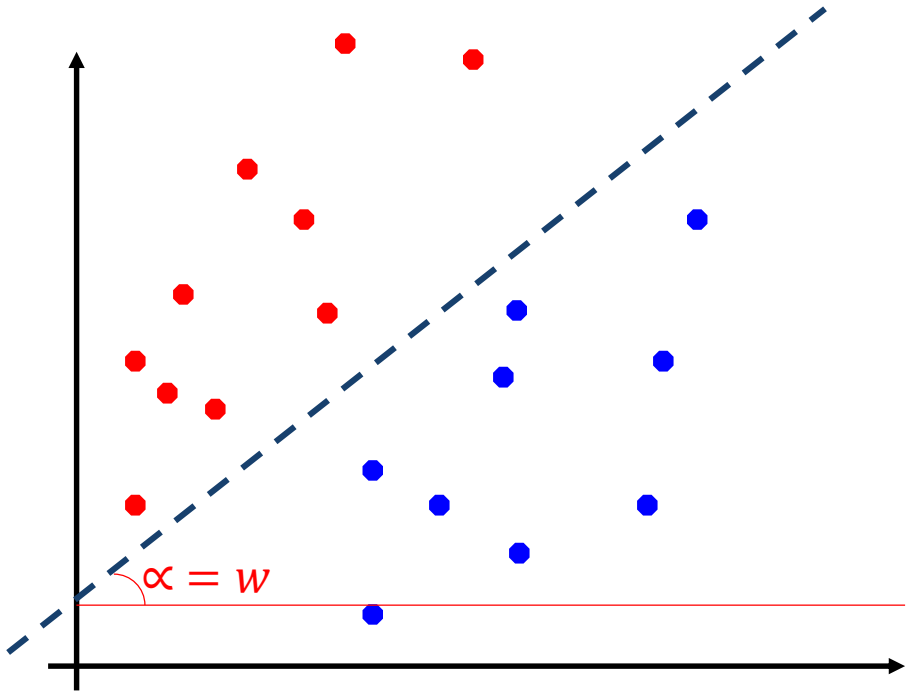
NOW THAT YOU MENTION IT...

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



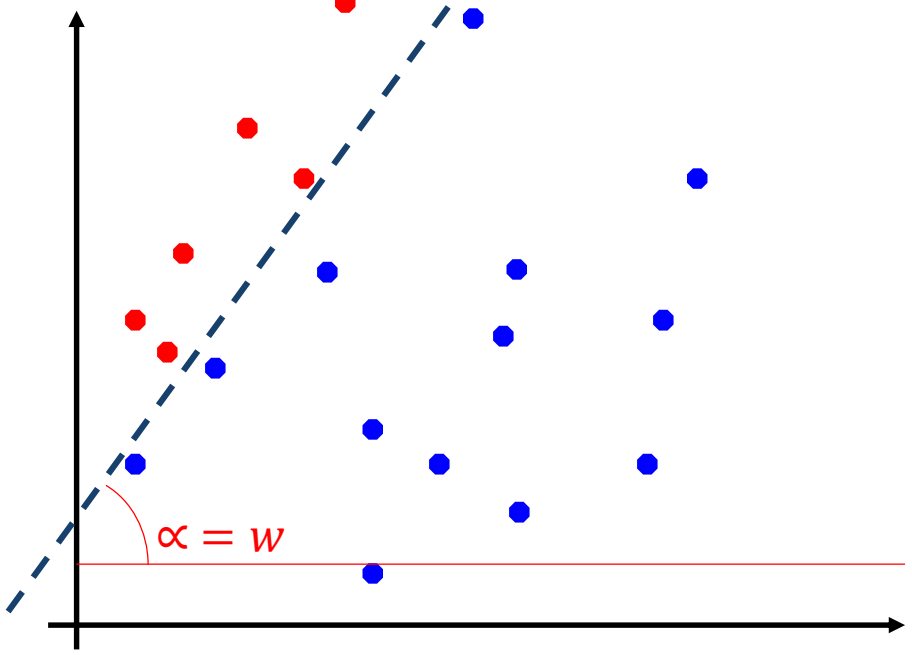
NOW THAT YOU MENTION IT...

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



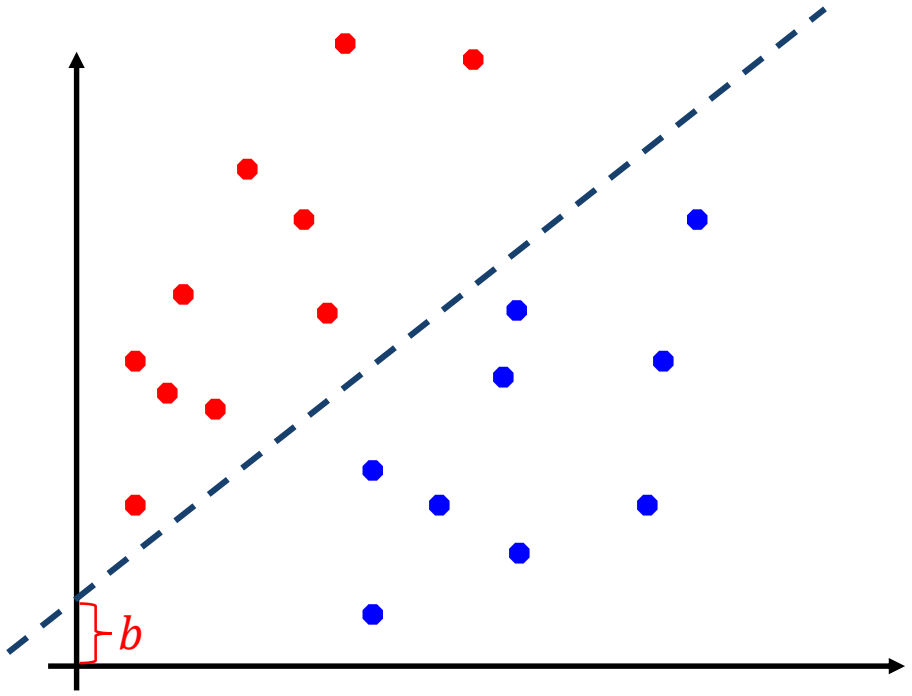
NOW THAT YOU MENTION IT...

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



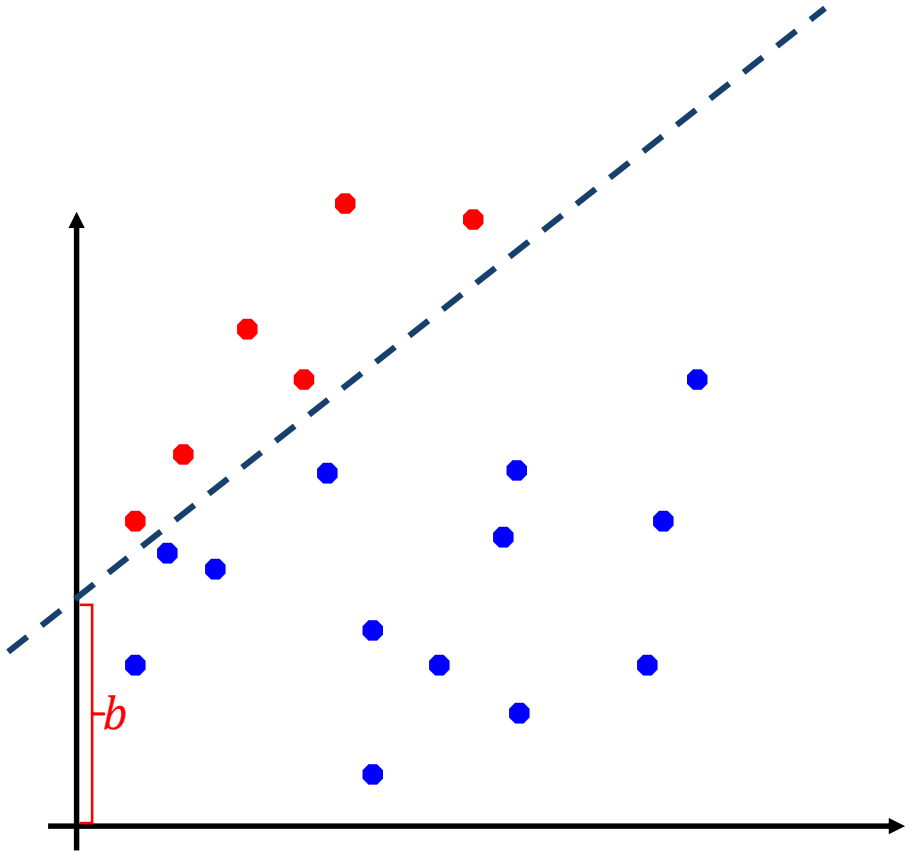
NOW THAT YOU MENTION IT...

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

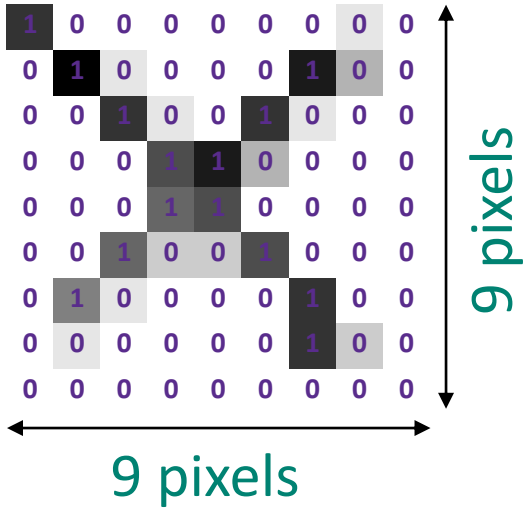


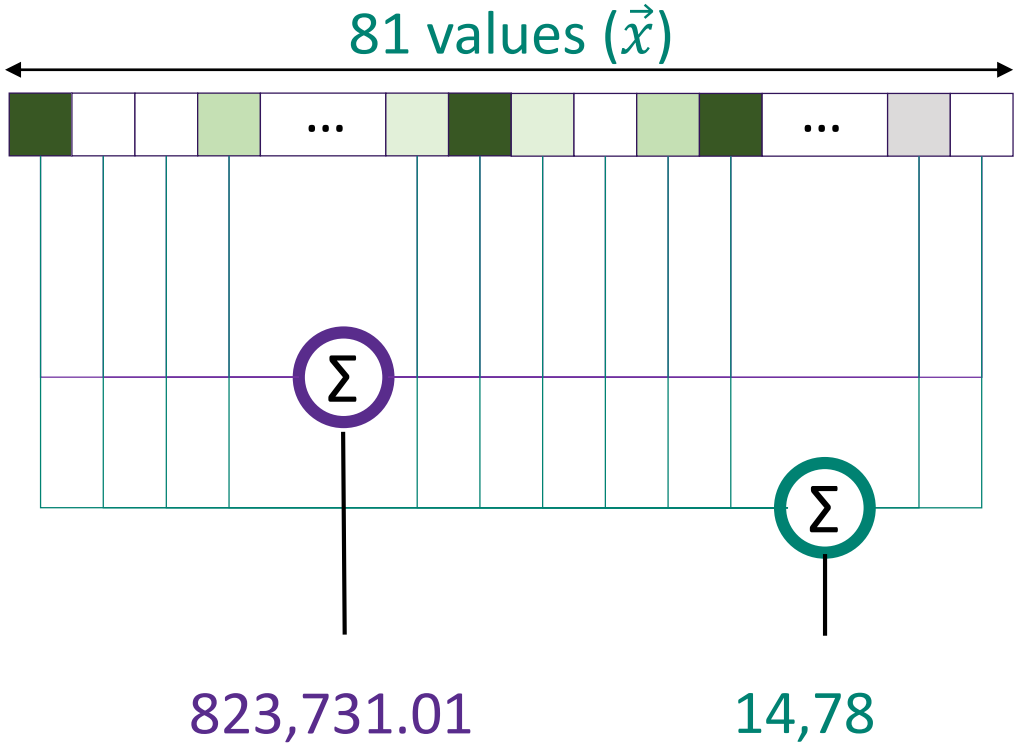
NOW THAT YOU MENTION IT...

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

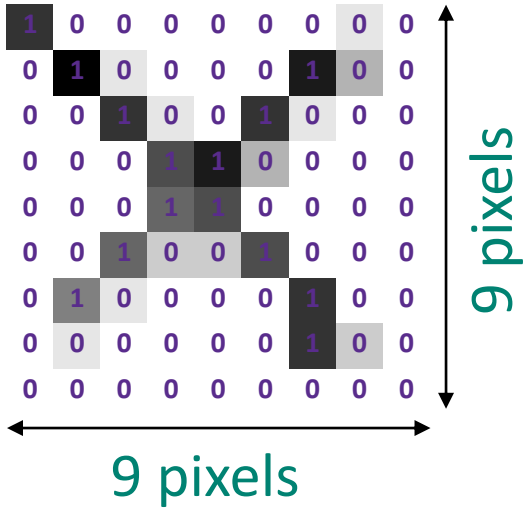


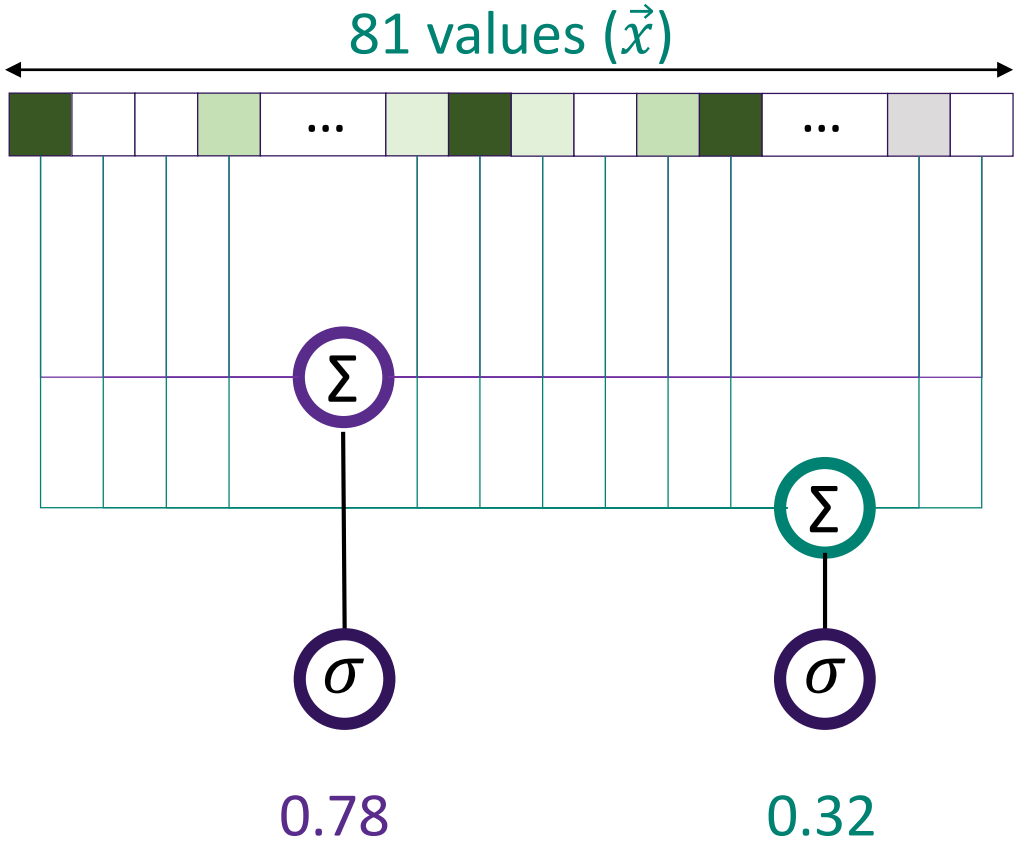
SIGMOID

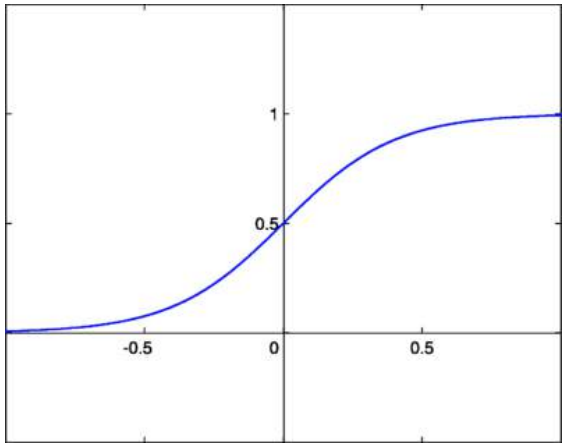

1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0
0 0 1 0 0 1 0 0 0
0 0 0 1 1 0 0 0 0
0 0 0 1 1 0 0 0 0
0 0 1 0 0 1 0 0 0
0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0
9 pixels
9 pixels


81 values ($\vec{x}$)
...
...
Σ
Σ
823,731.01
14,78

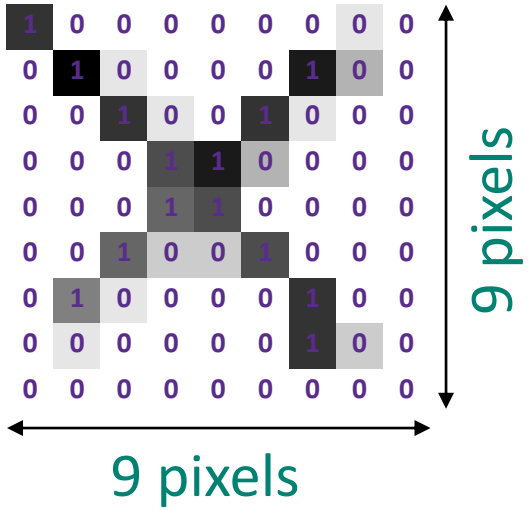
SIGMOID

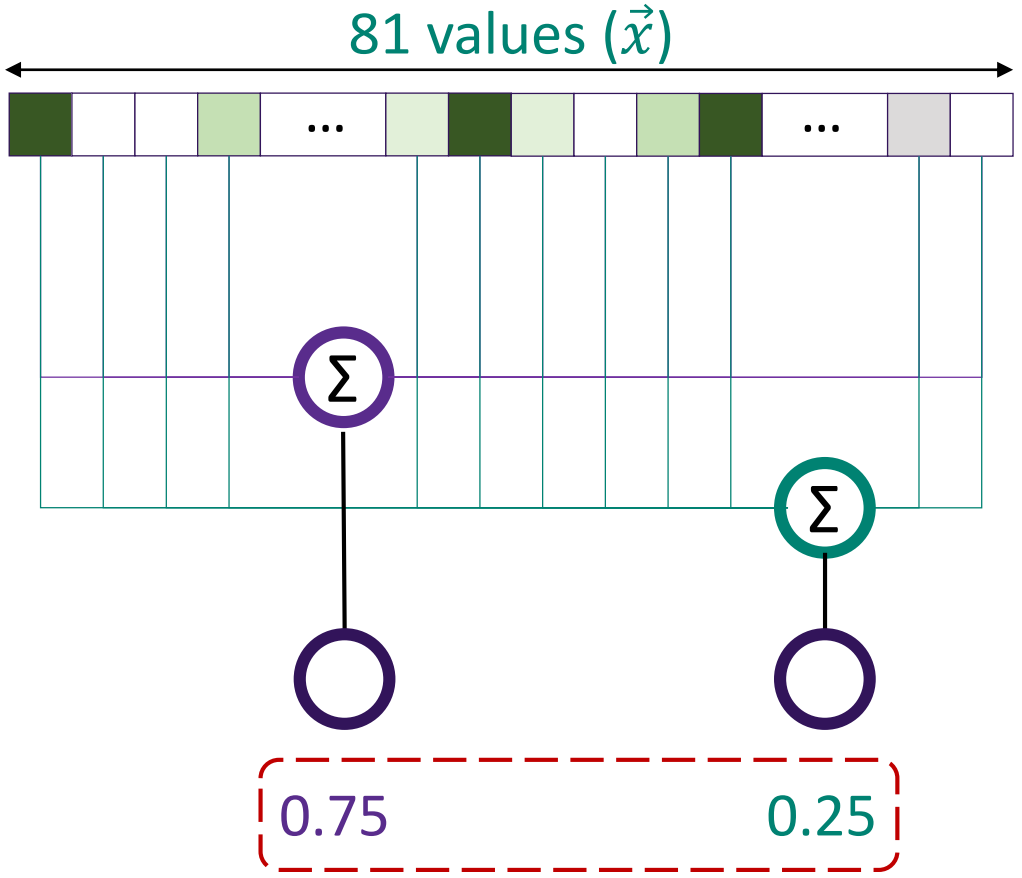






SOFTMAX

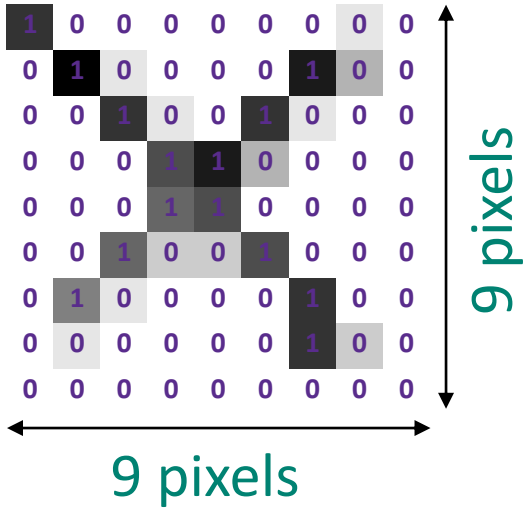




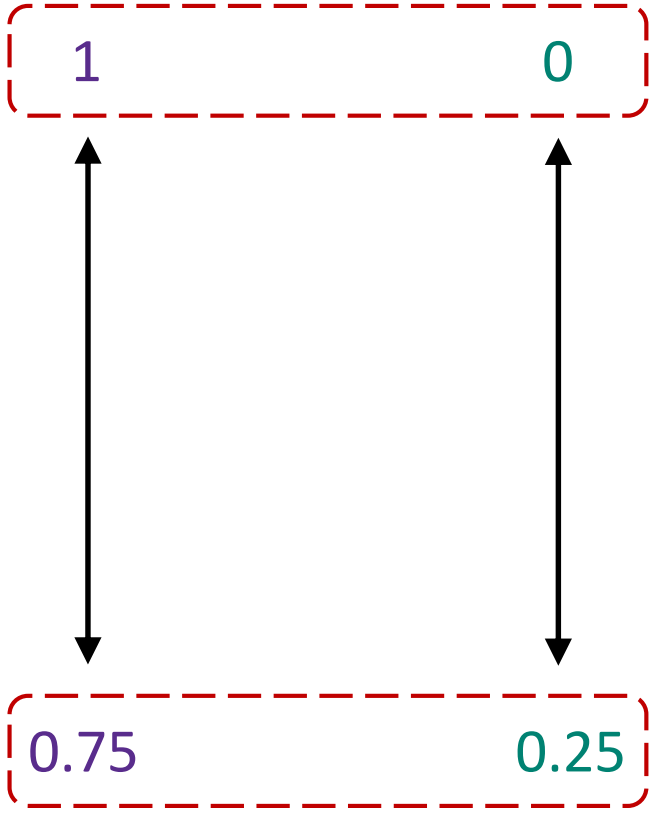
$$S_{(y_i)} = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

These are actual probabilities

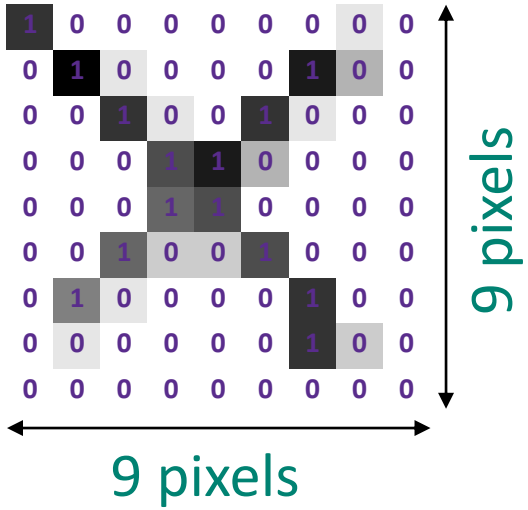
LOSS FUNCTION



One-hot encoding



LOSS FUNCTION

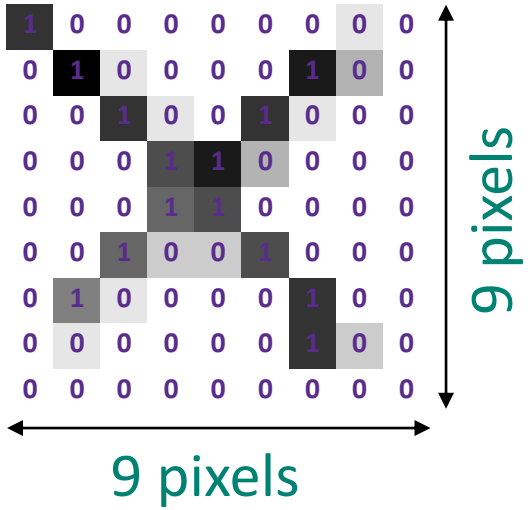


1	0

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

0.75	0.25

LOSS FUNCTION

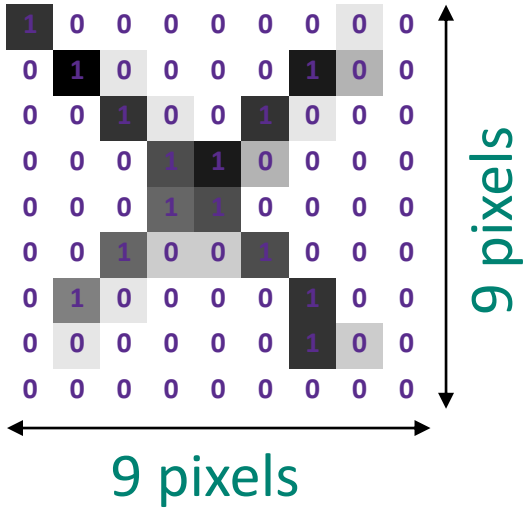


1 0

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

0.75 0.25

LOSS FUNCTION



1 0

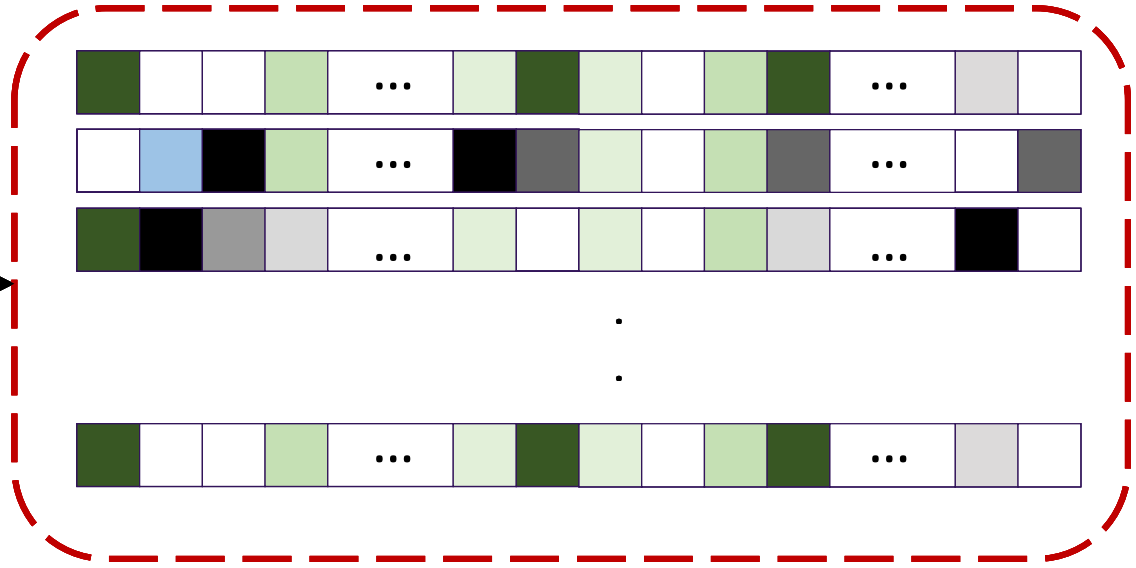
Cross Entropy Error = $-\sum_{c=1}^{M} y_{0,c} \log(p_{0,c})$

Binary: −(y*log(p) + (1−y)*log(1−p))

0.75 0.25

TRAINING

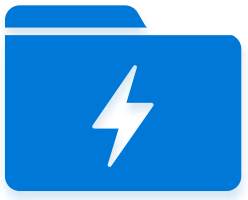
x = vector of pixels (input)



z = times(x, W) + b

cross_entropy_with_softmax(z, y)

classification_error(z, y)



Training Data

(1, 0)

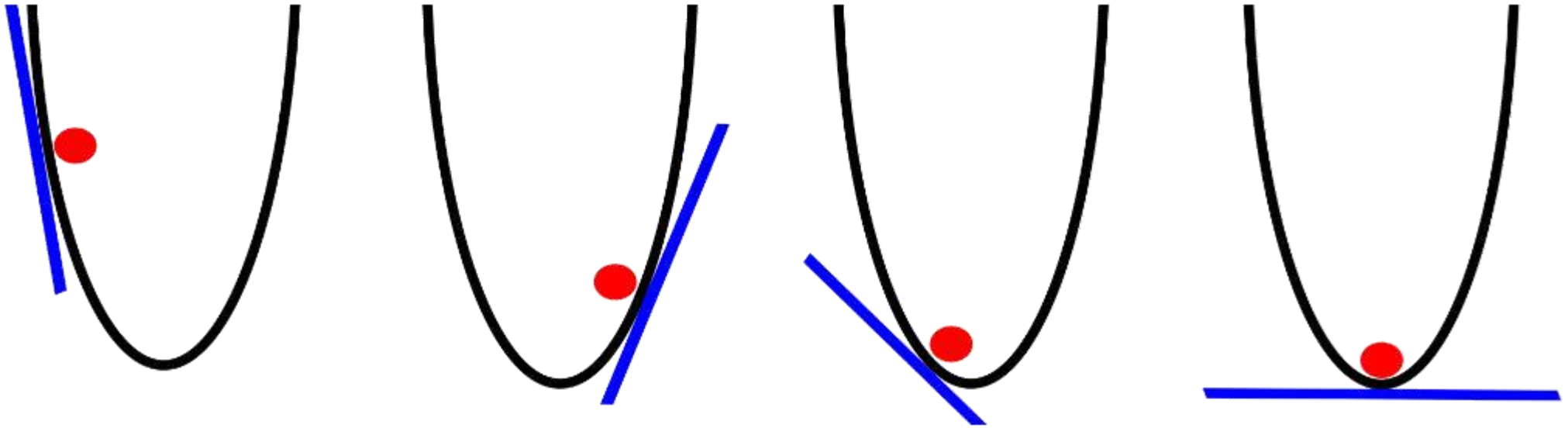
(0, 1)

(1, 0)

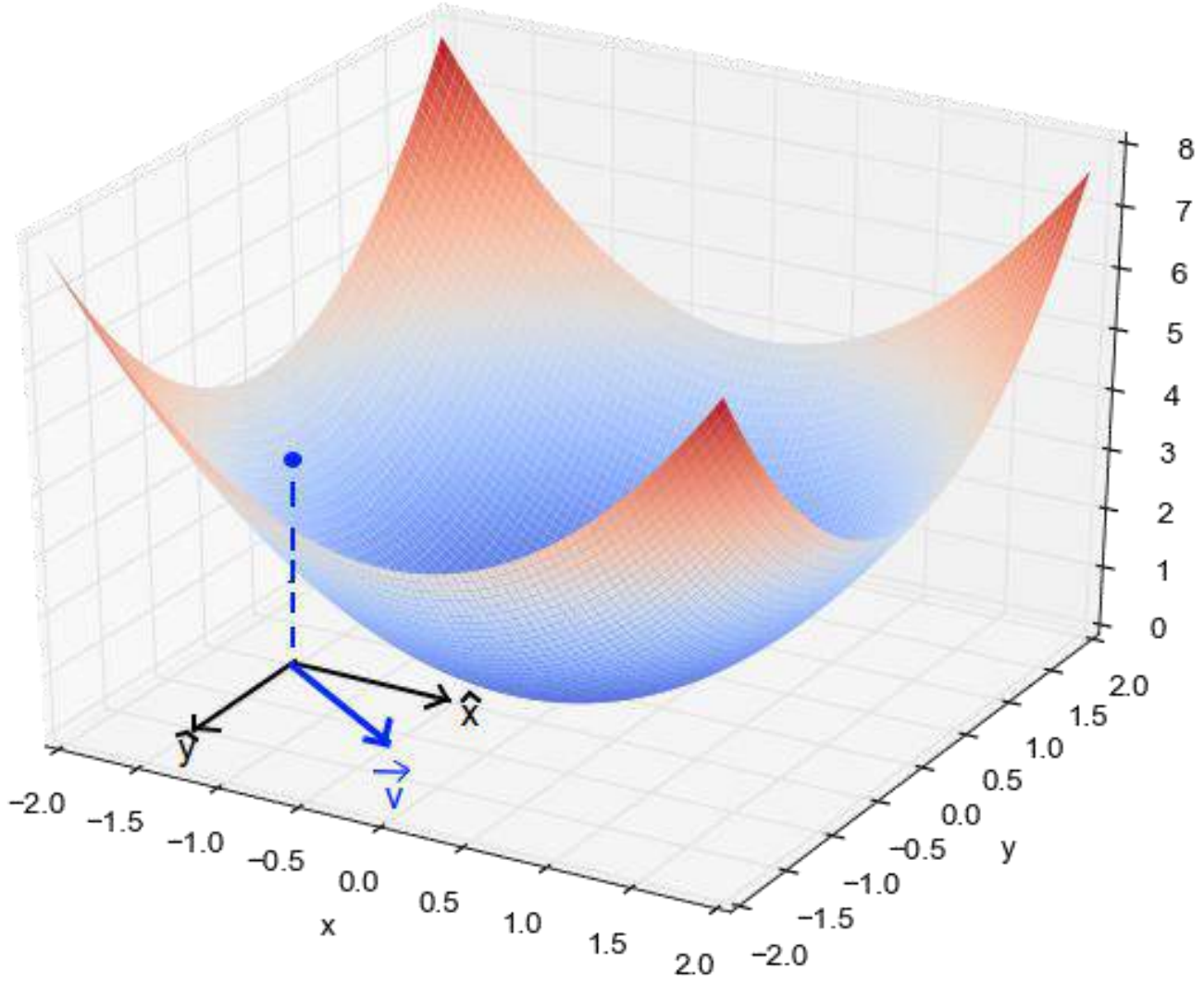
y = label vector

(one-hot)

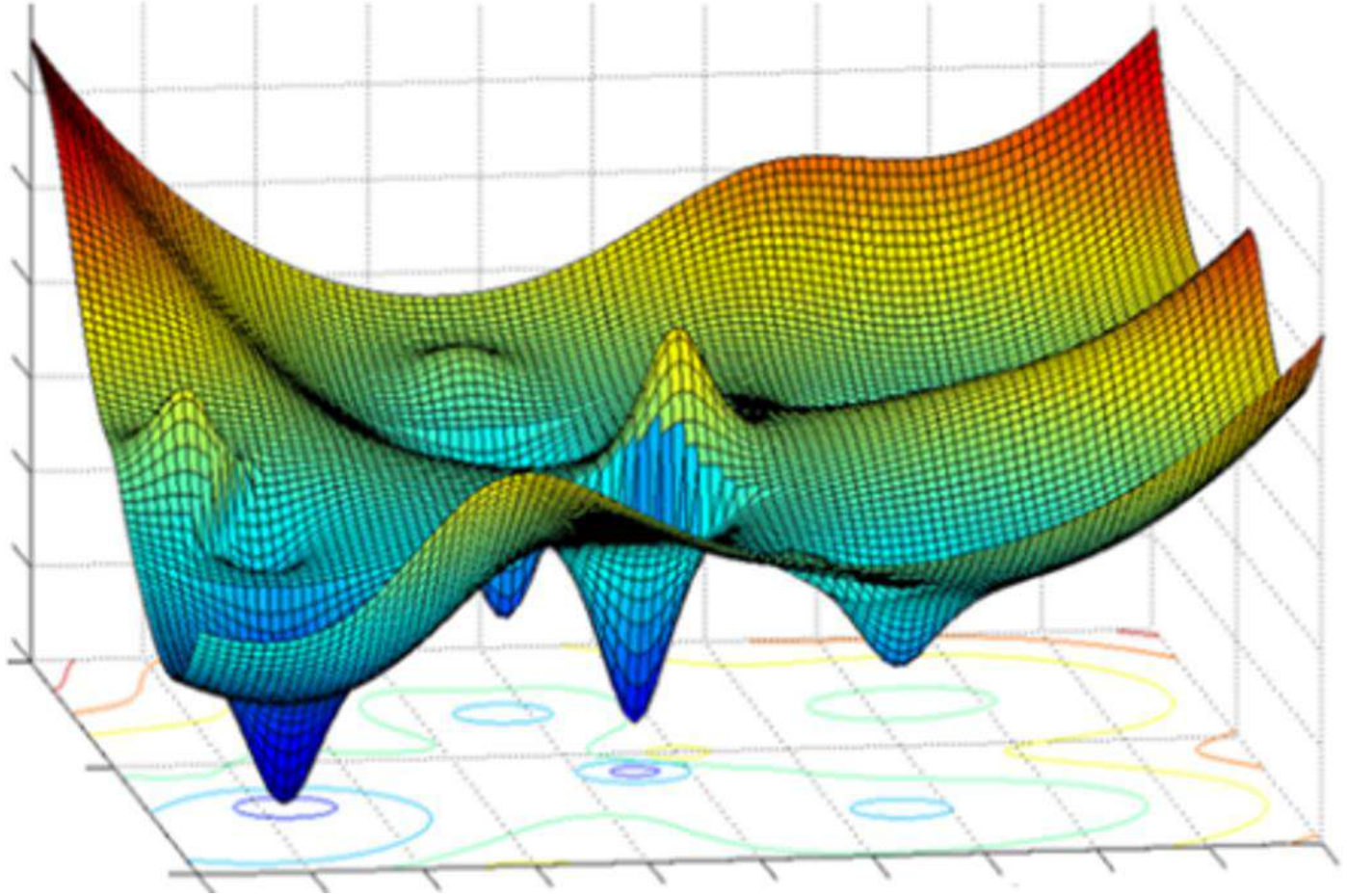
GRADIENT DESCENT

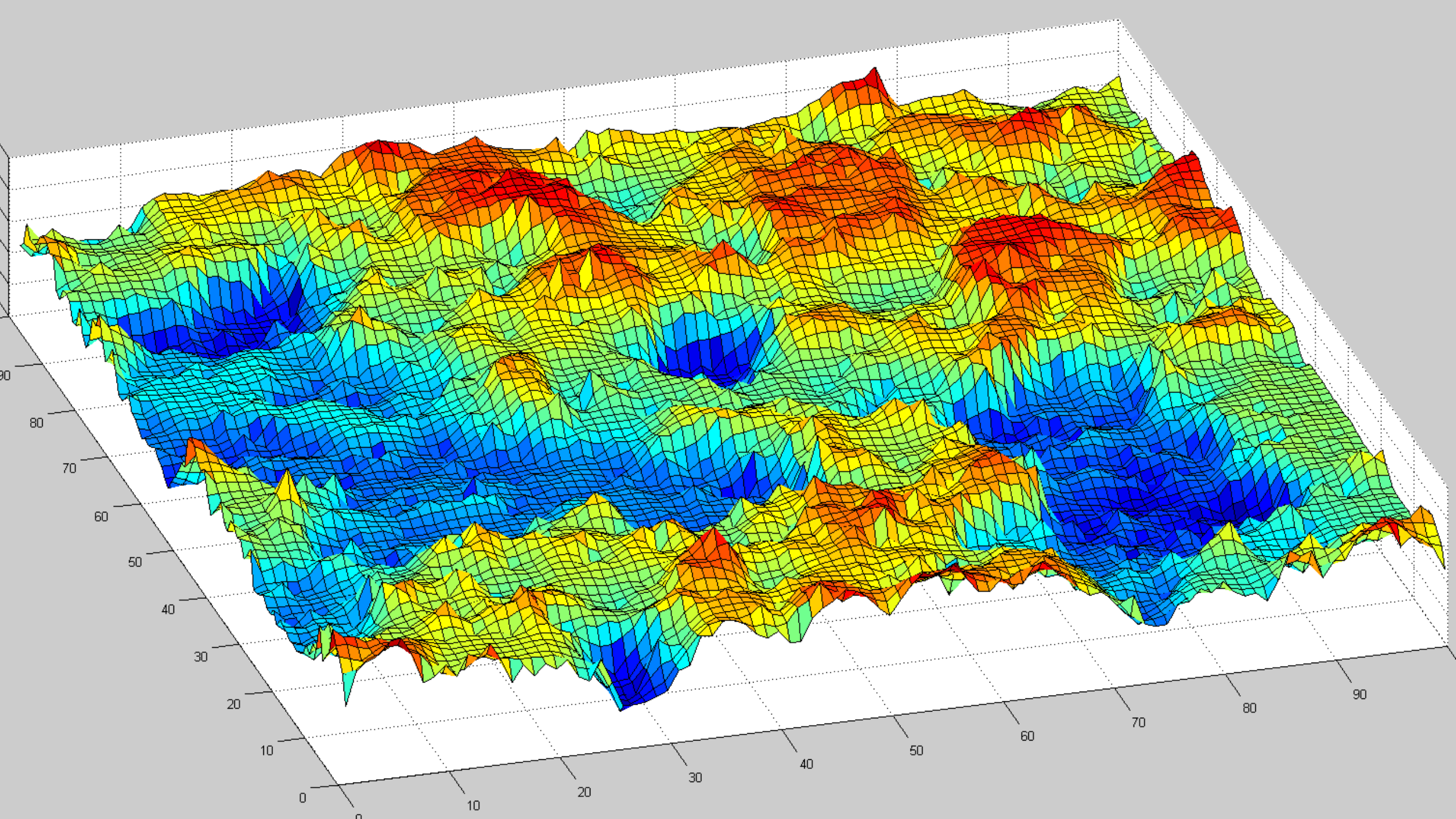


GRADIENT DESCENT



GRADIENT DESCENT




90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90

LAB 5

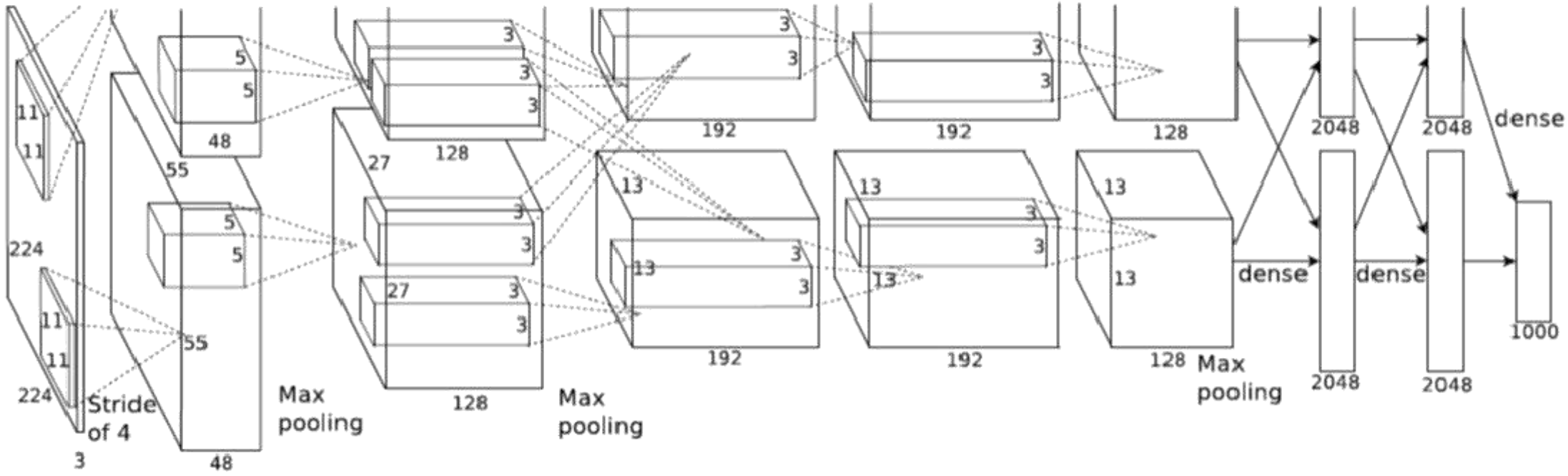
Deep Learning

Deep Learning

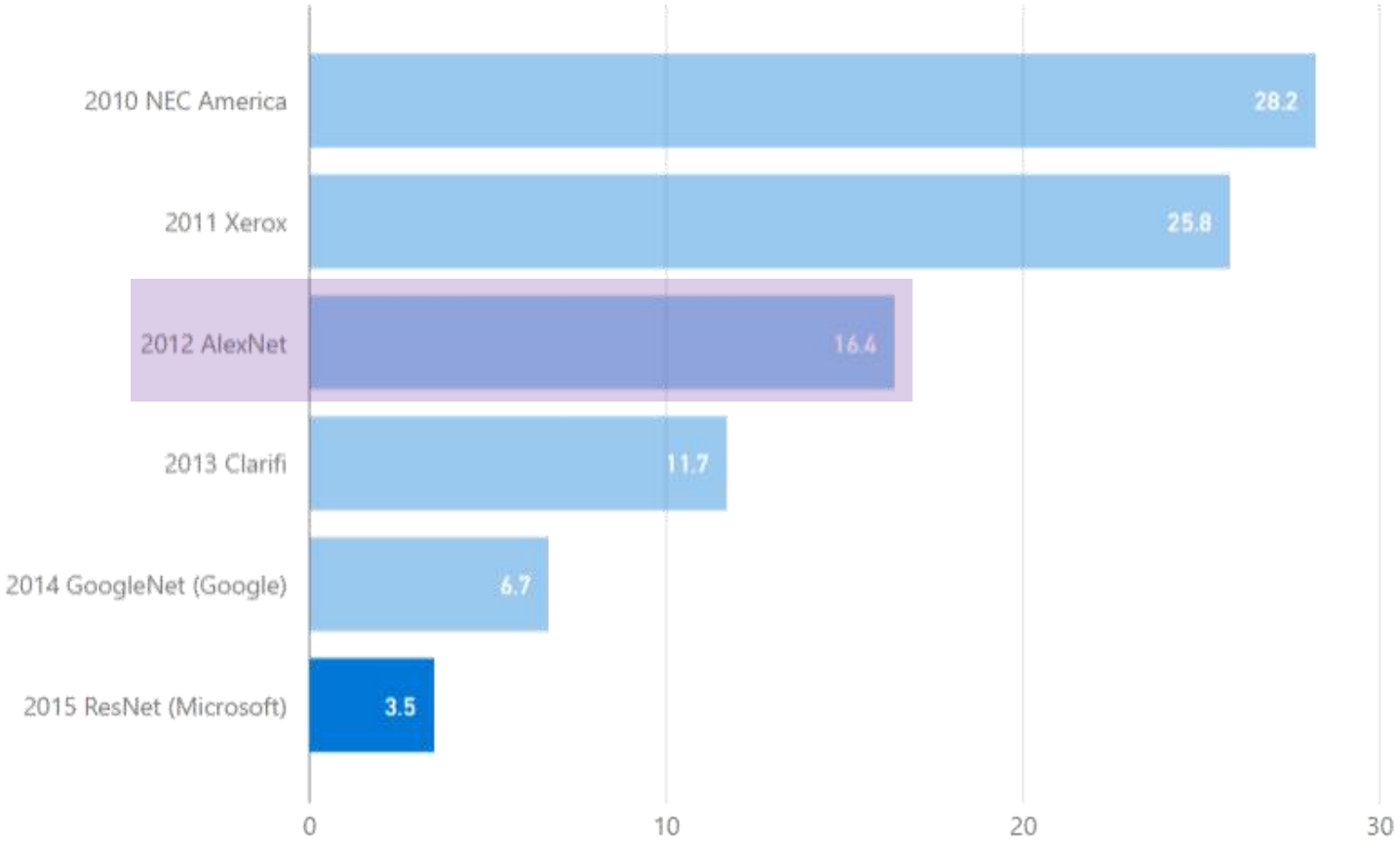
- Create a Deep Learning Model.
- Use notebook “Lab05_DeepLearning.ipynb”
- Try to create a Convolutional Neural Network to classify image per category.
- https://www.tensorflow.org/
- Use Azure Machine learning Workspace to train it and record metrics.
- Save the model and register it.

Convolutional Networks

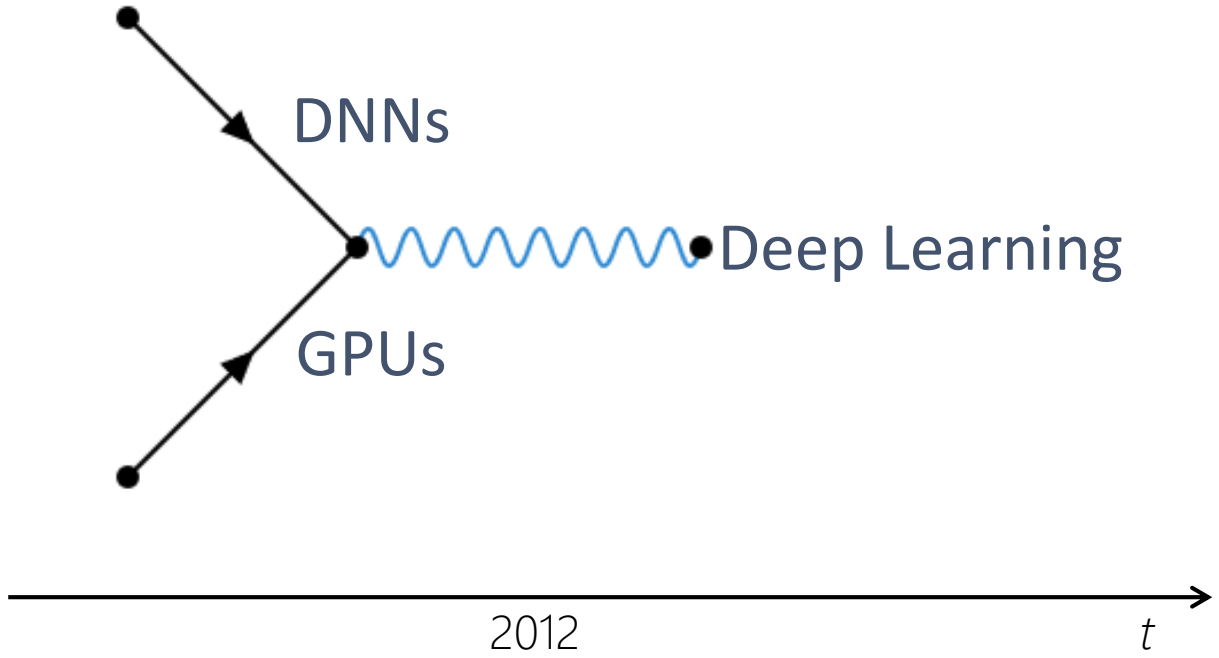
REMEMBER THIS THING?



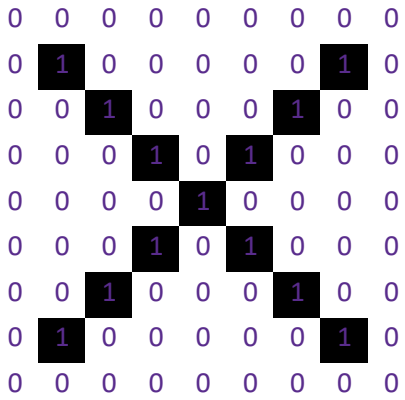
IMAGENET CHALLENGE (CLASSIFICATION)

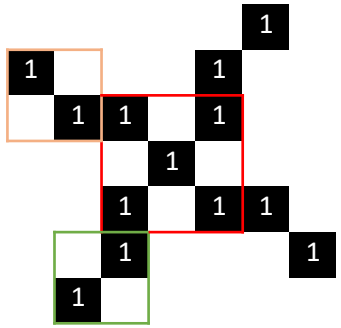


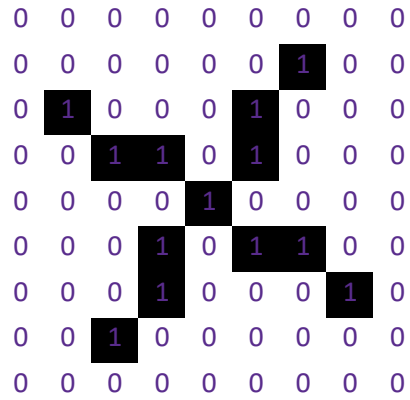
WHAT HAPPENED IN 2012?



FILTERING


0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
0 0 1 0 0 0 1 0 0
0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0

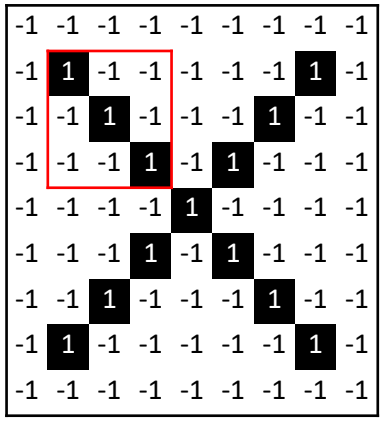

1
1 1
1 1 1
1
1 1 1
1 1
1

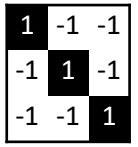

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0
0 1 0 0 0 1 0 0 0
0 0 1 1 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 1 1 0 0
0 0 0 1 0 0 0 1 0
0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

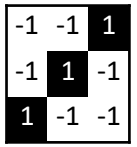
FILTERING

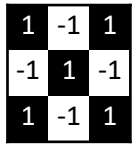
…looking for common characteristics

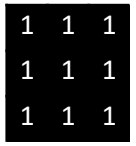
- Filter 1 Filter 2 Filter 3

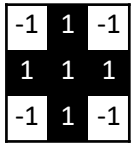


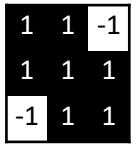






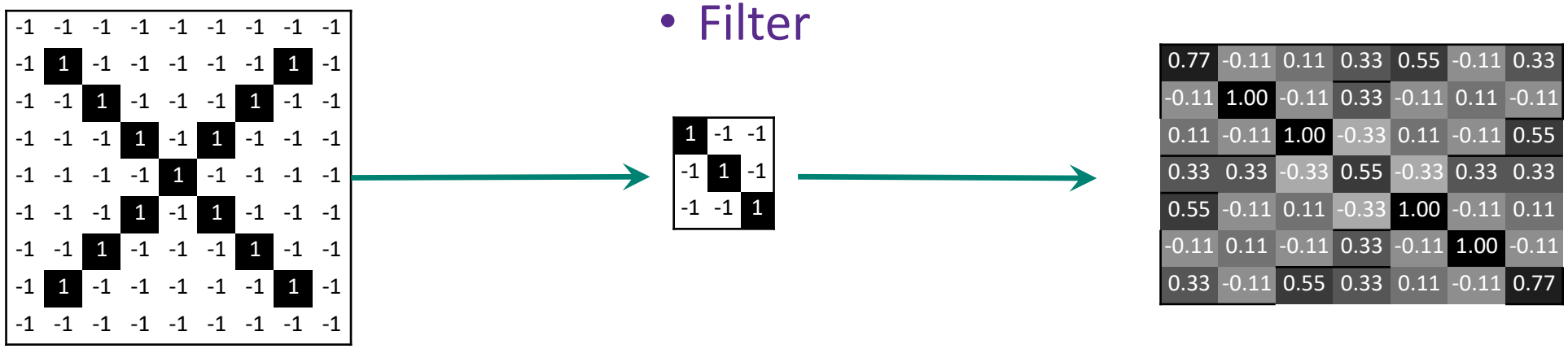




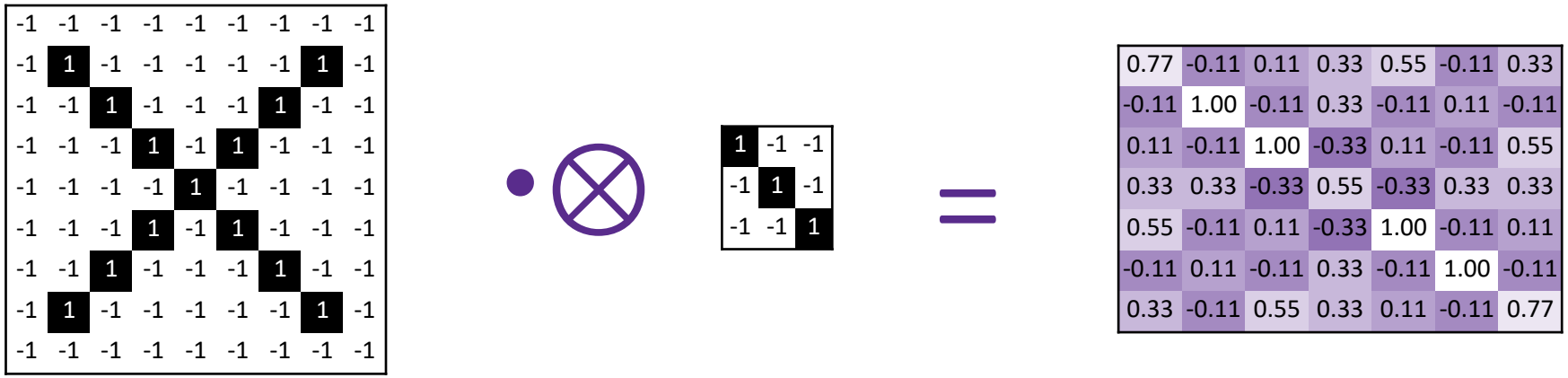


1.0 0.11 0.55

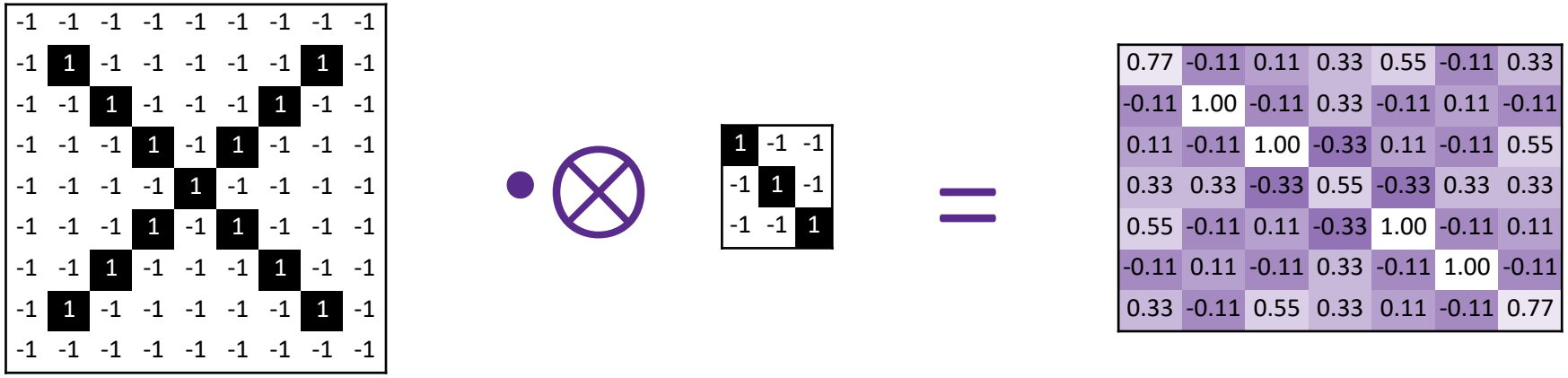
CONVOLUTION

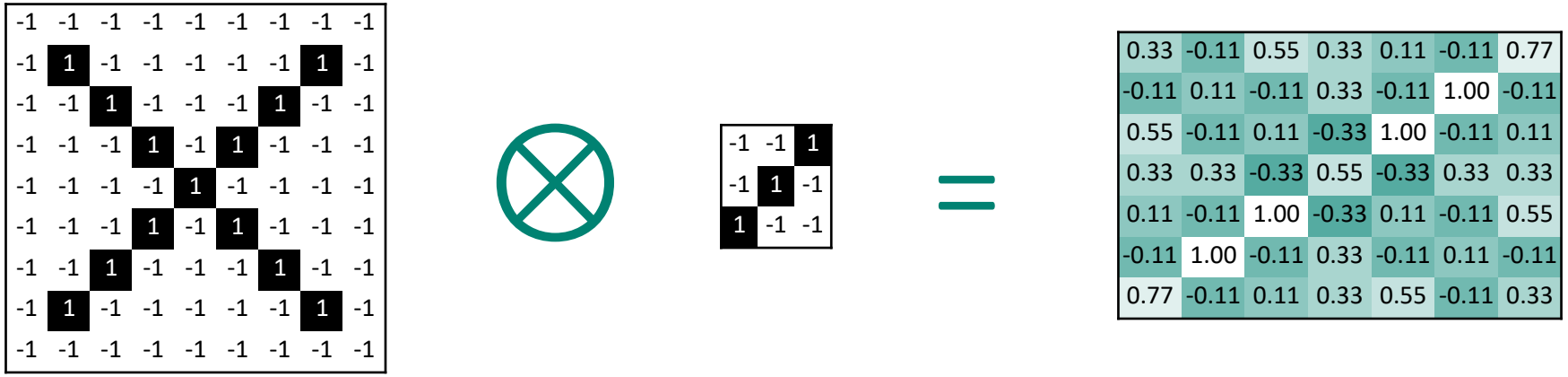


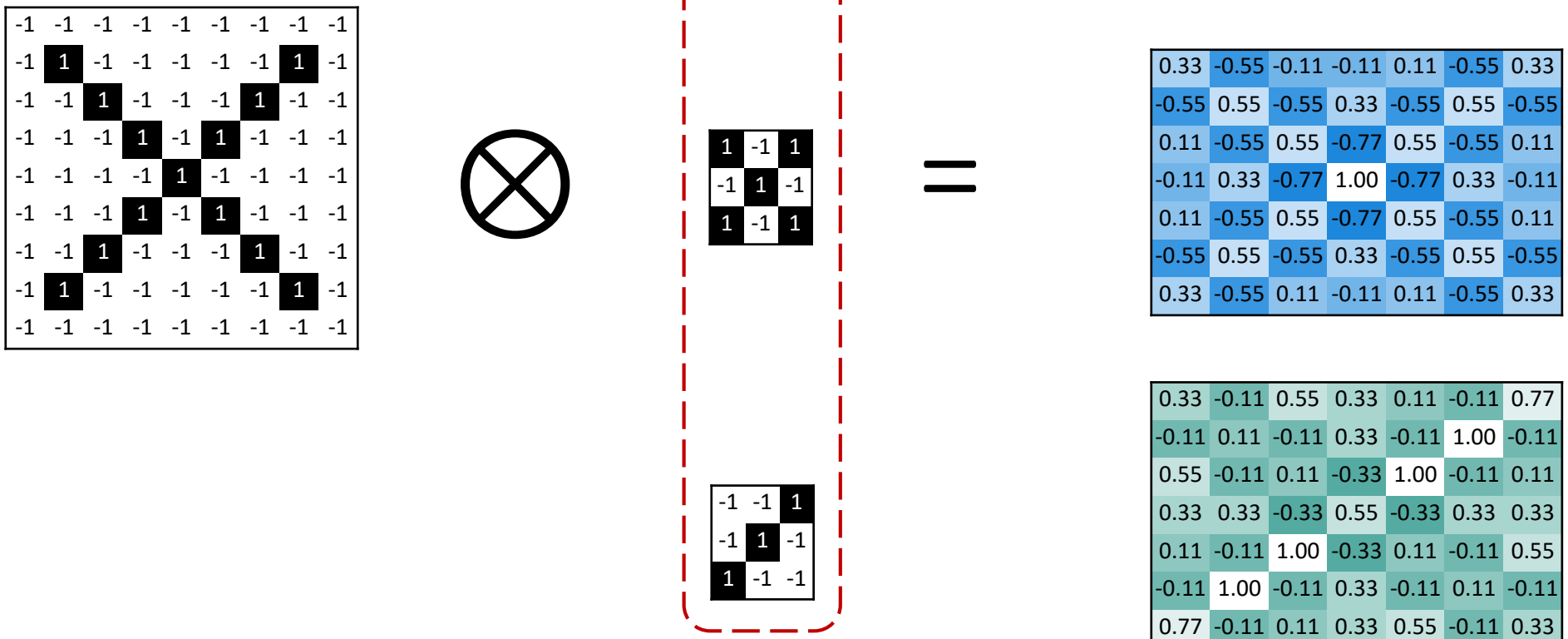
CONVOLUTION



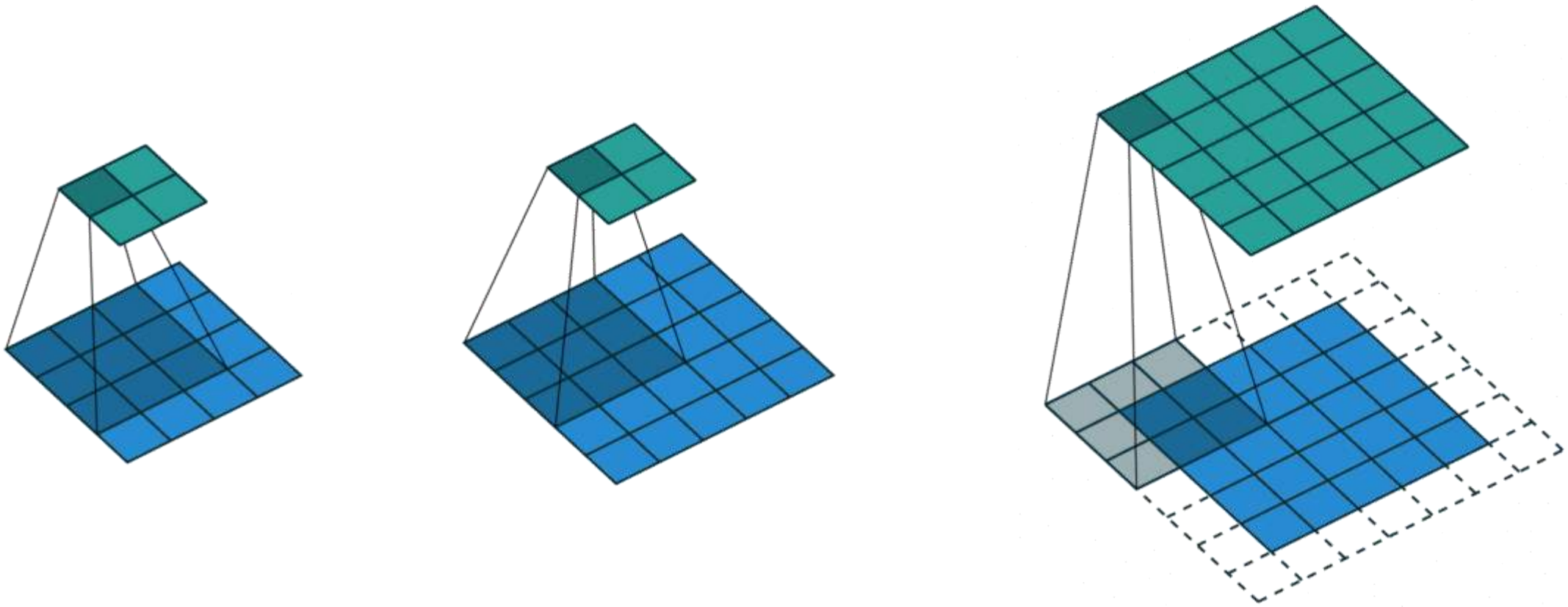
CONVOLUTION





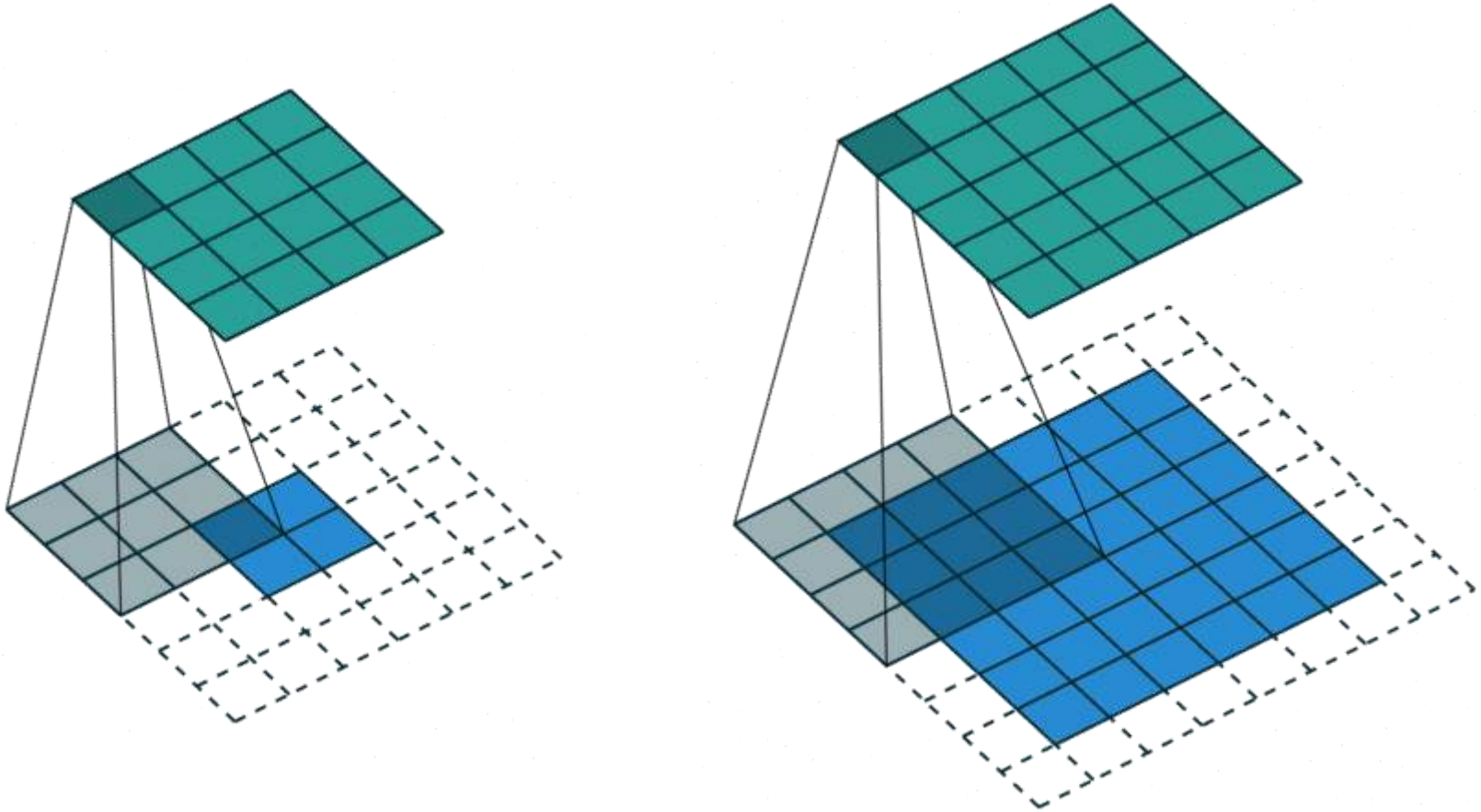

-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 -1 -1 1 -1 -1 -1 -1
-1 -1 -1 1 -1 1 -1 -1 -1
-1 -1 1 -1 -1 -1 1 -1 -1
-1 1 -1 -1 -1 -1 -1 1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
1 -1 -1
-1 1 -1
-1 -1 1
1 -1 1
-1 1 -1
1 -1 1
-1 -1 1
-1 1 -1
1 -1 -1
0.77 -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



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

DECONVOLUTION – TRANSPOSED CONVOLUTION

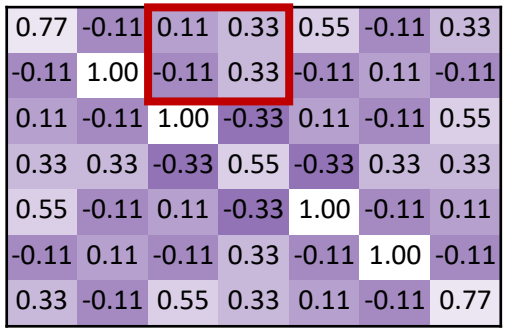


source: https://github.com/vdumoulin/conv_arithmetic

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

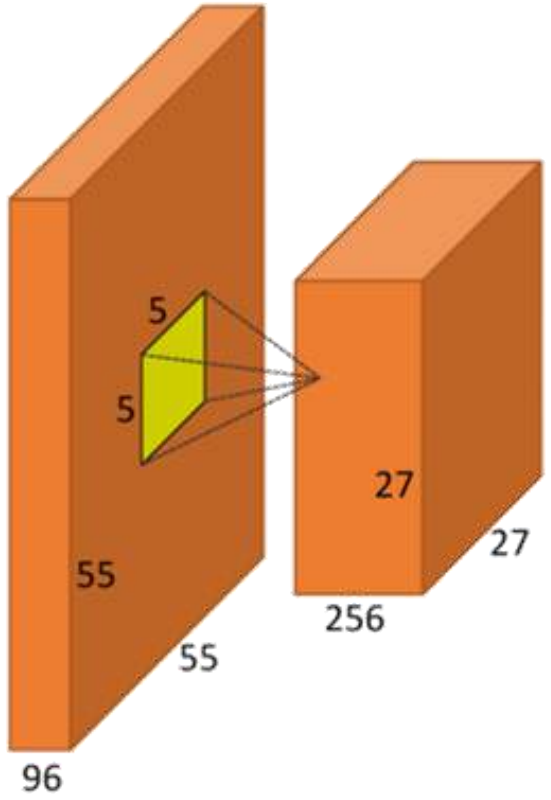
AVERAGE POOLING – STRIDE 2

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

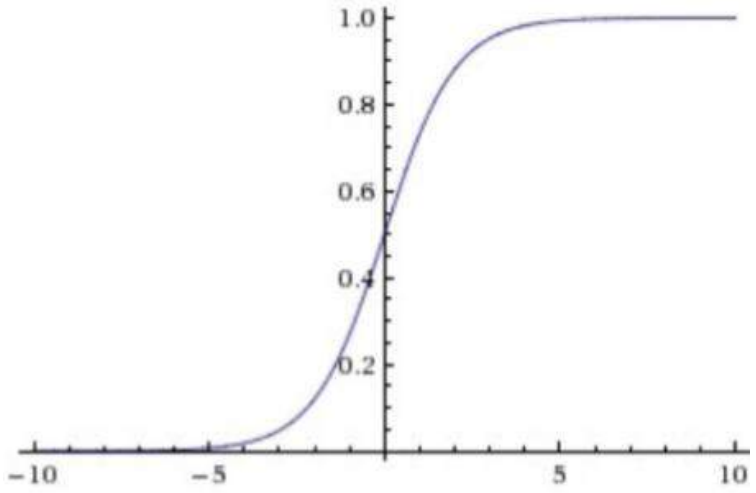
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

WHY POOLING?

ACTIVATION FUNCTIONS?



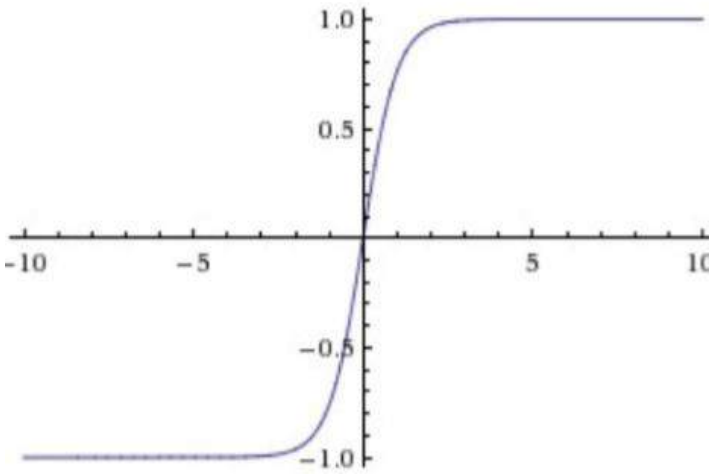
ACTIVATION FUNCTIONS?



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 :)

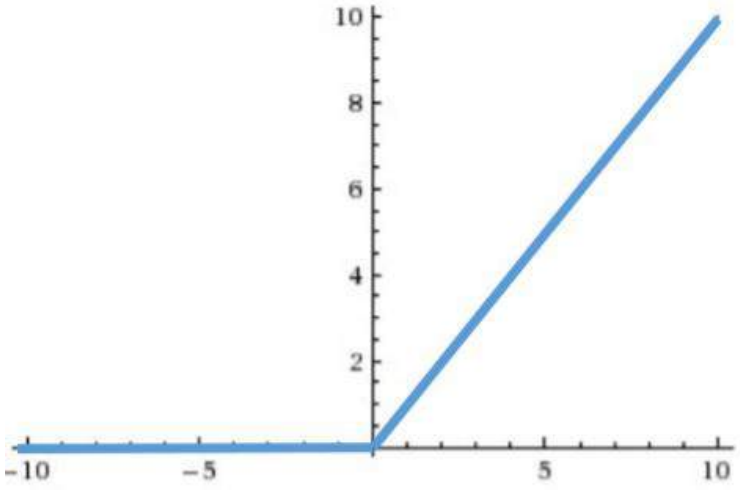
ACTIVATION FUNCTIONS?



HYPERBOLIC TANGENT (*tanh*):

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

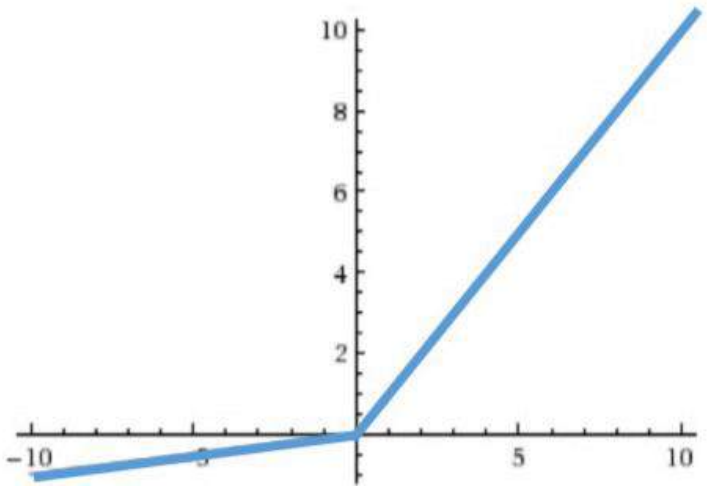
ACTIVATION FUNCTIONS?



RECTIFIED LINEAR UNITS (ReLU):

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

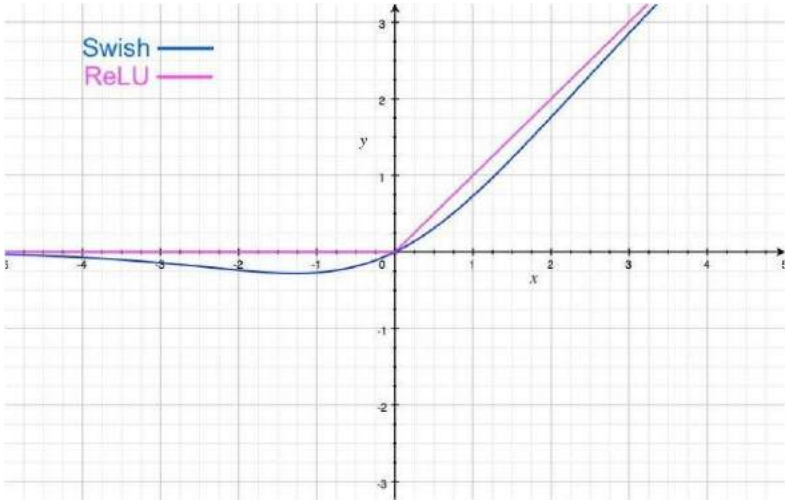
ACTIVATION FUNCTIONS?



LEAKY ReLU:

- More stable than traditional ReLu
- Particular case of *MaxOut*

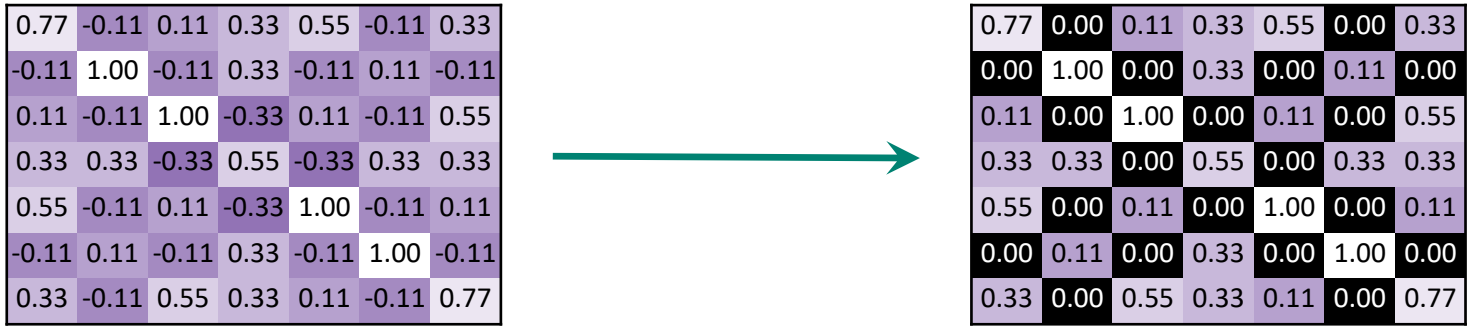
ACTIVATION FUNCTIONS?



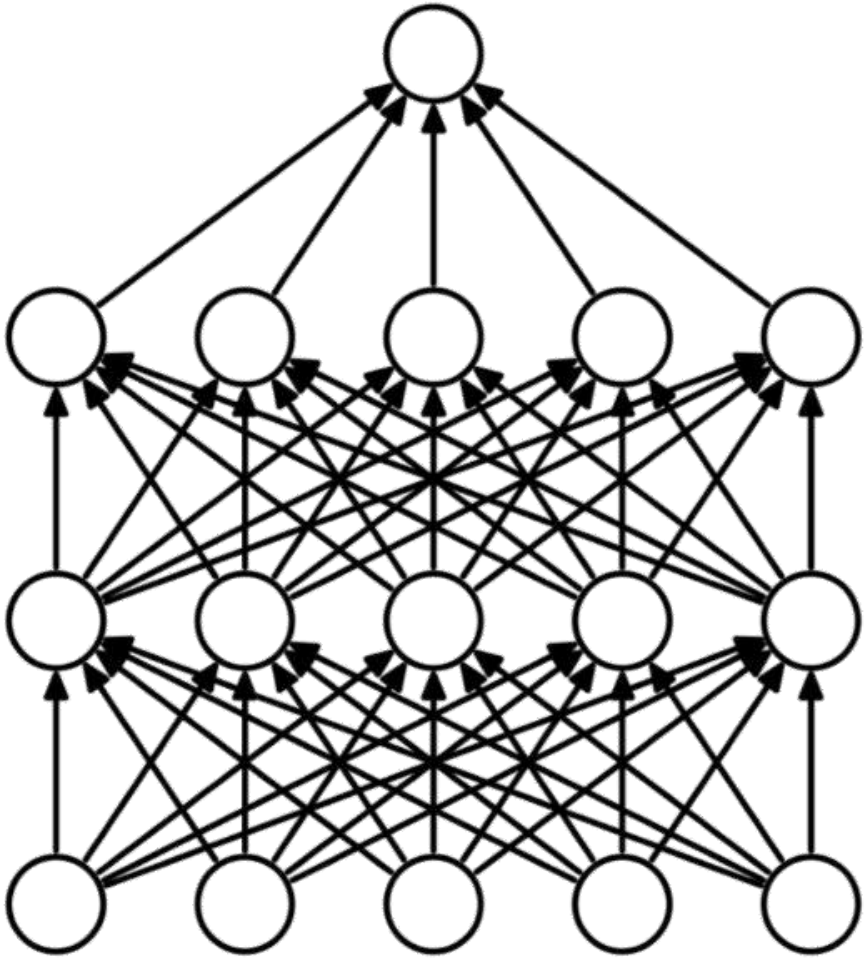
SWISH:

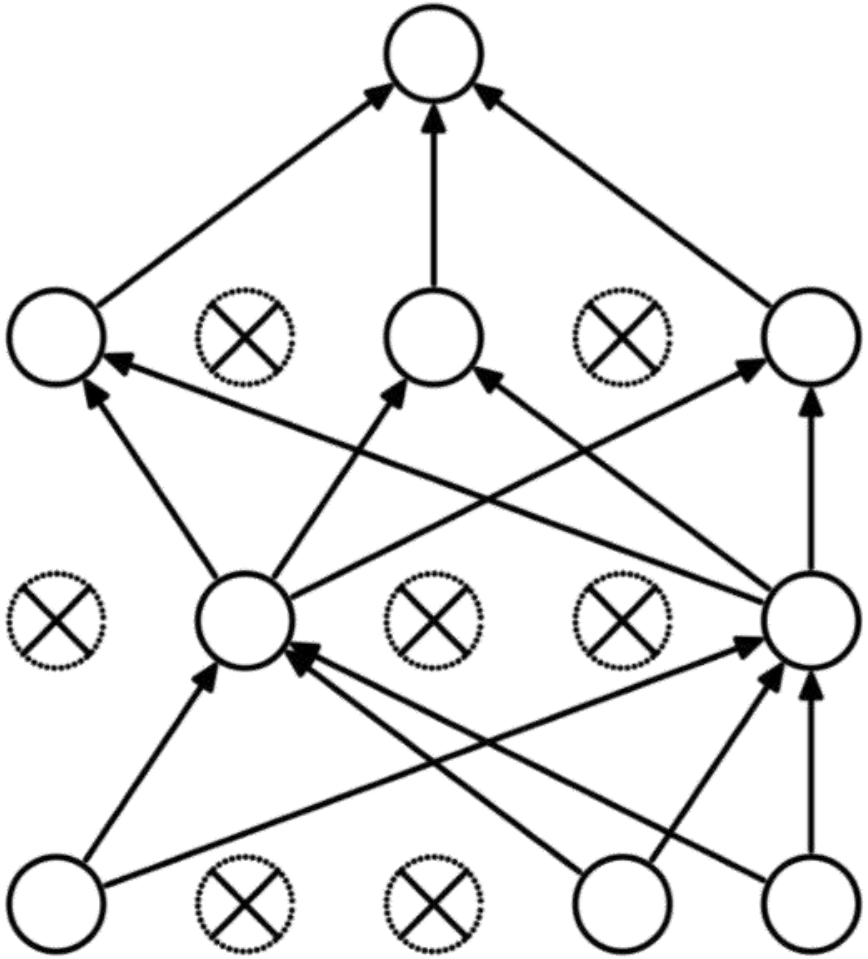
- Self-Gating activation function
- Better performance than ReLu

RECTIFIED LINEAR UNITS (ReLU)

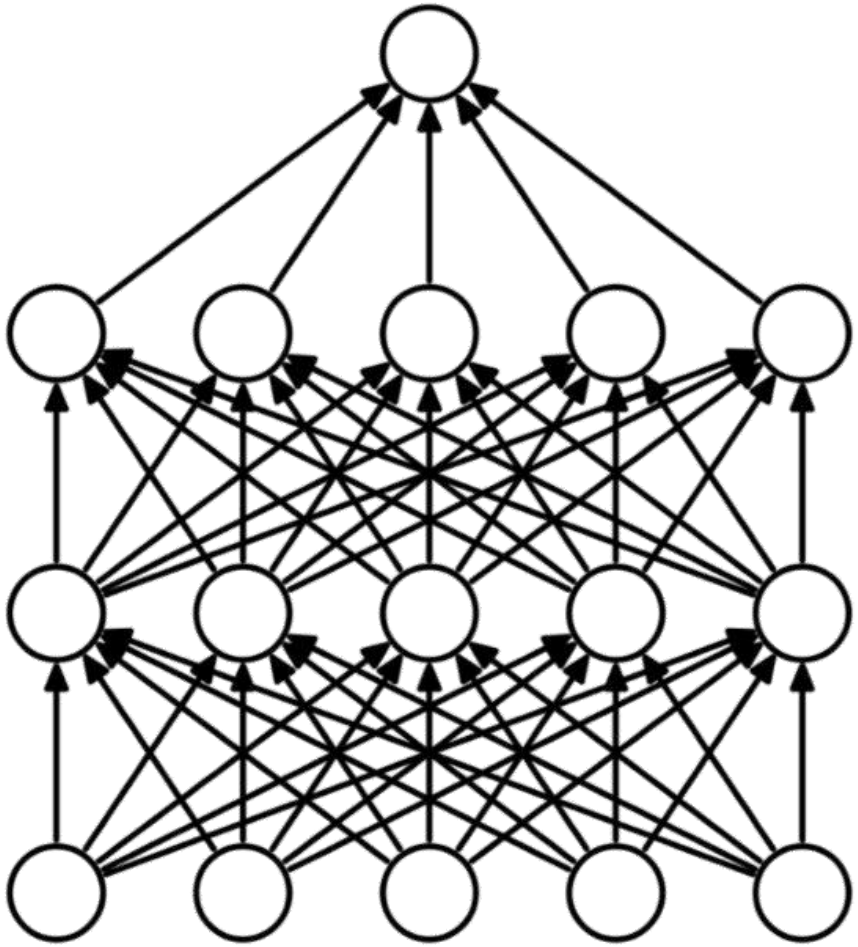


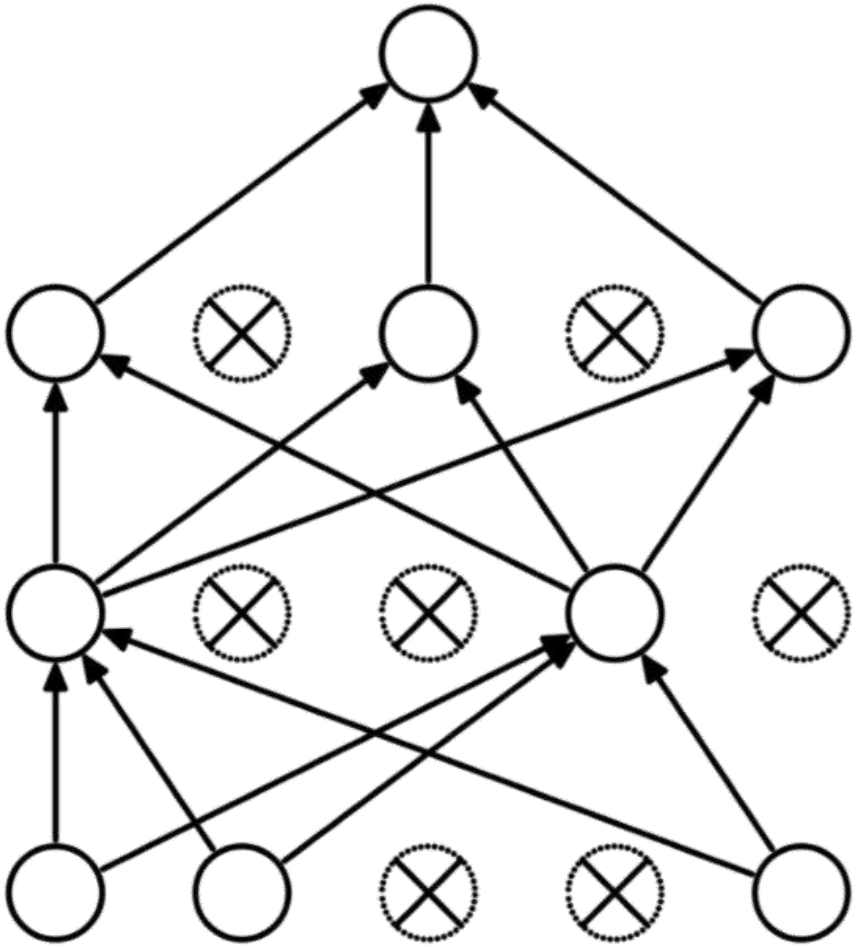
DROPOUT





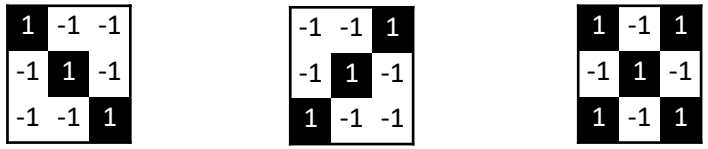
DROPOUT





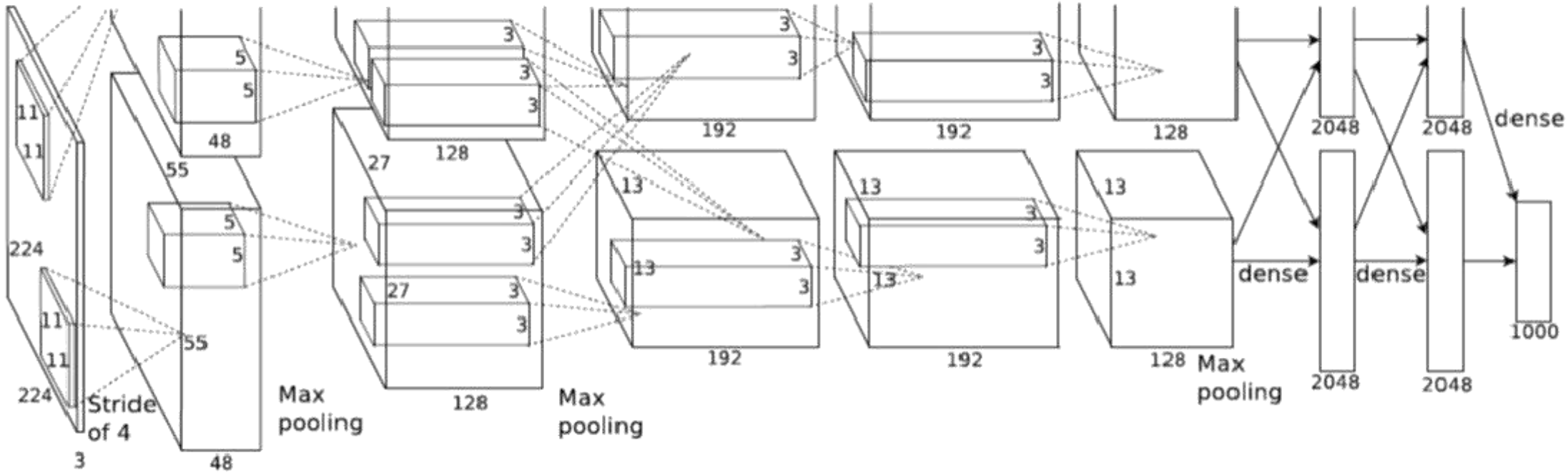
WHAT IS MISSING?

- Filter 1 Filter 2 Filter 3



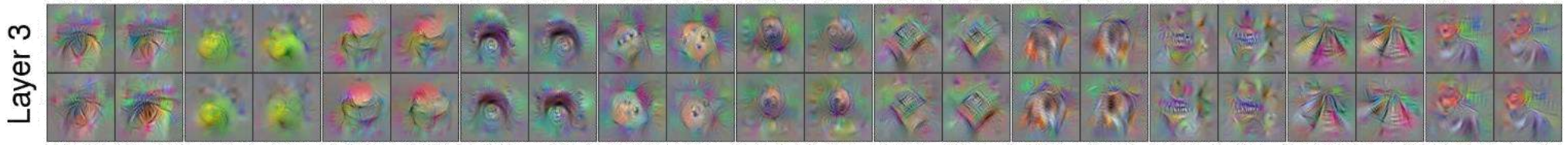
How have these filters been selected/created?

AlexNet

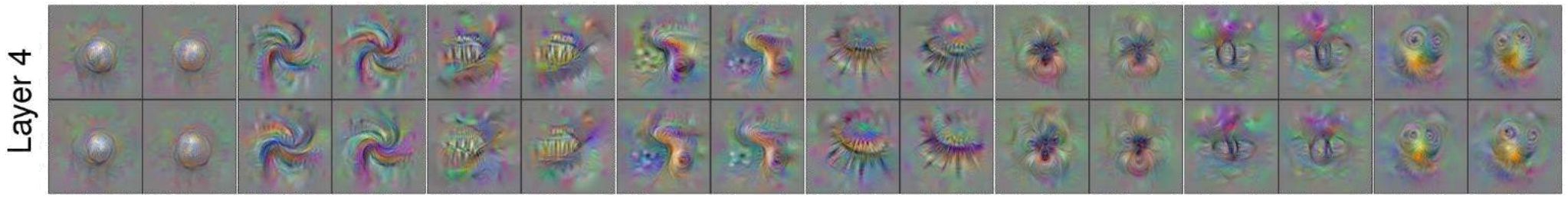

11
11
224
224
3
Stride
of 4
5
5
48
48
55
55
5
5
3
3
3
128
128
27
27
3
3
3
3
3
192
192
128
13
13
13
13
13
13
3
3
3
3
3
3
192
192
128
Max
pooling
Max
pooling
Max
pooling
dense
dense
dense
2048
2048
2048
2048
1000

HOW A CNN SEES

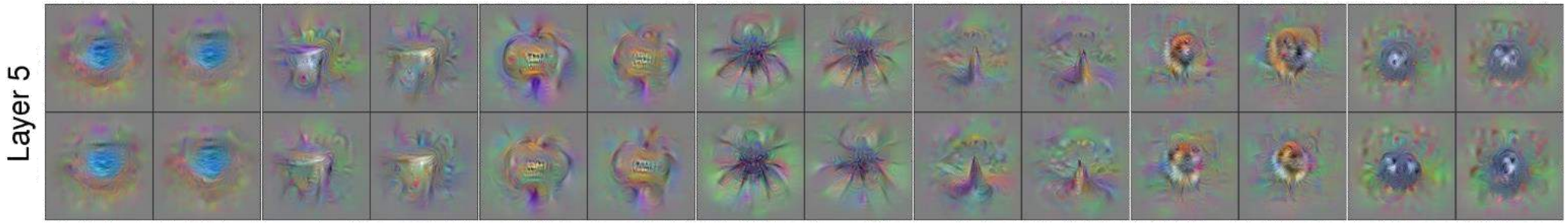
HOW A CNN SEES



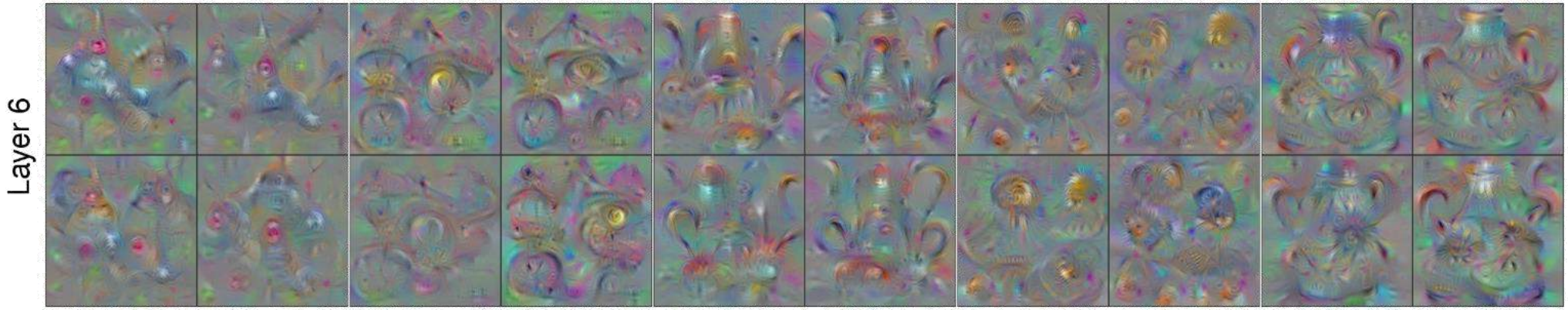
HOW A CNN SEES



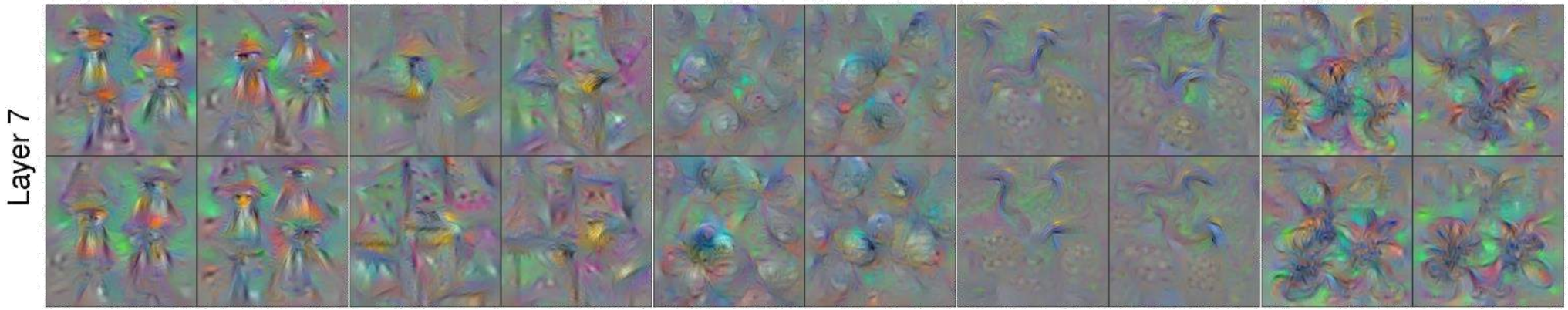
HOW A CNN SEES



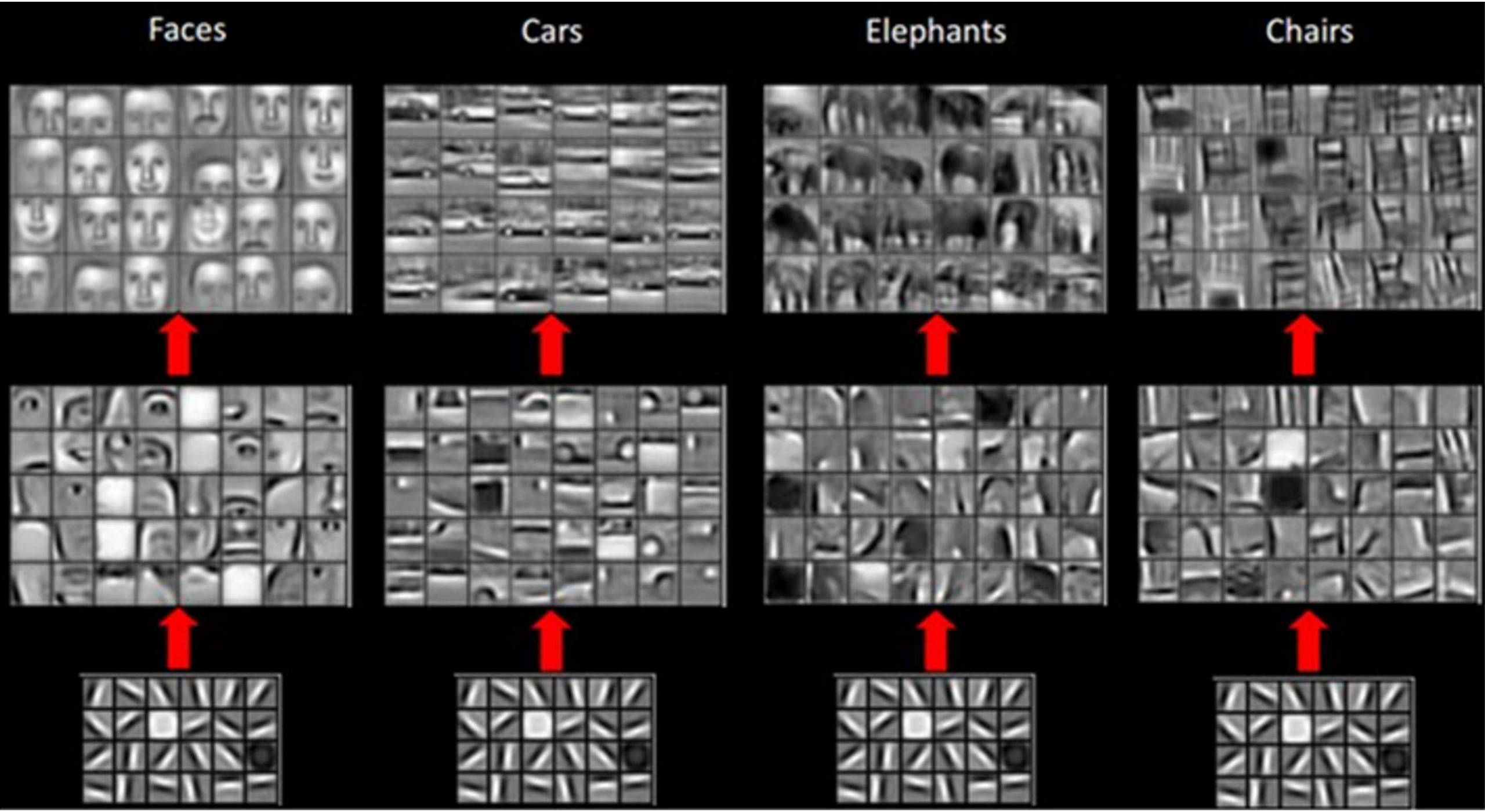
HOW A CNN SEES



HOW A CNN SEES



HOW A CNN SEES


Faces
Cars
Elephants
Chairs

LAB 6

Deploy

Deploy

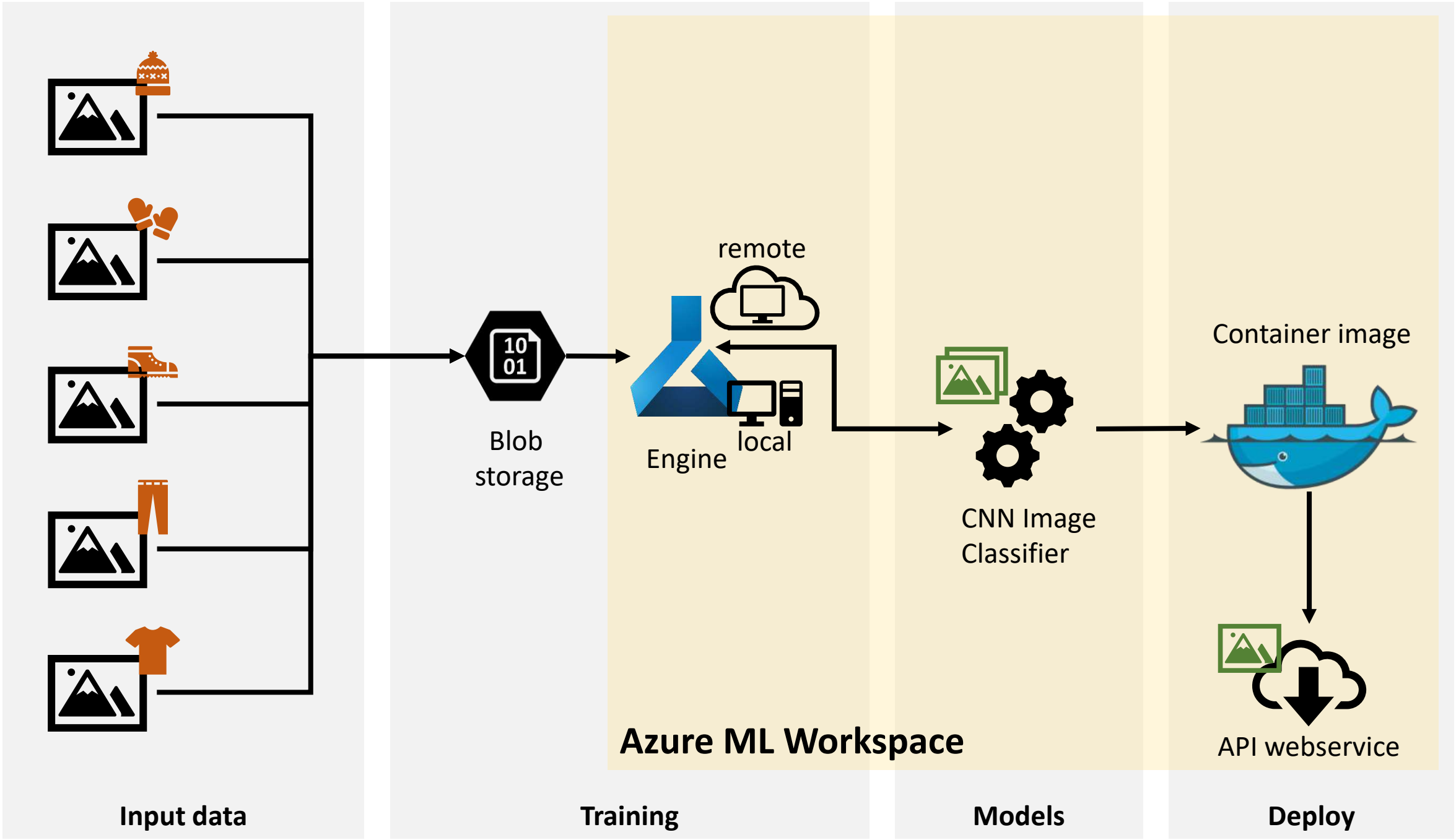
Use notebook *"Lab06_deploy.ipynb"*

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


Blob storage
remote
Engine
local
CNN Image Classifier
Container image
API webservice
Azure ML Workspace
Input data
Training
Models
Deploy

Some Advanced Ideas

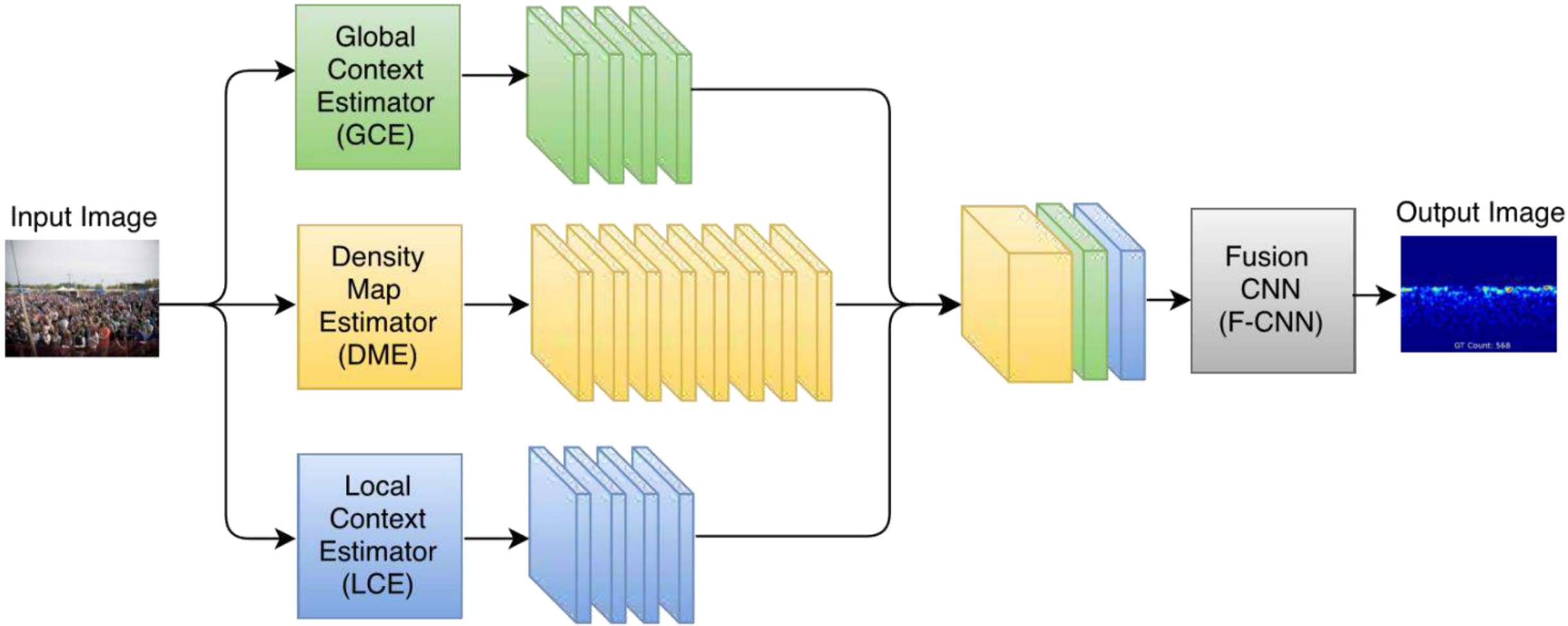
Counting people in a multitude

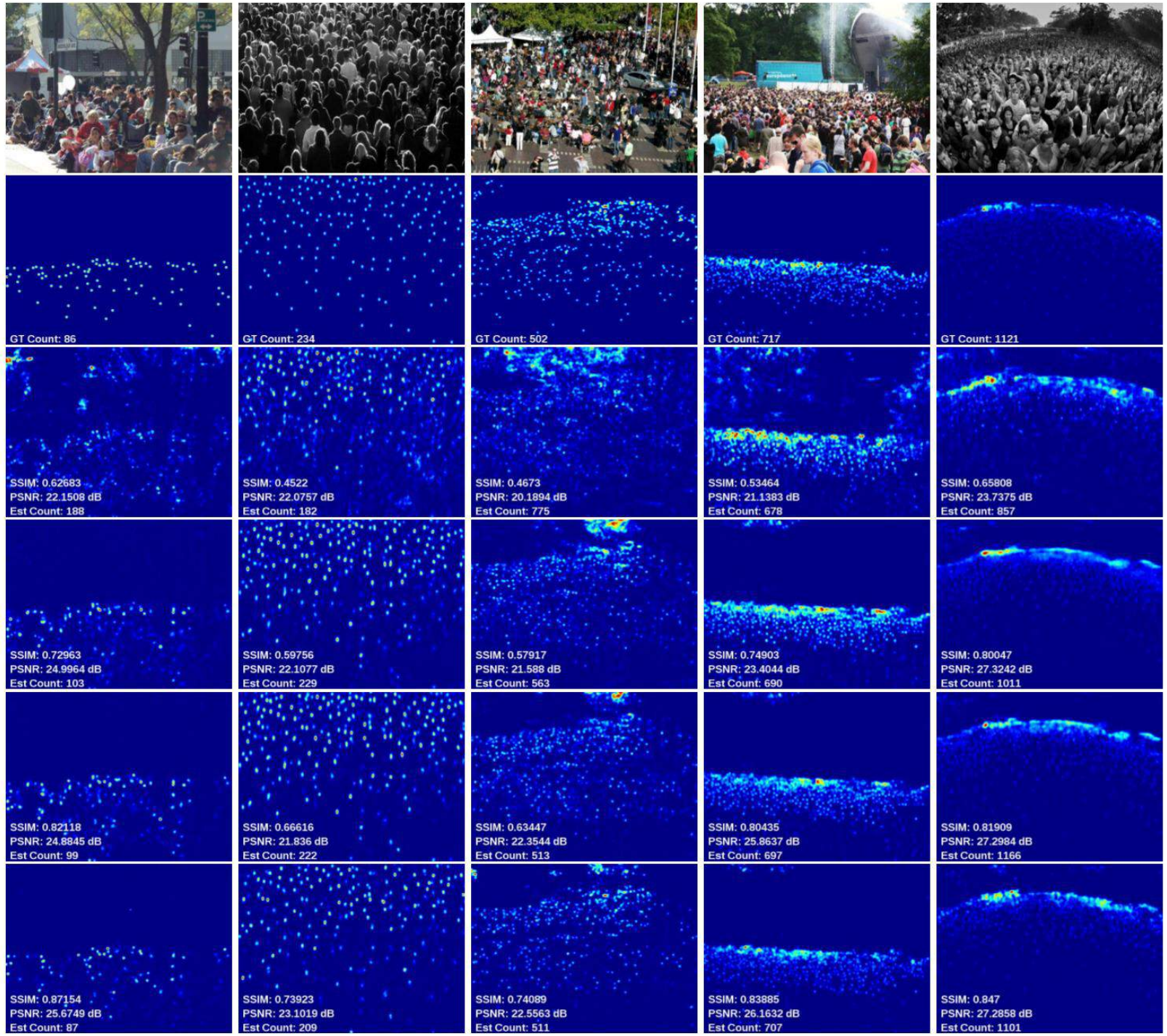
- CONTEXTUAL PYRAMID CNN (CP-CNN)

A new method named Contextual Pyramid CNN (CP-CNN) is proposed here to generate density maps and influx estimations, by explicitely 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

plain concepts


Input Image
Global Context Estimator (GCE)
Density Map Estimator (DME)
Local Context Estimator (LCE)
Fusion CNN (F-CNN)
Output Image
GT Count: 568

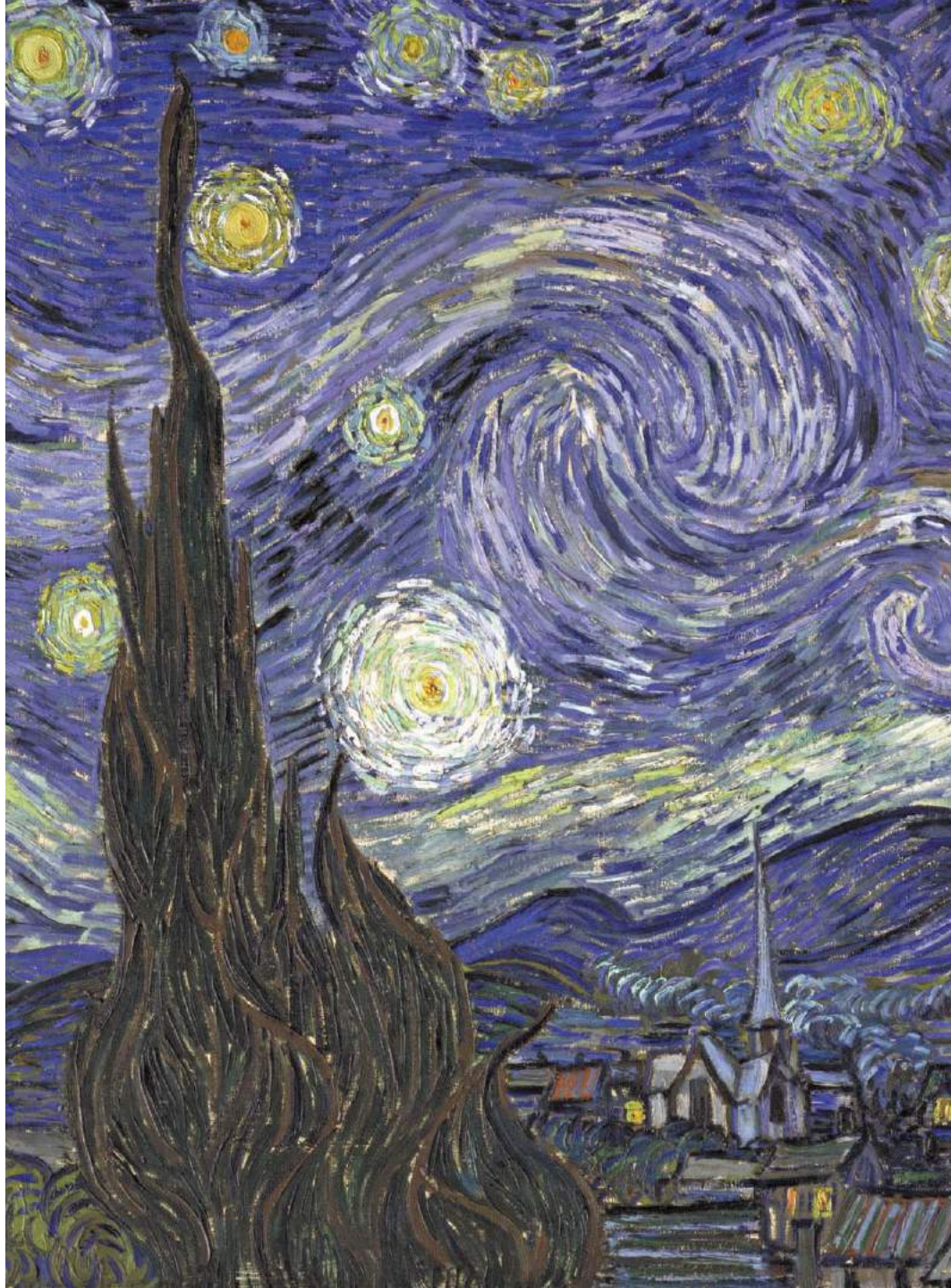

GT Count: 86
GT Count: 234
GT Count: 502
GT Count: 717
GT Count: 1121
SSIM: 0.62683
PSNR: 22.1508 dB
Est Count: 188
SSIM: 0.4522
PSNR: 22.0757 dB
Est Count: 182
SSIM: 0.4673
PSNR: 20.1894 dB
Est Count: 775
SSIM: 0.53464
PSNR: 21.1383 dB
Est Count: 678
SSIM: 0.65808
PSNR: 23.7375 dB
Est Count: 857
SSIM: 0.72963
PSNR: 24.9964 dB
Est Count: 103
SSIM: 0.59756
PSNR: 22.1077 dB
Est Count: 229
SSIM: 0.57917
PSNR: 21.588 dB
Est Count: 563
SSIM: 0.74903
PSNR: 23.4044 dB
Est Count: 690
SSIM: 0.80047
PSNR: 27.3242 dB
Est Count: 1011
SSIM: 0.82118
PSNR: 24.8845 dB
Est Count: 99
SSIM: 0.66616
PSNR: 21.836 dB
Est Count: 222
SSIM: 0.63447
PSNR: 22.3544 dB
Est Count: 513
SSIM: 0.80435
PSNR: 25.8637 dB
Est Count: 697
SSIM: 0.81909
PSNR: 27.2984 dB
Est Count: 1166
SSIM: 0.87154
PSNR: 25.6749 dB
Est Count: 87
SSIM: 0.73923
PSNR: 23.1019 dB
Est Count: 209
SSIM: 0.74089
PSNR: 22.5563 dB
Est Count: 511
SSIM: 0.83885
PSNR: 26.1632 dB
Est Count: 707
SSIM: 0.847
PSNR: 27.2858 dB
Est Count: 1101

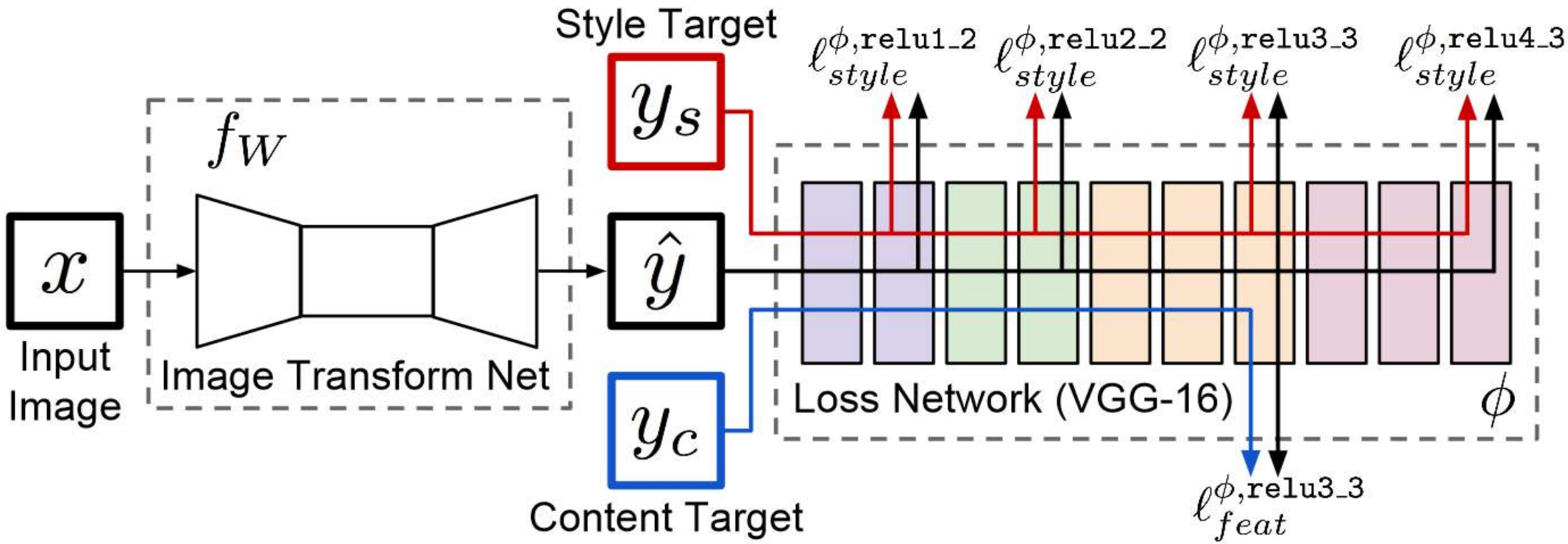
Transferring style between images

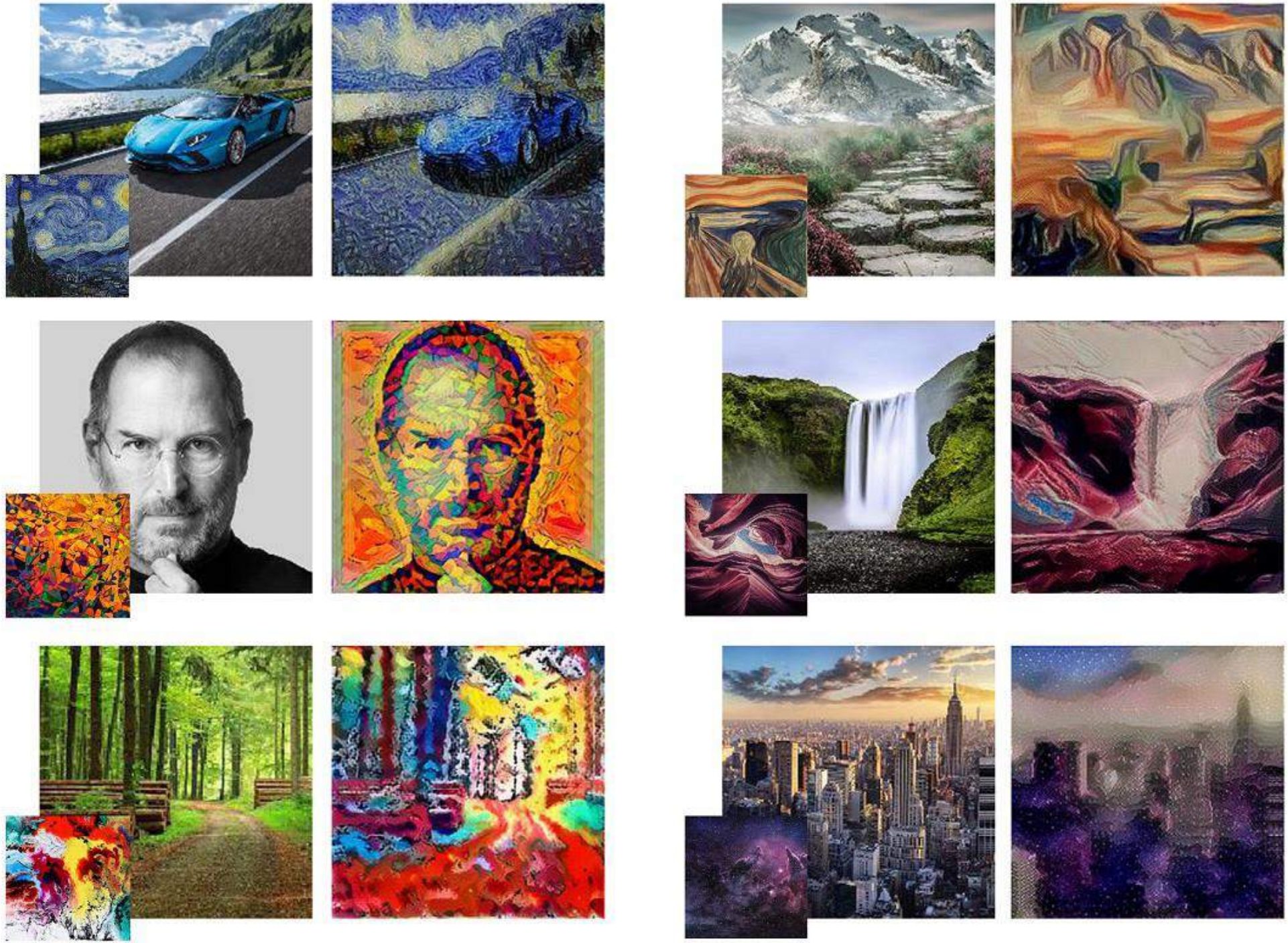
- CONVOLUTIONAL NEURAL NETWORKS

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




Style Target
y_s
$\ell^{\phi,\texttt{relu1\_2}}_{style}$
$\ell^{\phi,\texttt{relu2\_2}}_{style}$
$\ell^{\phi,\texttt{relu3\_3}}_{style}$
$\ell^{\phi,\texttt{relu4\_3}}_{style}$
f_W
x
Input Image
Image Transform Net
$\hat{y}$
Loss Network (VGG-16)
ϕ
y_c
Content Target
$\ell^{\phi,\texttt{relu3\_3}}_{feat}$



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.

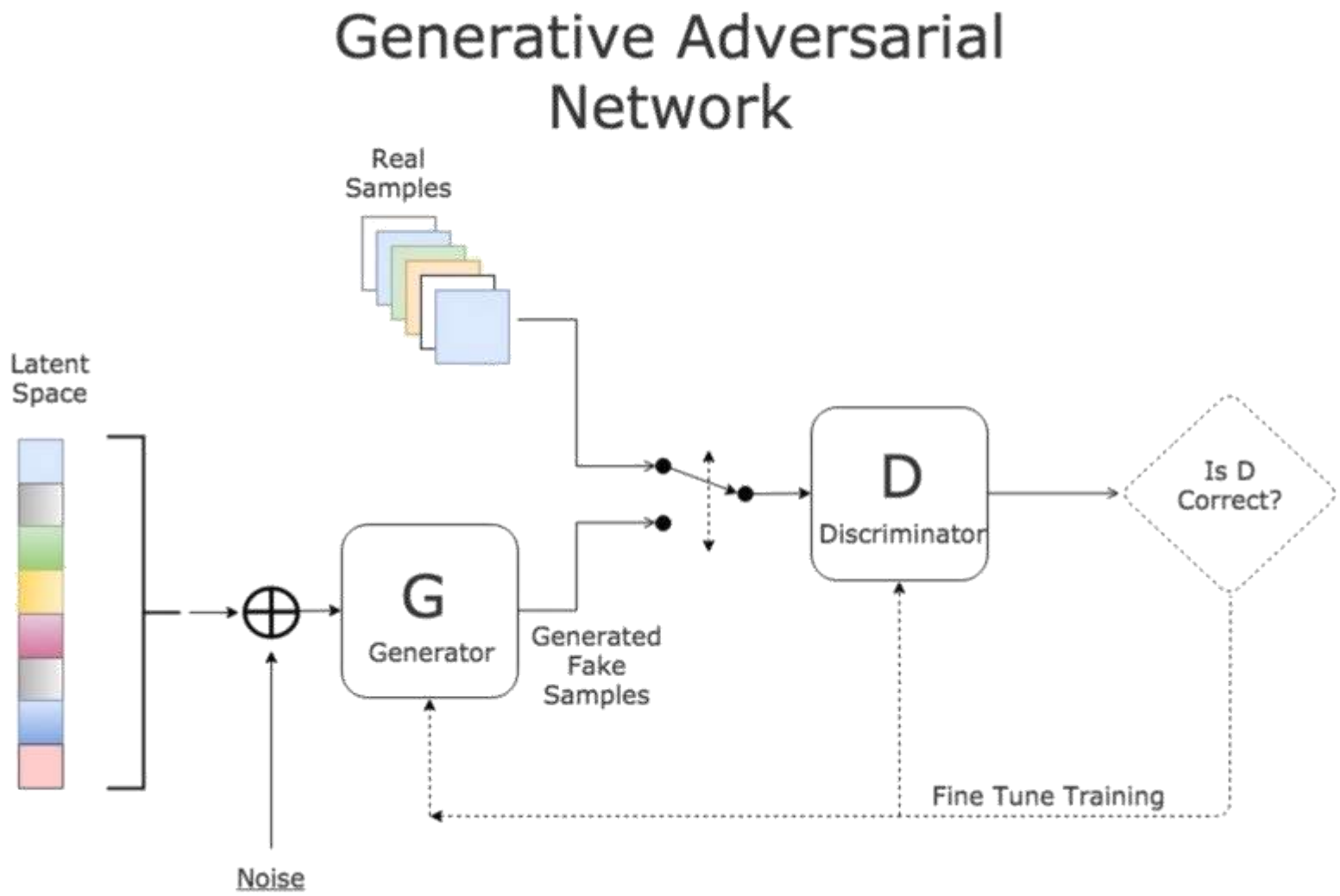
Jaime Deverall, Jiwoo Lee, Miguel Ayala; Using Generative Adversarial Networks to Design Shoes

plain concepts


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

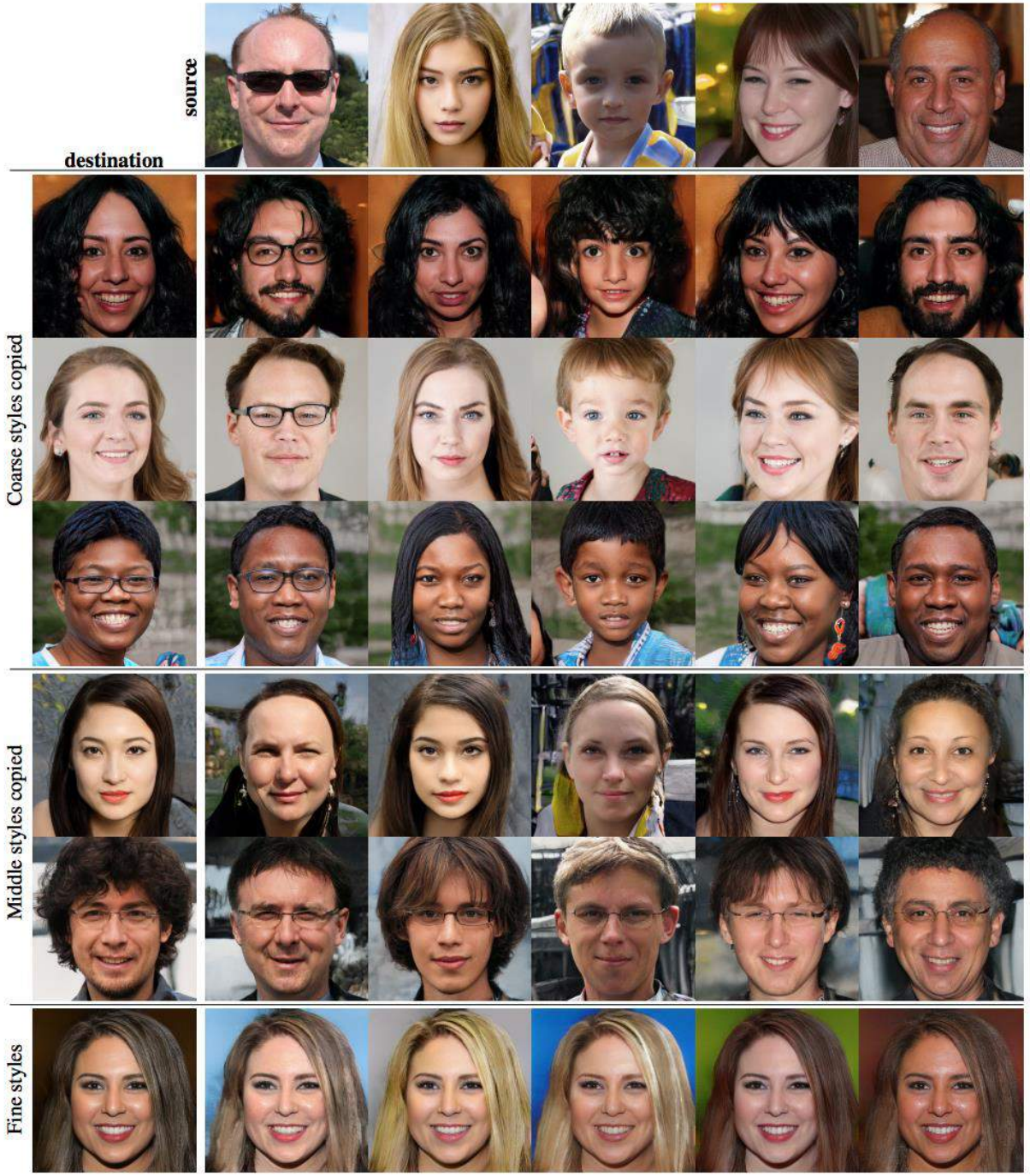
HOW DOES THIS WORK?



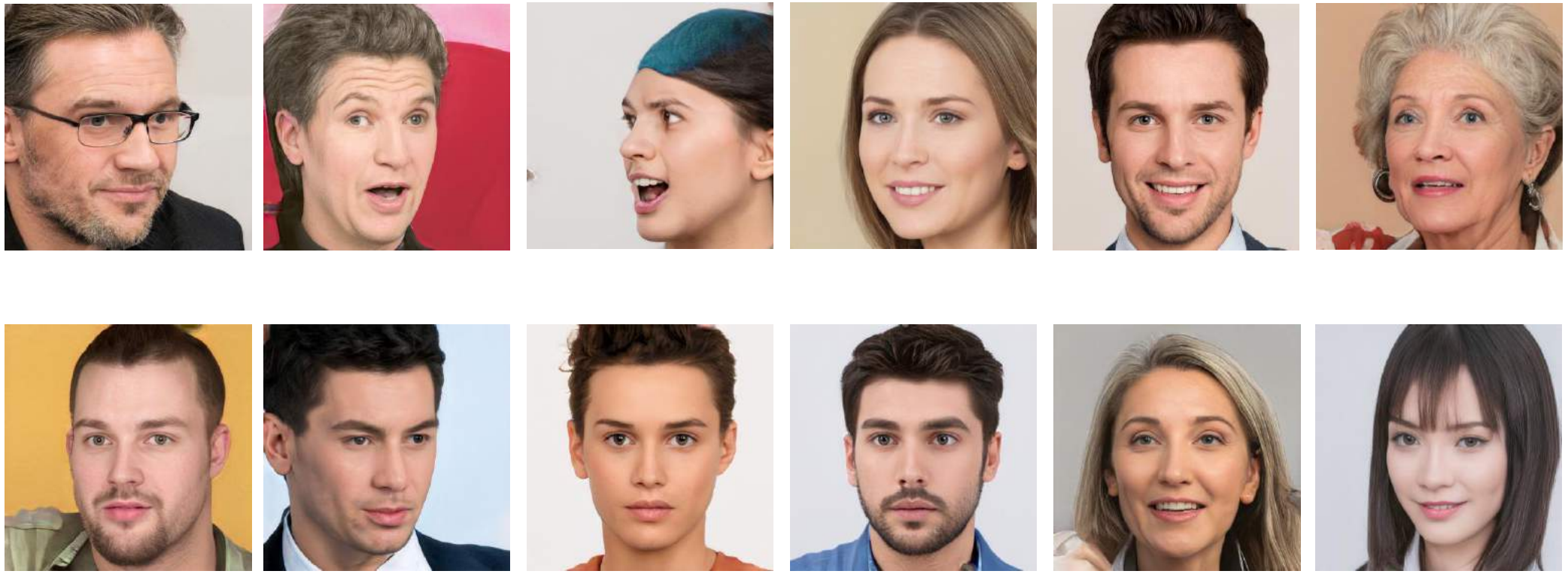

NVIDIA GauGAN Beta Demo

StyleGAN =
GAN + Style Transfer

by NVlabs



Even create your own people



https://generated.photos/

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.

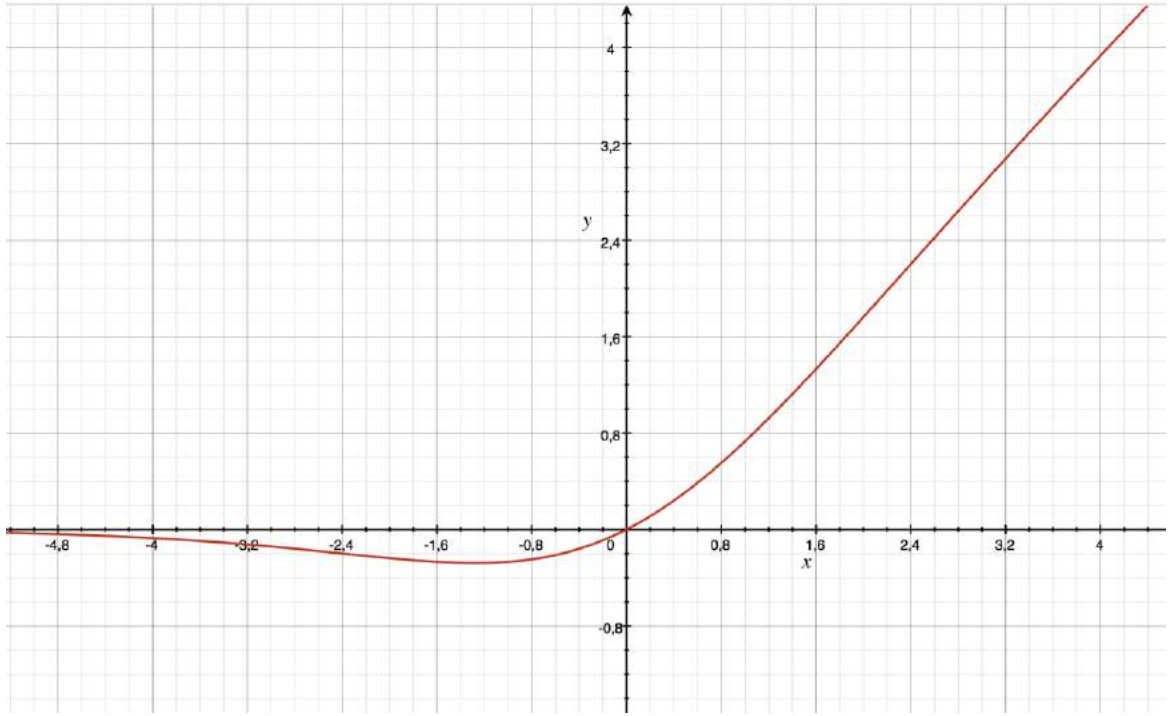
It was only a few minutes after the flick that my girlfriend noticed me looking at things in my pockets. I said I was just looking for something to read, and she said, "It's so bad, you could die!"

"What are you talking about," I said.

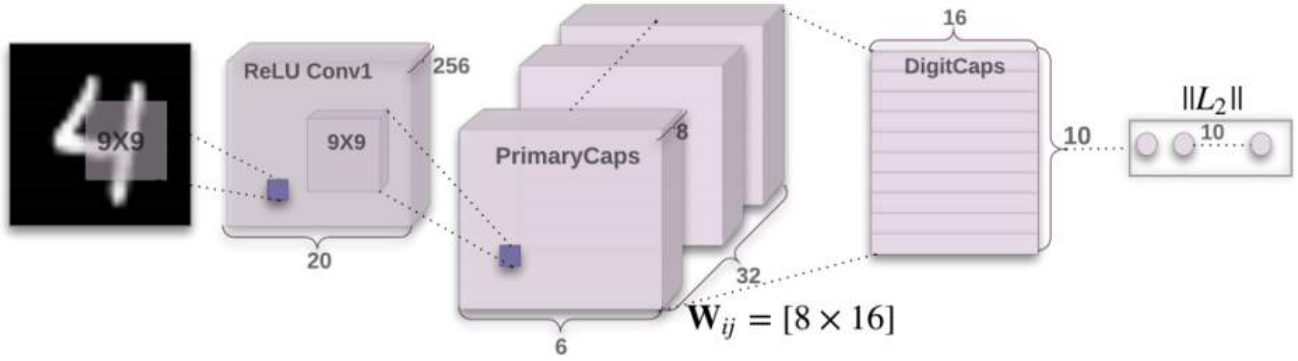
"Oh, what are you talking about? I'm going to die, what are you talking about?" she asked me.

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!


9X9
ReLU Conv1
256
9X9
20
PrimaryCaps
8
32
6
$\mathbf{W}_{ij} = [8 \times 16]$
16
DigitCaps
10
$\|L_2\|$
10

SOME INTERESTING PAPERS

- “ImageNet Classification with Deep Convolutional Neural Netowrks”
- [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