Technische Hochschule Rosenheim

Applications of & Introduction to Artificial Intelligence (A2I2)

Applications of
Image Classification and Object Recognition

Technische Hochschule Rosenheim

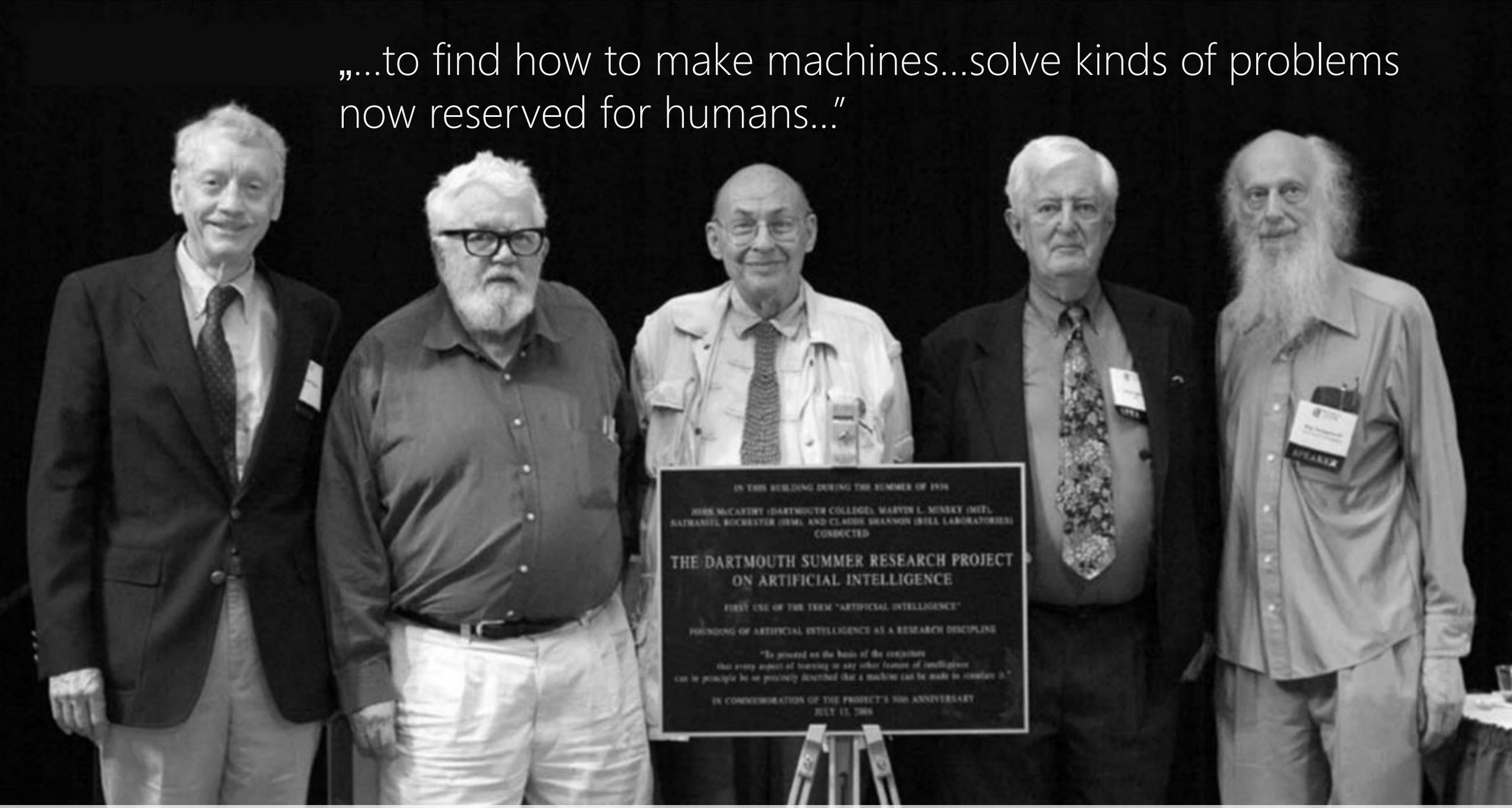
Sommer 2020

Prof. Dr. M. Tilly

Agenda

- Analyse Contents of images with Azure, Google and AWS
- Services for custom vision models and exporting for various frameworks
- Detect and identify people, emotions and age
- Extract text, key-value pairs, and tables from documents
- Recognize digital ink and handwriting

„…to find how to make machines…solve kinds of problems now reserved for humans…"



Dartmouth, 1956

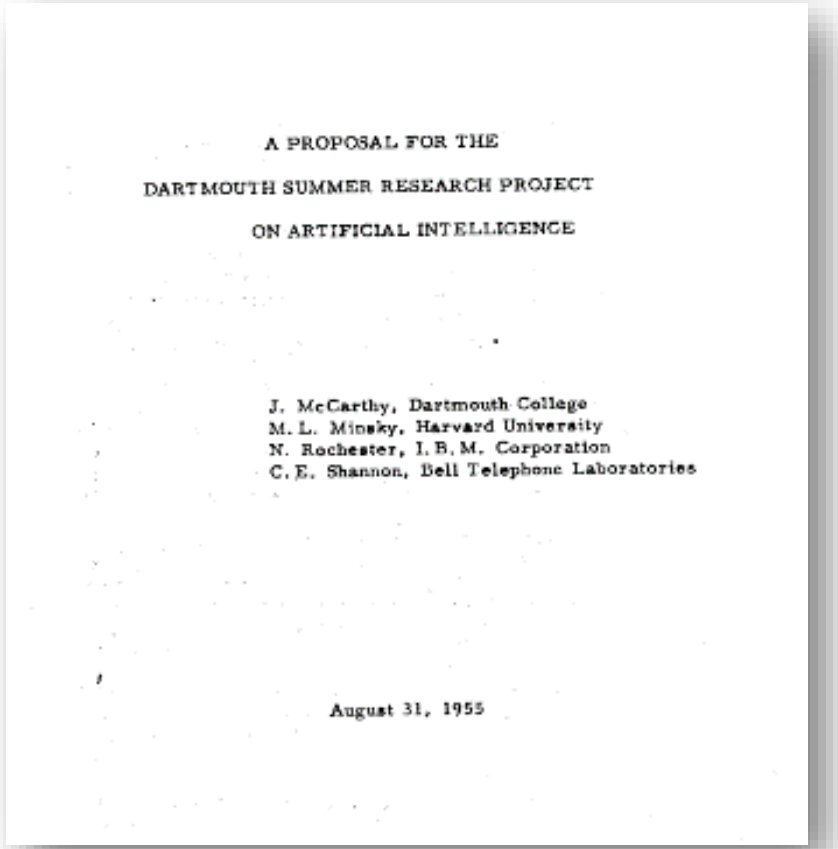
http://raysolomonoff.com/dartmouth/

The proposal on AI

On September 2, 1955, the project was formally proposed by McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon. The proposal is credited with introducing the term 'artificial intelligence'.

The Proposal states

> *"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."*



A PROPOSAL FOR THE

DARTMOUTH SUMMER RESEARCH PROJECT

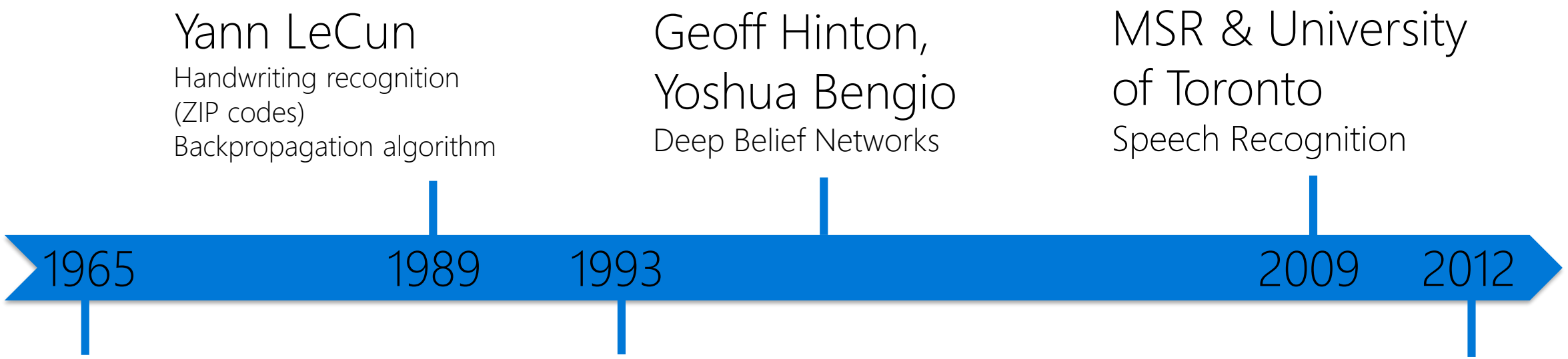
ON ARTIFICIAL INTELLIGENCE

J. McCarthy, Dartmouth College
M. L. Minsky, Harvard University
N. Rochester, I. B. M. Corporation
C. E. Shannon, Bell Telephone Laboratories

August 31, 1955

http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf

Journey of Deep Learning



1965 — Alexey Ivakhnenko
Supervised deep feedforward multilayer perceptrons

1989 — Yann LeCun
Handwriting recognition (ZIP codes)
Backpropagation algorithm

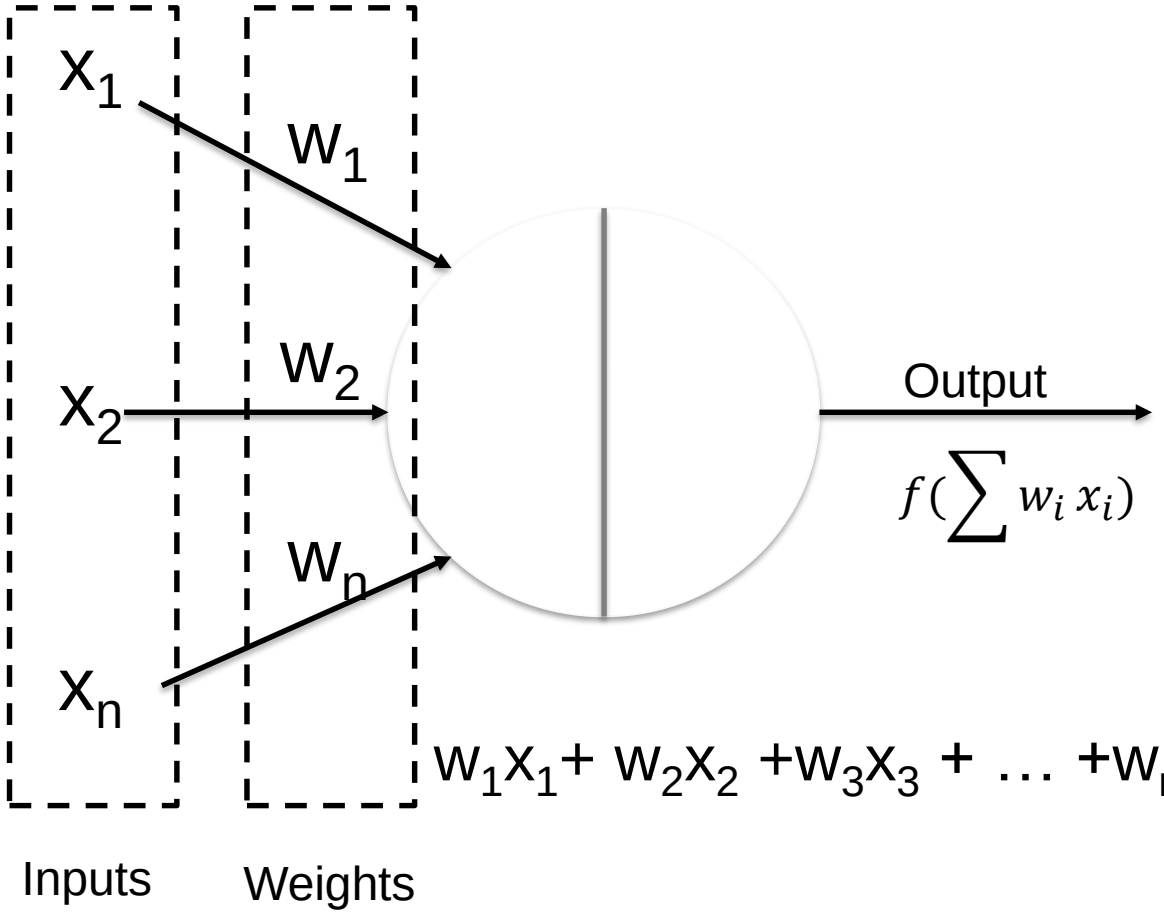
1993 — Jürgen Schmidhuber
Recurrent long and short term memories

2009 — Geoff Hinton, Yoshua Bengio
Deep Belief Networks

2012 — MSR & University of Toronto
Speech Recognition

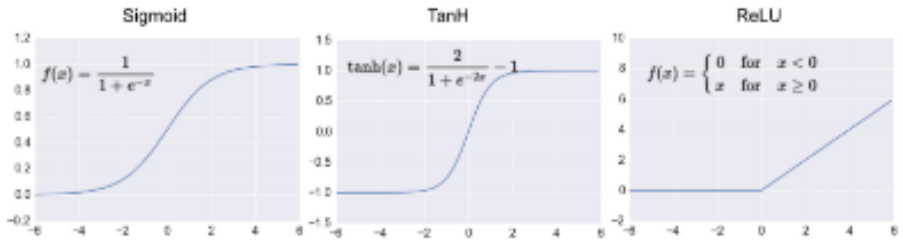
Andrew Ng
Deep Learning renaissance with cats

Towards *Deep* Learning?

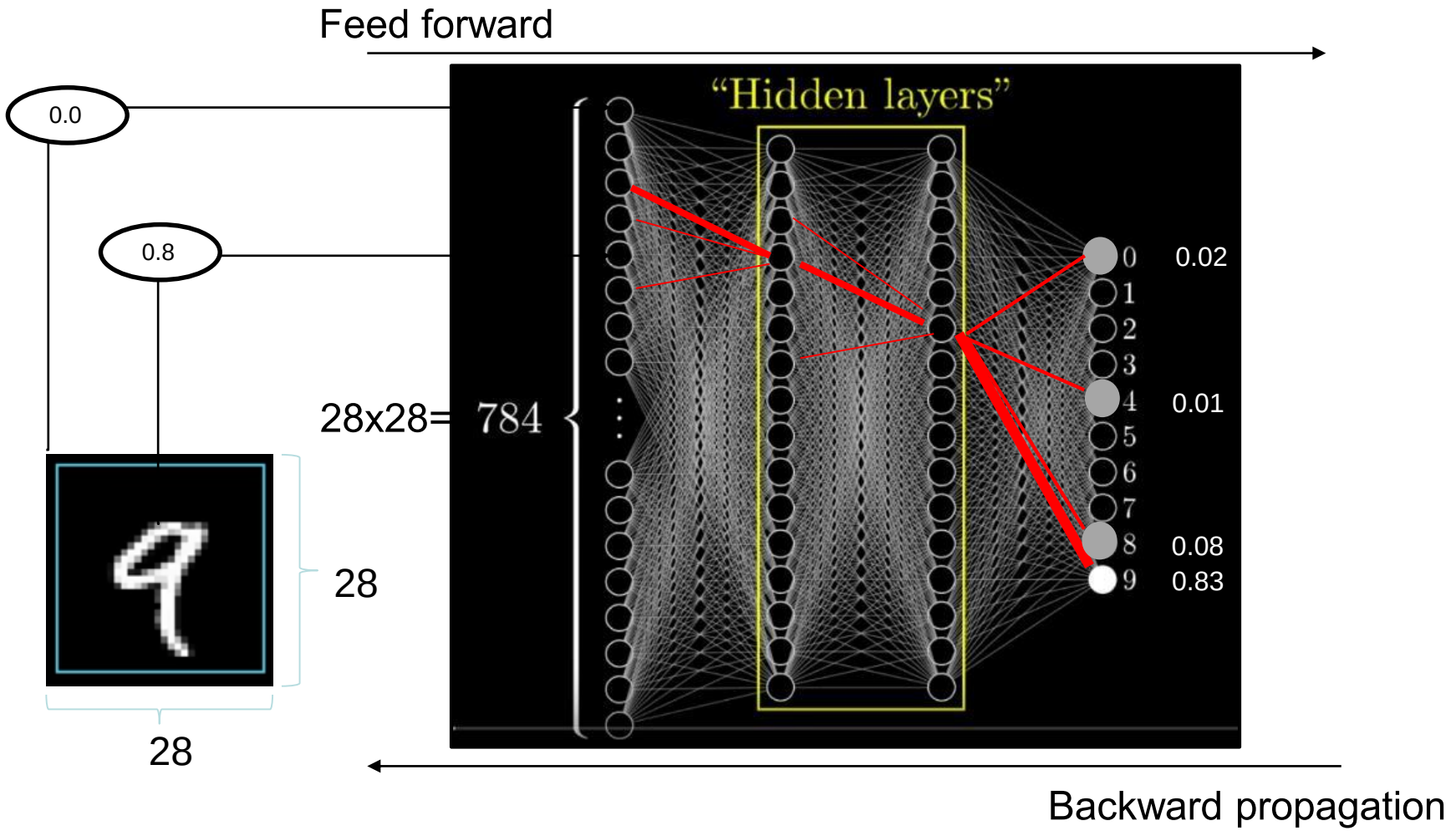


$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$

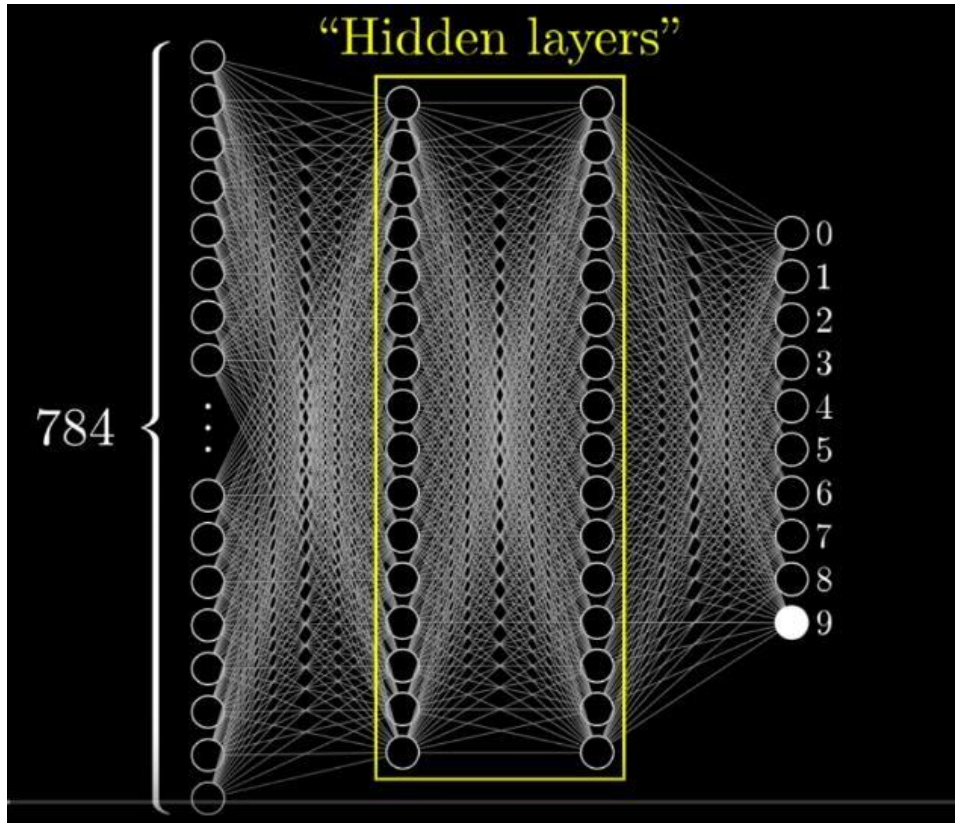
f = sigmoid(), tanh(), …



Deep Neural Networks (DNNs)



Feed forward

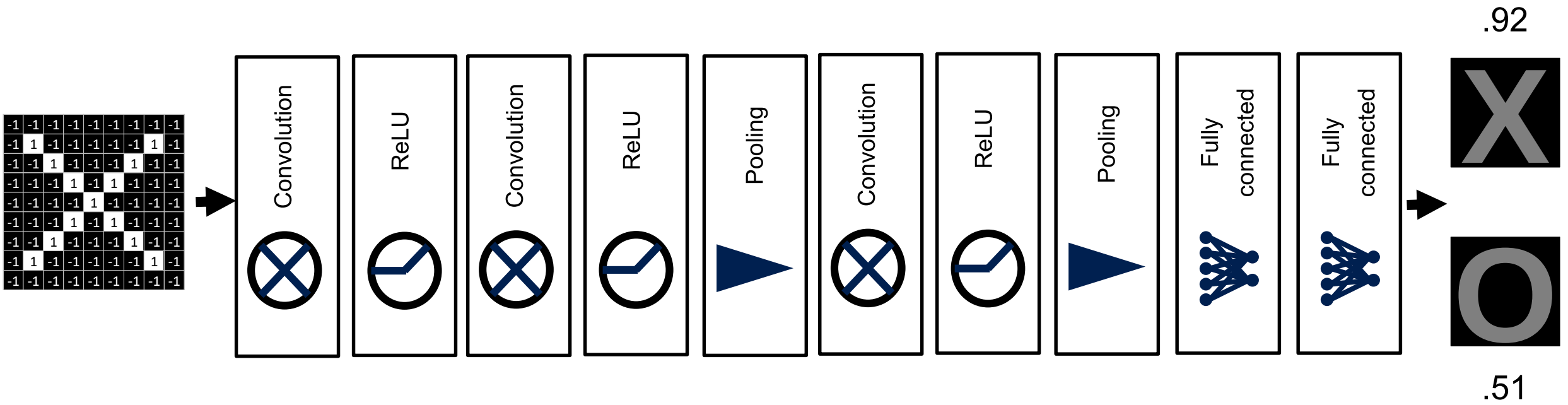


„dog“

„cat“

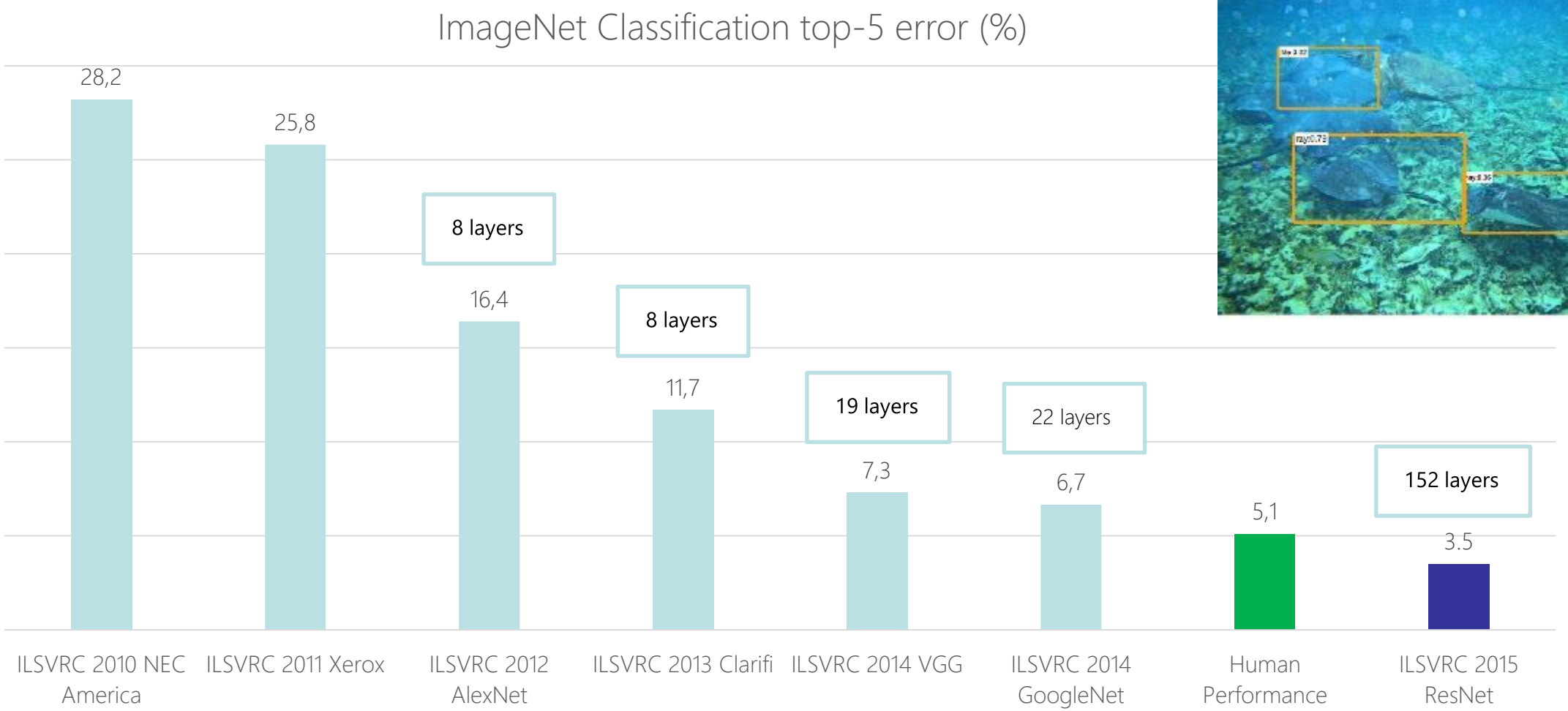
Backward propagation

Convolutional Neural Network



http://brohrer.github.io/how_convolutional_neural_networks_work.html

ImageNet Challenge



Computer Vision API

Content of Image:

[{ "name": "grass", "confidence": 0.9999992847442627 },
{ "name": "outdoor", "confidence": 0.9999072551727295 },
{ "name": "cow", "confidence": 0.99954754114151 },
{ "name": "field", "confidence": 0.9976195693016052 },
{ "name": "brown", "confidence": 0.988935649394989 },
{ "name": "animal", "confidence": 0.97904372215271 },
{ "name": "standing", "confidence": 0.9632768630981445 },
{ "name": "mammal", "confidence": 0.9366017580032349, "hint": "animal" },
{ "name": "wire", "confidence": 0.8946959376335144 },
{ "name": "green", "confidence": 0.8844101428985596 },
{ "name": "pasture", "confidence": 0.8332059383392334 },
{ "name": "bovine", "confidence": 0.5618471503257751, "hint": "animal" },
{ "name": "grassy", "confidence": 0.48627158999443054 },
{ "name": "lush", "confidence": 0.1874018907546997 },
{ "name": "staring", "confidence": 0.165890634059906 }]

Content of Image:

Describe

0.975 "a brown cow standing on top of a lush green field“
0.974 “a cow standing on top of a lush green field”
0.965 “a large brown cow standing on top of a lush green field”

What matters?

3 key elements:

- Algorithms are key: AlexNet, VGG, ResNet, …
- Data is important!
- Compute power

Where to find image data?

- **ImageNet** (http://www.image-net.org/): ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images.
  - Total number of images: ~1,500,000; each with multiple bounding boxes and respective class labels
- **COCO Dataset** (https://cocodataset.org): COCO is a large-scale object detection, segmentation, and captioning dataset.
  - **Number of Records:** 330K images, 80 object categories, 5 captions per image, 250,000 people with key points
- **Open Images Dataset V6 (https://storage.googleapis.com/openimages)**
  - **Number of Records:** 9,011,219 images with more than 5k labels
- **Visual** (https://visualqa.org/): VQA is a new dataset containing open-ended questions about images. These questions require an understanding of vision, language and commonsense knowledge to answer.
  - **Number of Records:** 265,016 images, at least 3 questions per image, 10 ground truth answers per question
- **Kaggle** (https://www.kaggle.com/datasets): Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 19,000 public datasets and 200,000 public notebooks to conquer any analysis in no time.

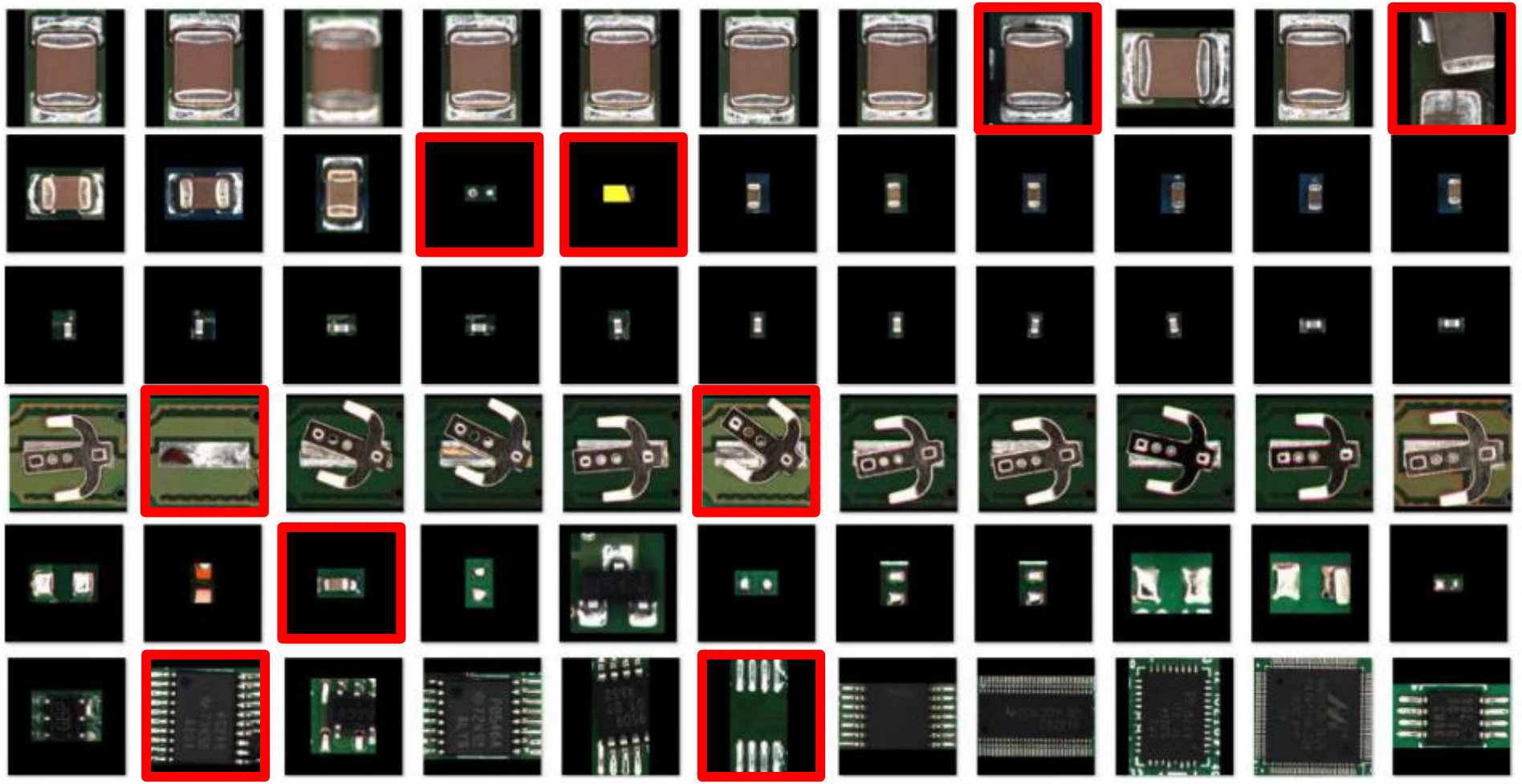
Computer Vision Service

- Microsoft Azure Cognitive Services (https://azure.microsoft.com/en-us/services/cognitive-services)
  - Computer Vision, Custom Vision, Face Recognition, Ink Recognizer (OCR), Form Recognizer
- AWS Rekognition (https://aws.amazon.com/rekognition)
  - Labels, Custom Labels, Content Moderation, Text Detection, Face Detection, Pathing
- Google Vision API (https://cloud.google.com/vision/)
  - Face Detection, Landmark Detection, Logo Detection, Label Detection, Text, Image Properties, Object Detection

Fun part ☺

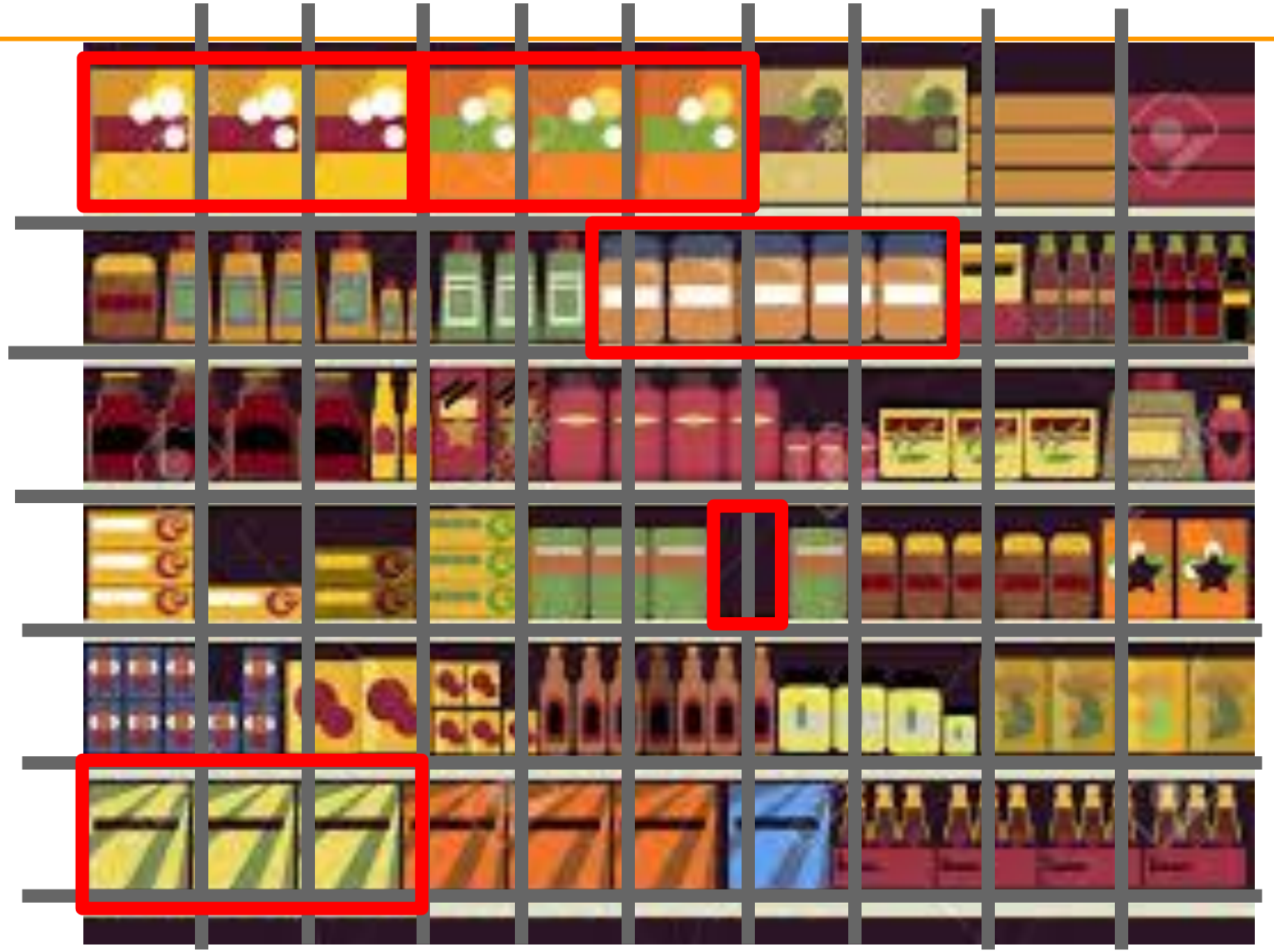

How-Old.net
How old do I look? #HowOldRobot
Search Faces…
Use this photo
Use your own photo
Microsoft

Example: Image Classification in Production

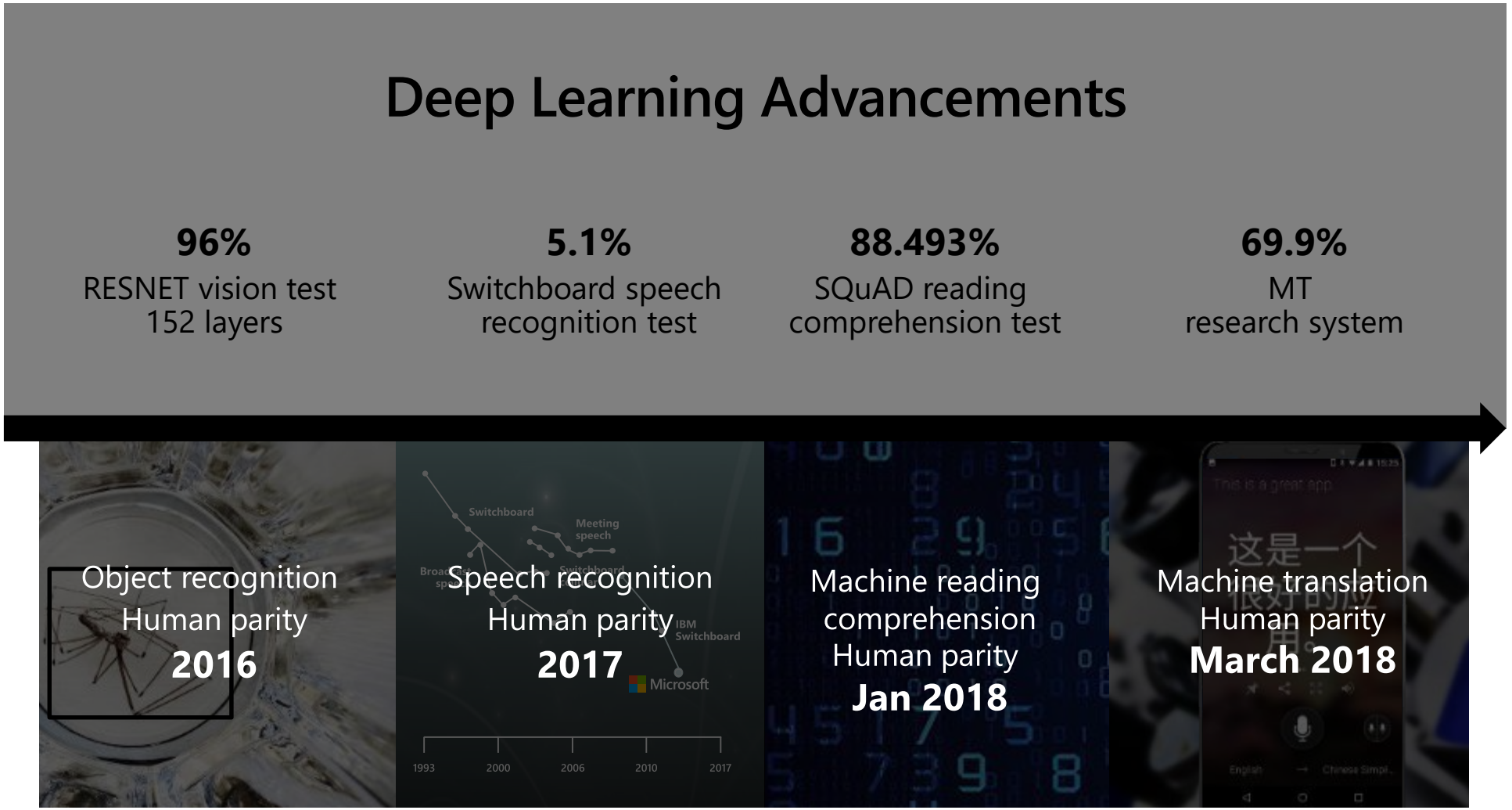


Example: Image Classification in Shop

- Grouping
- Placement compared to competitors
- Empty spot detection for refilling



Deep Learning Advancements



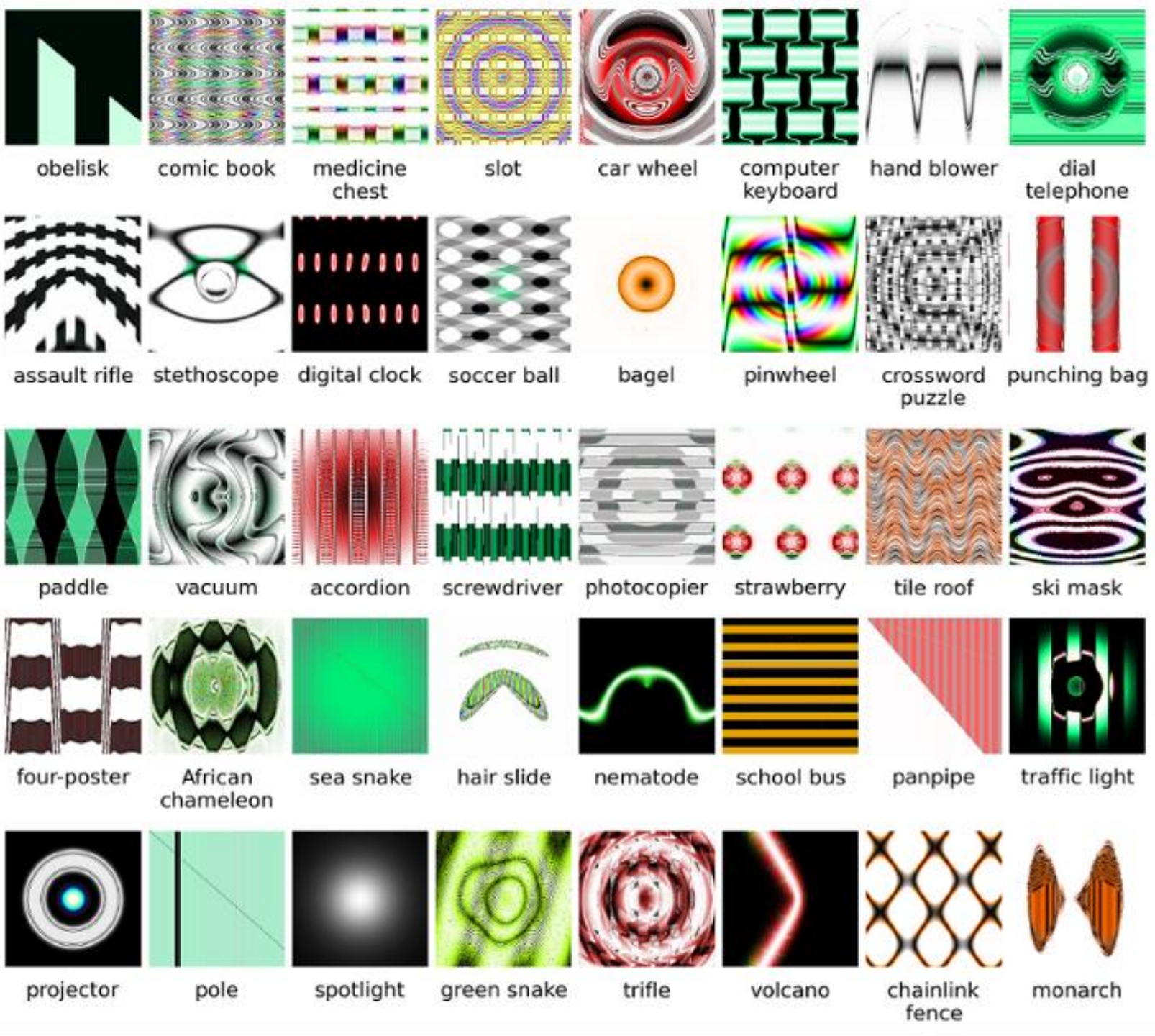
So what…

If you train with too many cats, everything looks like a cat

If there is something the computer has not seen, then the computer might think it is something else

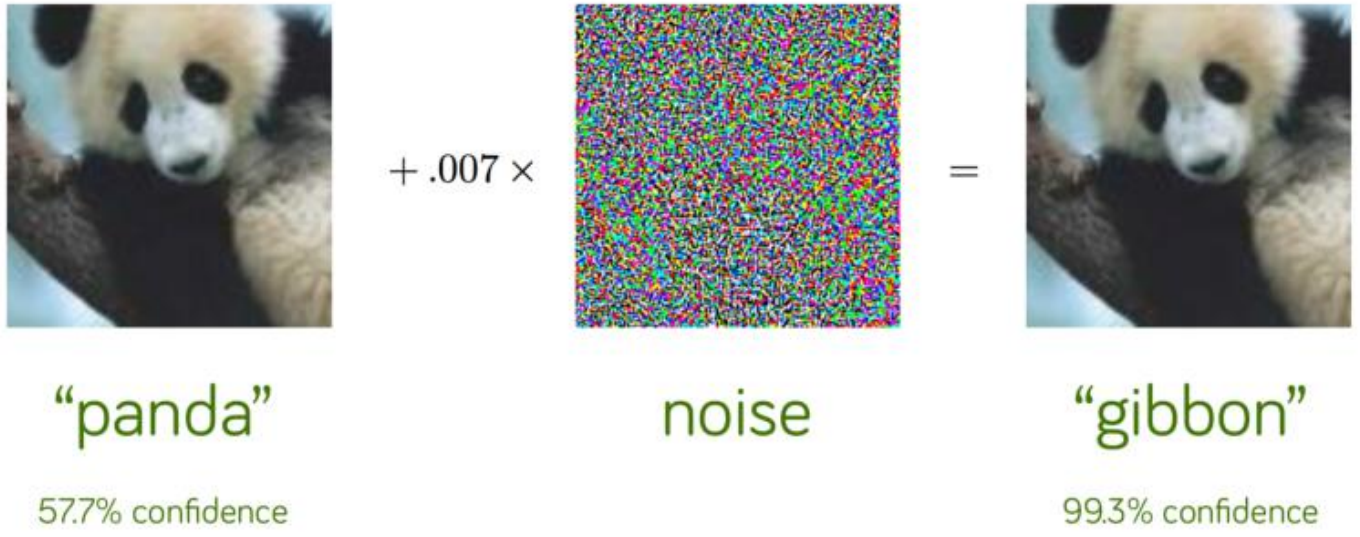
Deep neural networks are easily fooled

Nguyen A, Yosinski J, Clune J. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In Computer Vision and Pattern Recognition (CVPR '15), IEEE, 2015.



Towards Adversarial Examples

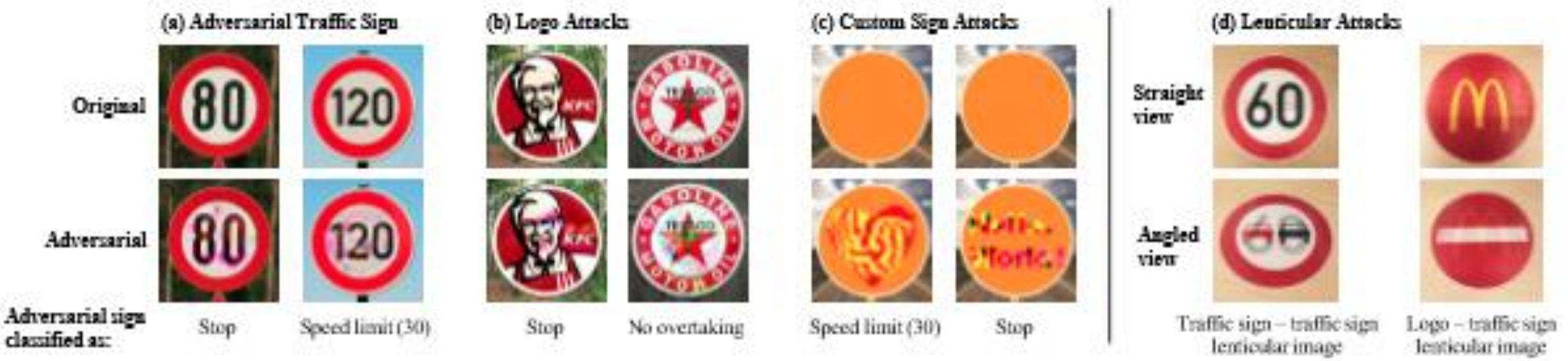
In 2014, a group of researchers at Google and NYU found that it was far too easy to fool ConvNets with an imperceivable, but carefully constructed nudge in the input.
Let's look at an example. We start with an image of a panda, which our neural network correctly recognizes as a "panda" with 57.7% confidence. Add a little bit of carefully constructed noise and the same neural network now thinks this is an image of a gibbon with 99.3% confidence



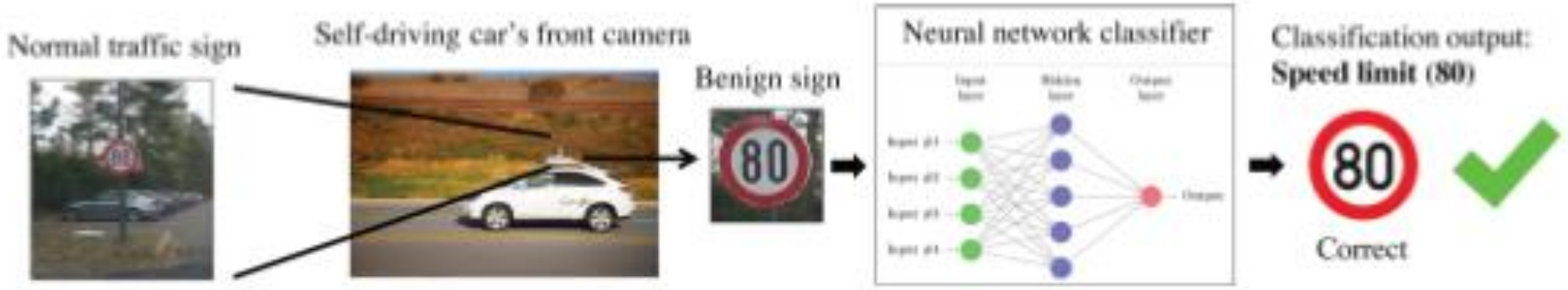
Source: Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015.

Adversarial Examples

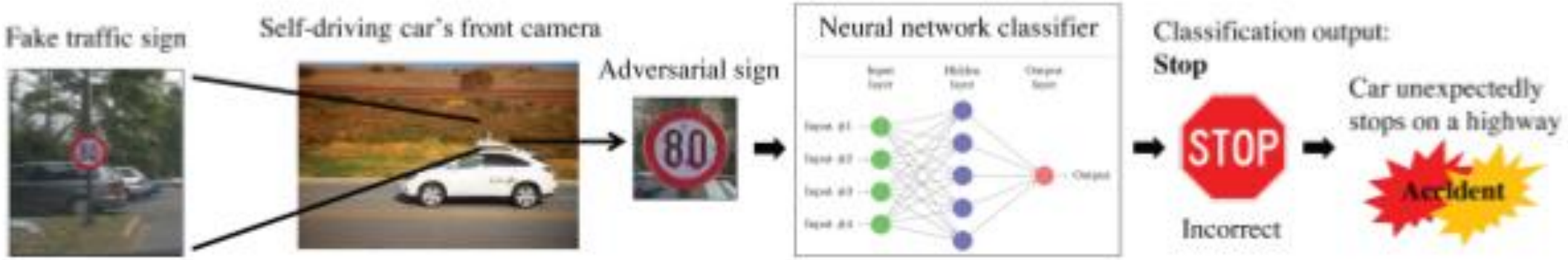
Adversarial examples are inputs to machine learning models that an attacker has intentionally designed to cause the model to make a mistake; they're like optical illusions for machines.



Autonomous Driving



(a) Operation of the computer vision subsystem of an AV under *benign conditions*

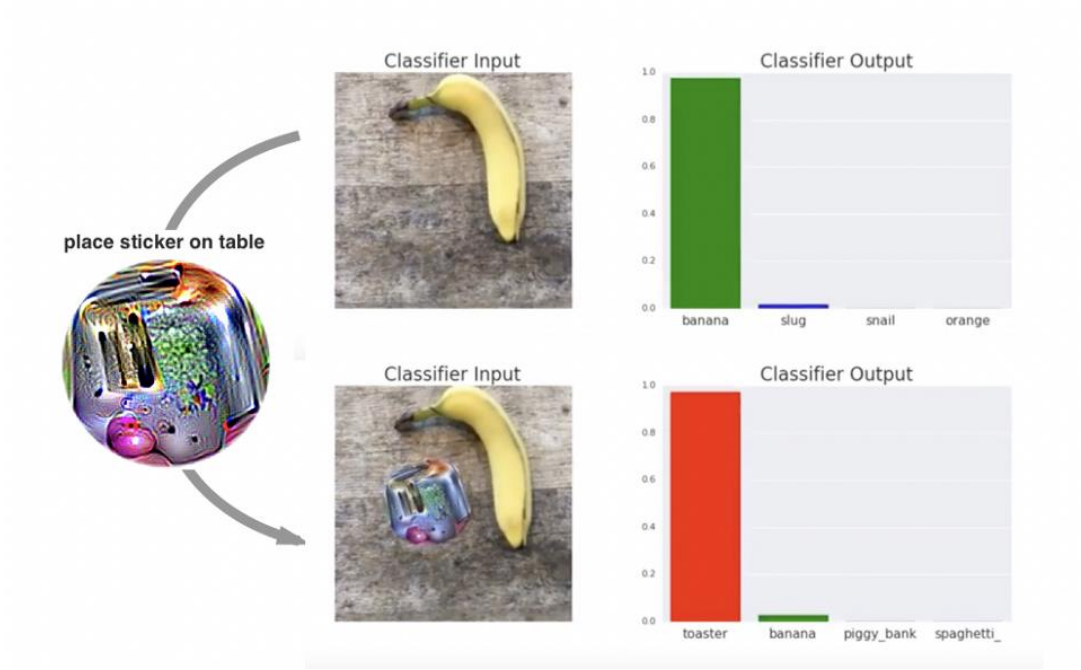


(b) Operation of the computer vision subsystem of an AV under *adversarial conditions*

https://arxiv.org/pdf/1802.06430.pdf

Adversarial Patch

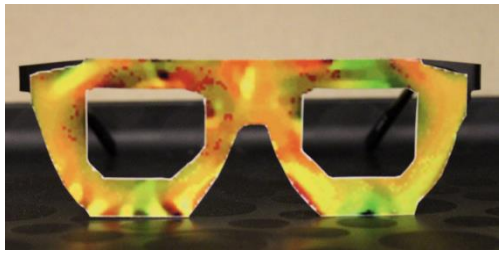
- *“Adversarial Patch”,* a paper published at NIPS 2017 demonstrated how to generate a patch that can be placed anywhere within the field of view of the classifier and cause the classifier to output a targeted class.



Source: Adversarial Patch: https://arxiv.org/pdf/1712.09665.pdf

Accessorize to a Crime (Adversarial Examples)

“Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition” showed that one can fool facial recognition software by constructing adversarial glasses by dodging face detection altogether



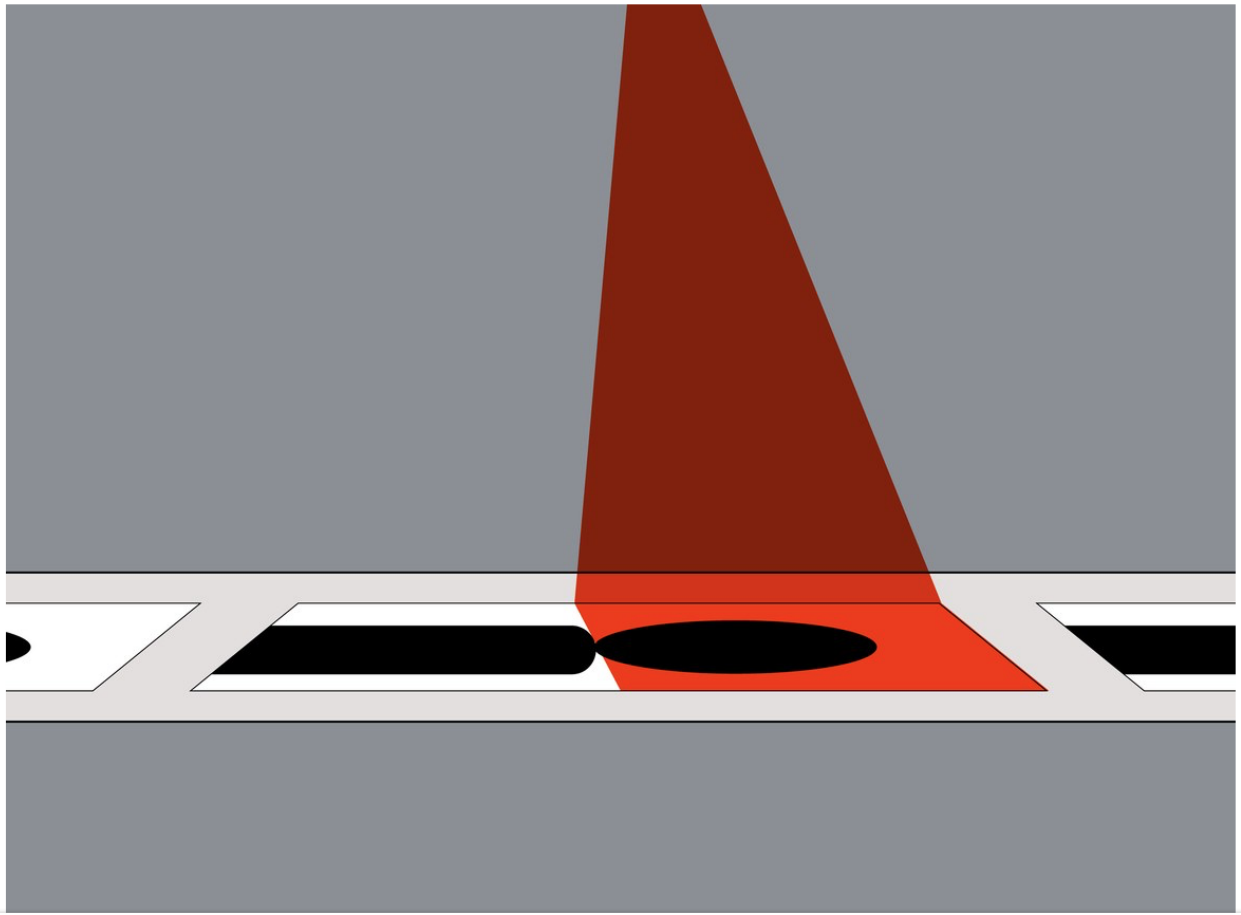
[Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition](). Sharif et al.

What is remarkable?

- It is easy to attain **high confidence** in the incorrect classification of an adversarial example
  - recall that in the first “panda” example we looked at, the network is less sure of an actual image looking like a panda (57.7%) than our adversarial example on the right looking like a gibbon (99.3%).
- Another intriguing point is how **imperceptibly little noise** we needed to add to fool the system — after all, clearly, the added noise is not enough to fool us, the human
- Adversarial examples don’t depend much on the specific deep neural network used for the task
  - an adversarial example trained for one network seems to confuse another one as well
  - This “**transferability**” enables attackers to fool systems in what are known as “black-box attacks” where they don’t have access to the model’s architecture, parameters or even the training data used to train the network.s

TOM SIMONITE BUSINESS 02.06.18 06:21 PM

PHOTO ALGORITHMS ID WHITE MEN FINE—BLACK WOMEN, NOT SO MUCH



Technische Hochschule Rosenheim

Recent test with Microsoft, IBM, and Face++ facial recognition:

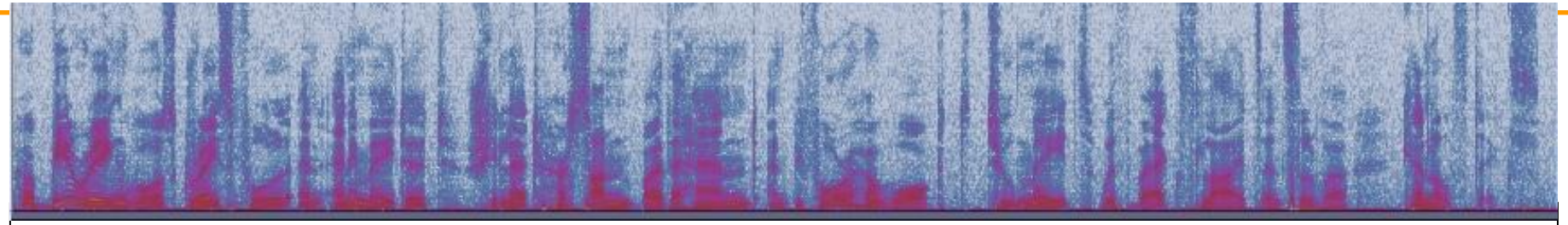
The companies' algorithms proved near perfect at identifying the gender of men with lighter skin, but frequently erred when analyzing images of women with dark skin.

1,270 photos of parliamentarians from Europe and Africa

Lightest male faces - error < 0.3%
Darker female faces – error > 21%

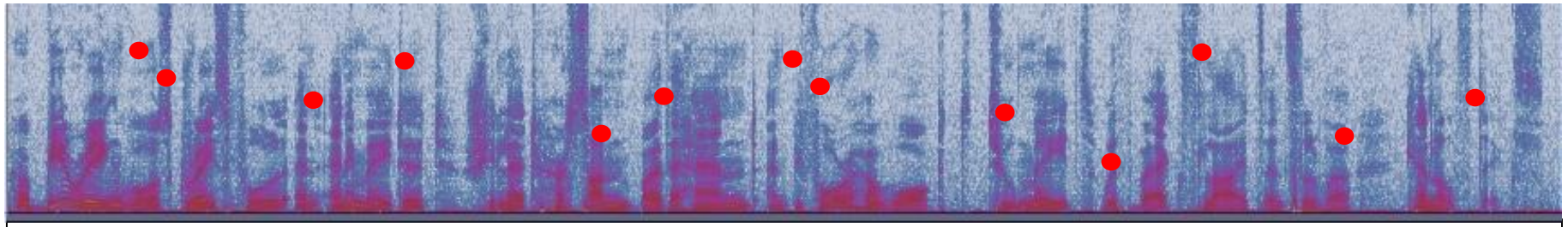
Remember Google's black people problem

Houdini: Fooling Deep Structured Prediction Models



Original transcribe:

Her bearing was graceful and animated she led her son by the hand and before her walked two maids with wax lights and silver candlesticks.



After modification:

Mary was grateful then admitted she let her son before the walks to Mays would like slice furnace filter count six

https://arxiv.org/abs/1707.05373

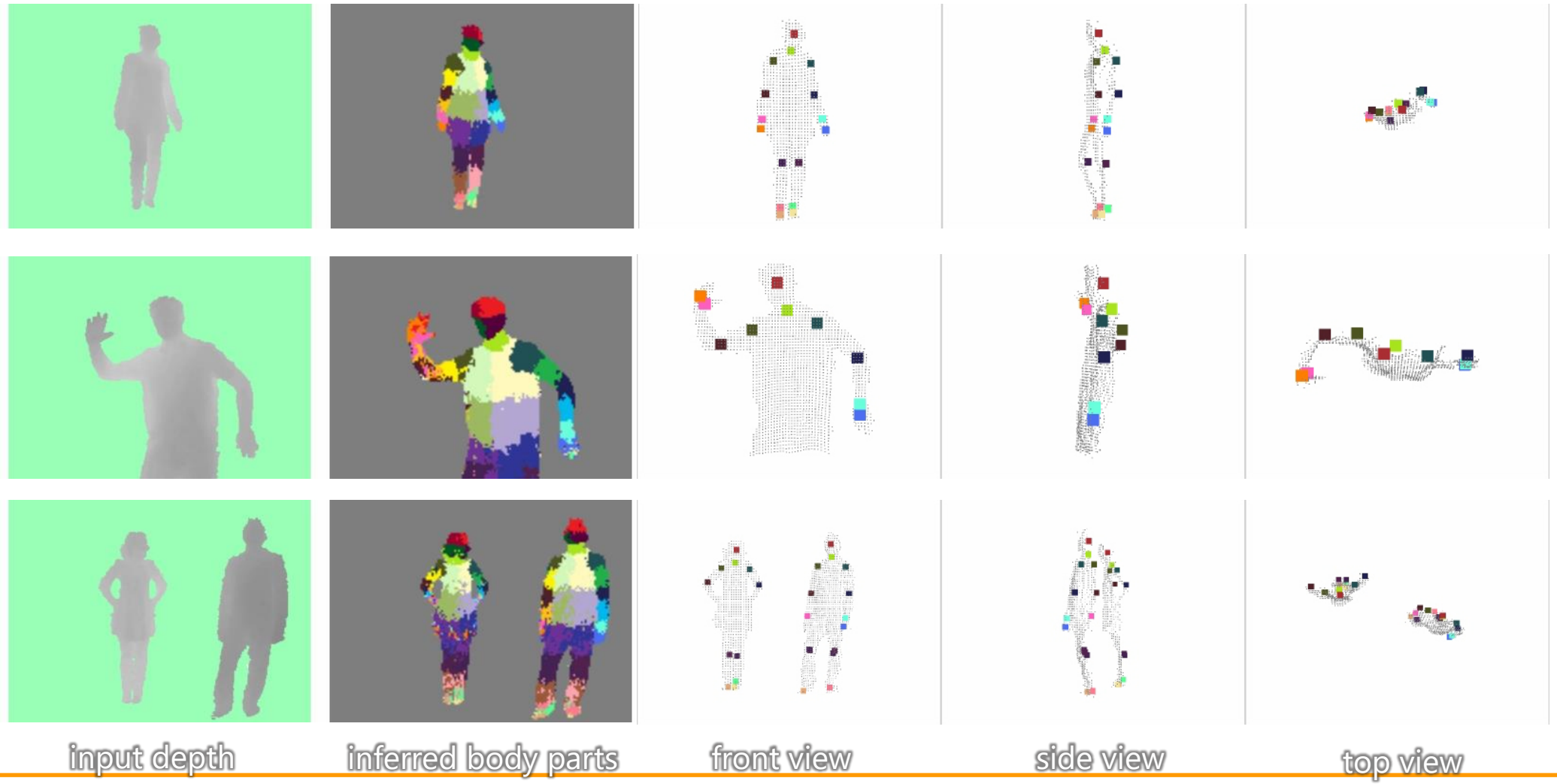
It is all about data!

- How to get the data you need?

- **Create yourself** – cumbersome
- **Datasets** – well, what if I do need different ones

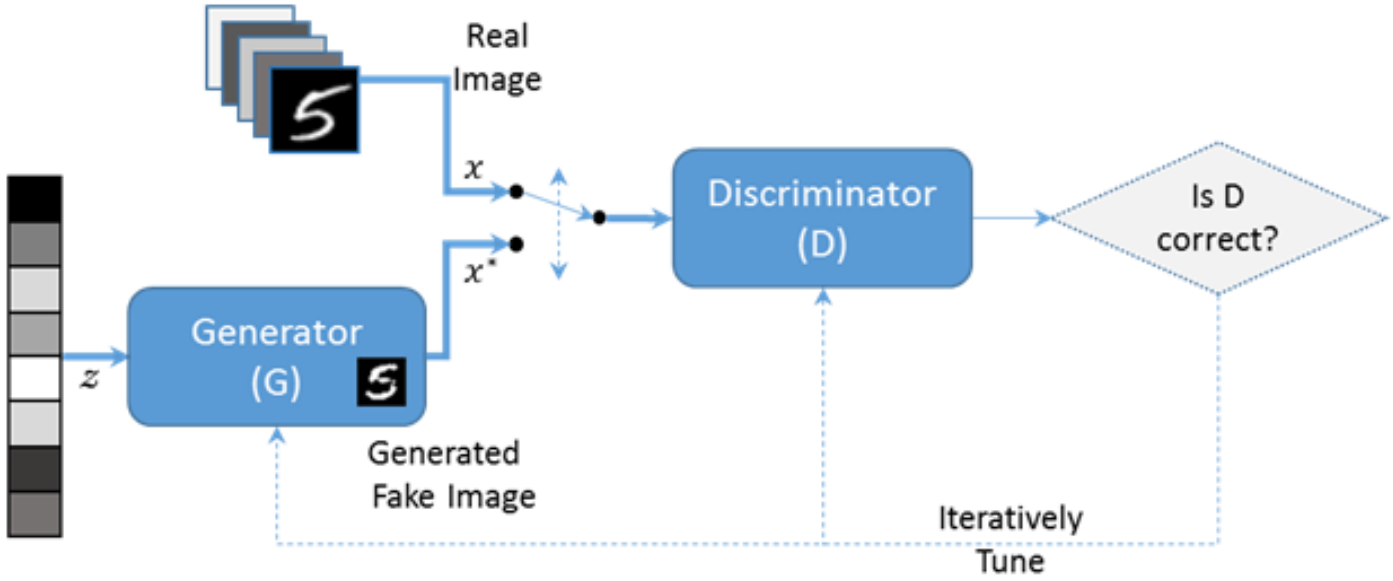
- **Generate**
  - Mix with real world images
  - From CAD programs
  - Script yourself

Computer Vision



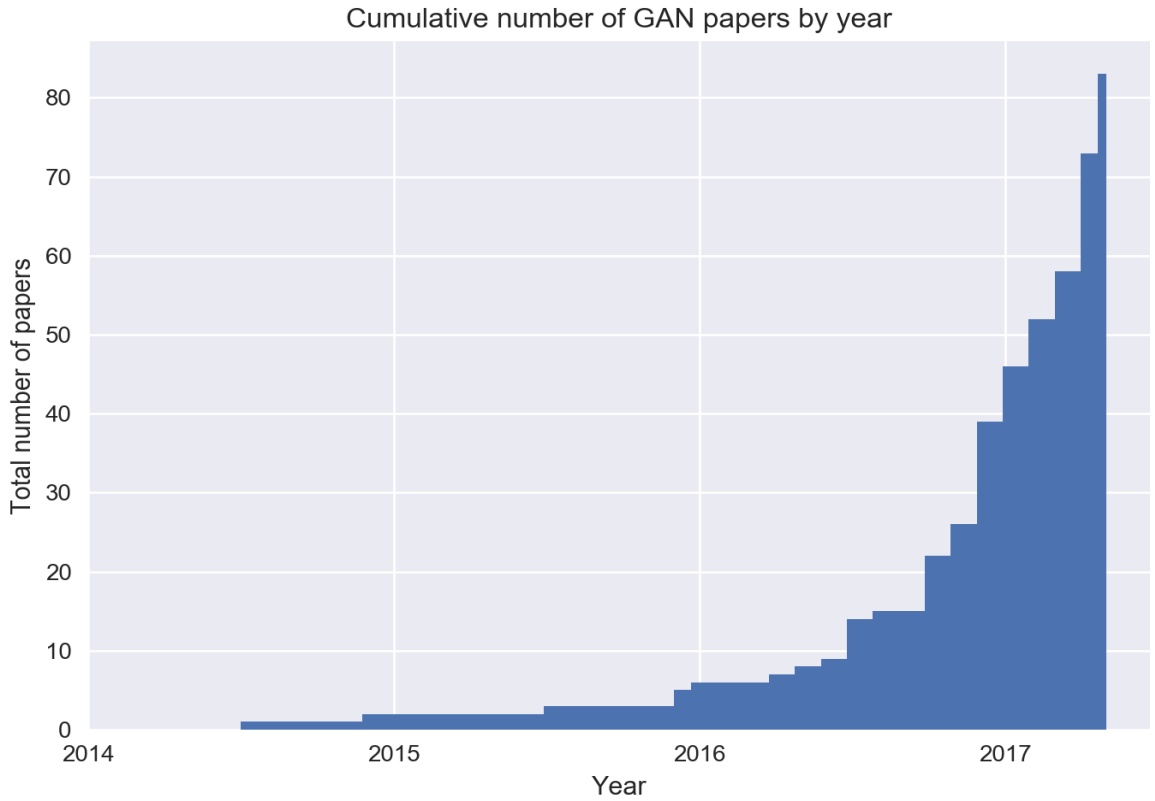
Generative Adversarial Network (GAN)

- Generative Adversarial Networks (GANs) are one of the most interesting ideas in computer science today. Two networks
  - A *Generator* takes input, generates images that look real
  - A *Discriminator* takes images, determines if fake
  - Train both at same time – generator learns to fool discriminator
  - https://www.tensorflow.org/tutorials/generative/dcgan



How fast are GANs changing?

Yann LeCun (Quora session):
"The most interesting idea in ML in the last 10 years"



Credit: Bruno Gavranović

CycleGAN Results

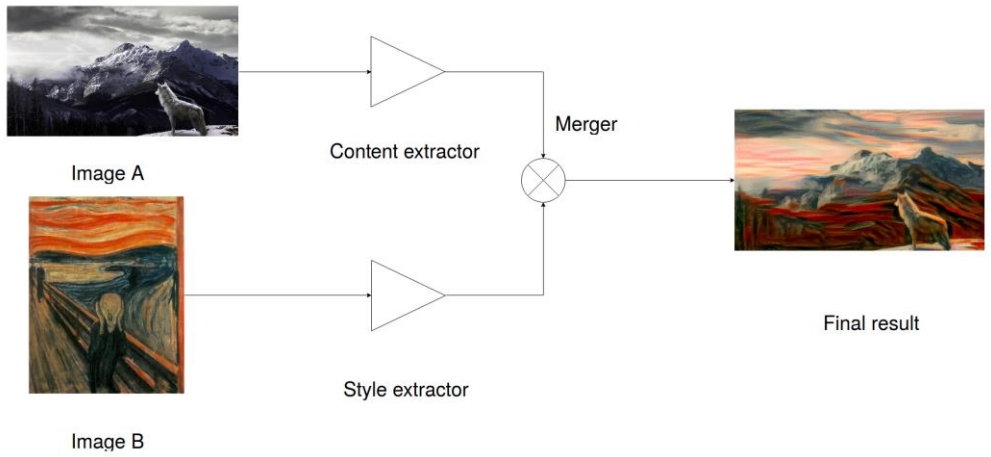
https://blog.statsbot.co/deep-learning-achievements-4c563e034257

Neural Style Transfer

- **Neural style transfer** is an optimization technique used to take two images—a *content* image and a *style reference* image (such as an artwork by a famous painter)—and blend them together so the output image looks like the content image, but “painted” in the style of the style reference image.
- https://www.tensorflow.org/tutorials/generative/style_transfer

- One image A contains the content
- One image B contains the style
- One image x is the target
- Loss functions
  - $L_{content}(c,x) \rightarrow 0$
  - $L_{style}(s,x) \rightarrow 0$
- Try to solve

$$x = \text{argmin}(L_{content}(c,x), + b\ L_{style}(s,x))$$



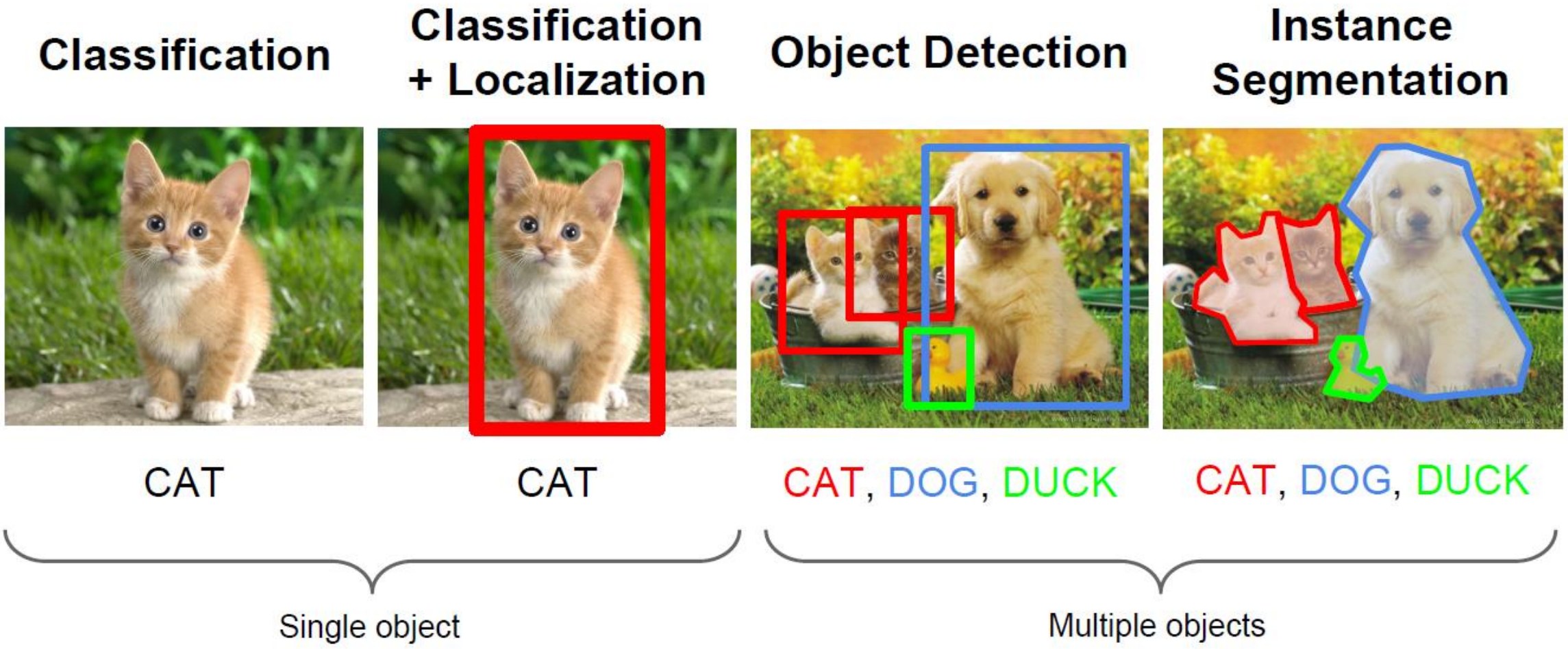
https://blog.paperspace.com/art-with-neural-networks/

Style Transfer Examples

https://www.eteknix.com/new-deep-photo-style-transfer-imaging-algorithm-accurately-copies-photo-styles/

Neural Style: Deepart.io

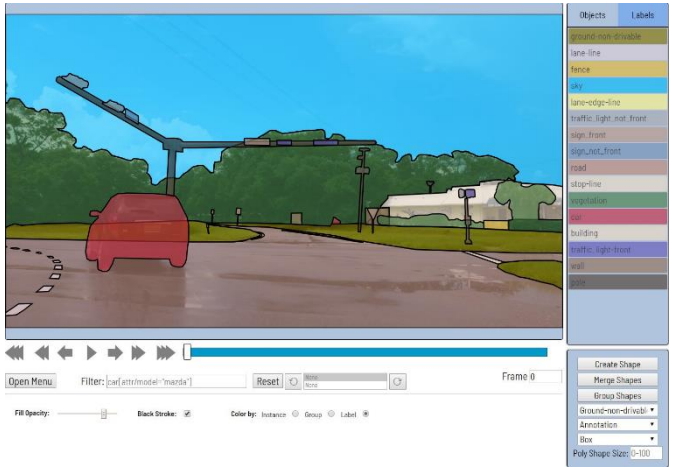
Taxonomy

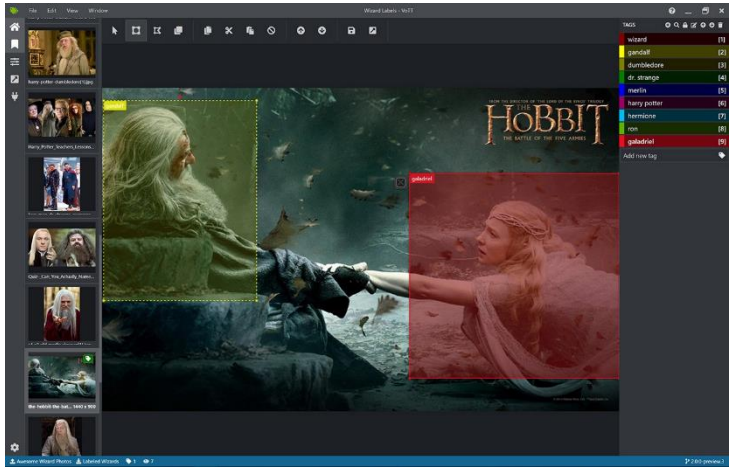


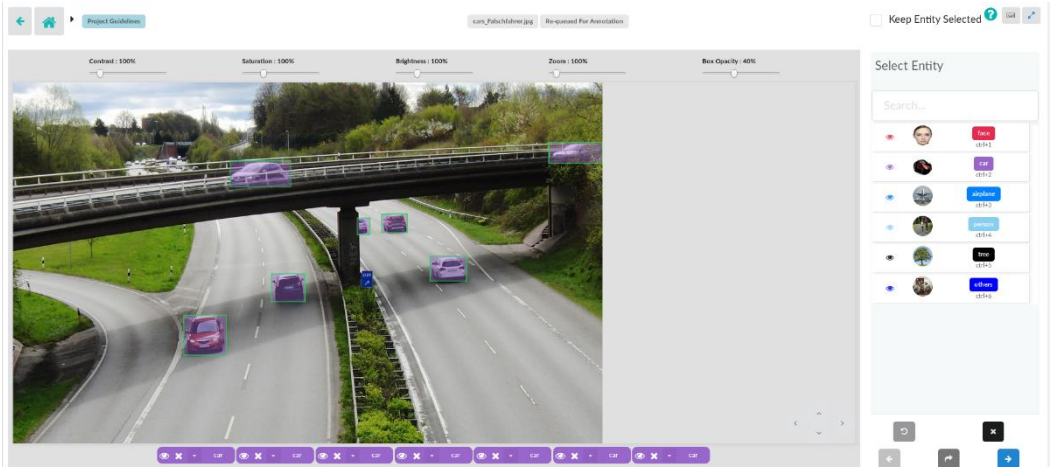
https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html

Data Preparation

- **VoTT (Visual Object Tagging Tool) - https://github.com/microsoft/vott**
- **Computer Vision Annotation Tool (CVAT) - https://github.com/opencv/cvat**
- **DataTurks - https://github.com/DataTurks/DataTurks**







Towards your own Image Classifier!

- We need data!
  - Screencast your own!
- We need a model!
  - Use: https://teachablemachine.withgoogle.com/
- We need some code!
  - First, capture your WebCam
  - Get frame-by-frame
  - Classify each frame
  - Show the result

Capture your WebCam!

- Use **openCV** and **Python**!
- Install openCV and Tensorflow

```
pip(3) install tensorflow
pip(3) install opencv-python
```

```
import cv2

def show_webcam(mirror=False):
    cam = cv2.VideoCapture(0)
    while True:

        ret_val, img = cam.read()
        # do something with img

        cv2.imshow('Webcam', img)
        if cv2.waitKey(1) == 27:
            break  # esc to quit
    cv2.destroyAllWindows()

if __name__ == '__main__':
    show_webcam(mirror=True)
```

Classify each frame

```
def predict(model, image):
    global labels

    # Create the array of the right shape to feed into the keras model
    # The 'length' or number of images you can put into the array is
    # determined by the first position in the shape tuple, in this case 1.
    data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

    #resize the image to a 224x224 with the same strategy as in TM2:
    #resizing the image to be at least 224x224 and then cropping from the center
    res = cv2.resize(image, dsize=(224, 224), interpolation=cv2.INTER_CUBIC)

    # Normalize the image
    normalized_image_array = (res.astype(np.float32) / 127.0) - 1

    # Load the image into the array
    data[0] = normalized_image_array

    # run the inference
    prediction = model.predict(data)
    return labels[np.argmax(prediction)-1], np.amax(prediction)
```

Do not forget to initialize the model!

```
import numpy as np
import tensorflow.keras

labels = []

def initialize():
    # load labels
    global labels

    with open("labels.txt", "r") as f:
        labels = f.read().splitlines()

    # Disable scientific notation for clarity
    np.set_printoptions(suppress=True)

    # Load the model
    return tensorflow.keras.models.load_model('keras_model.h5')
```

Summary

- Analyse Contents of images with Azure, Google and AWS
- Services for custom vision models and exporting for various frameworks
- Detect and identify people, emotions and age
- Extract text, key-value pairs, and tables from documents
- Recognize digital ink and handwriting