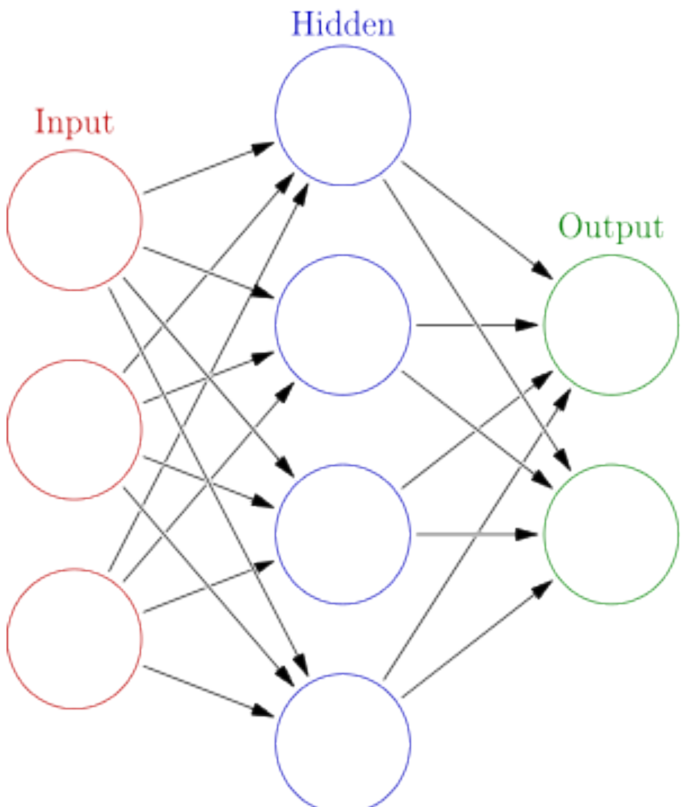
A Brief Introduction to Deep Learning

--Yangyan Li

A Brief Introduction to Deep Learning

- Artificial Neural Network
- Back-propagation
- Fully Connected Layer
- Convolutional Layer
- Overfitting
- Useful code & examples

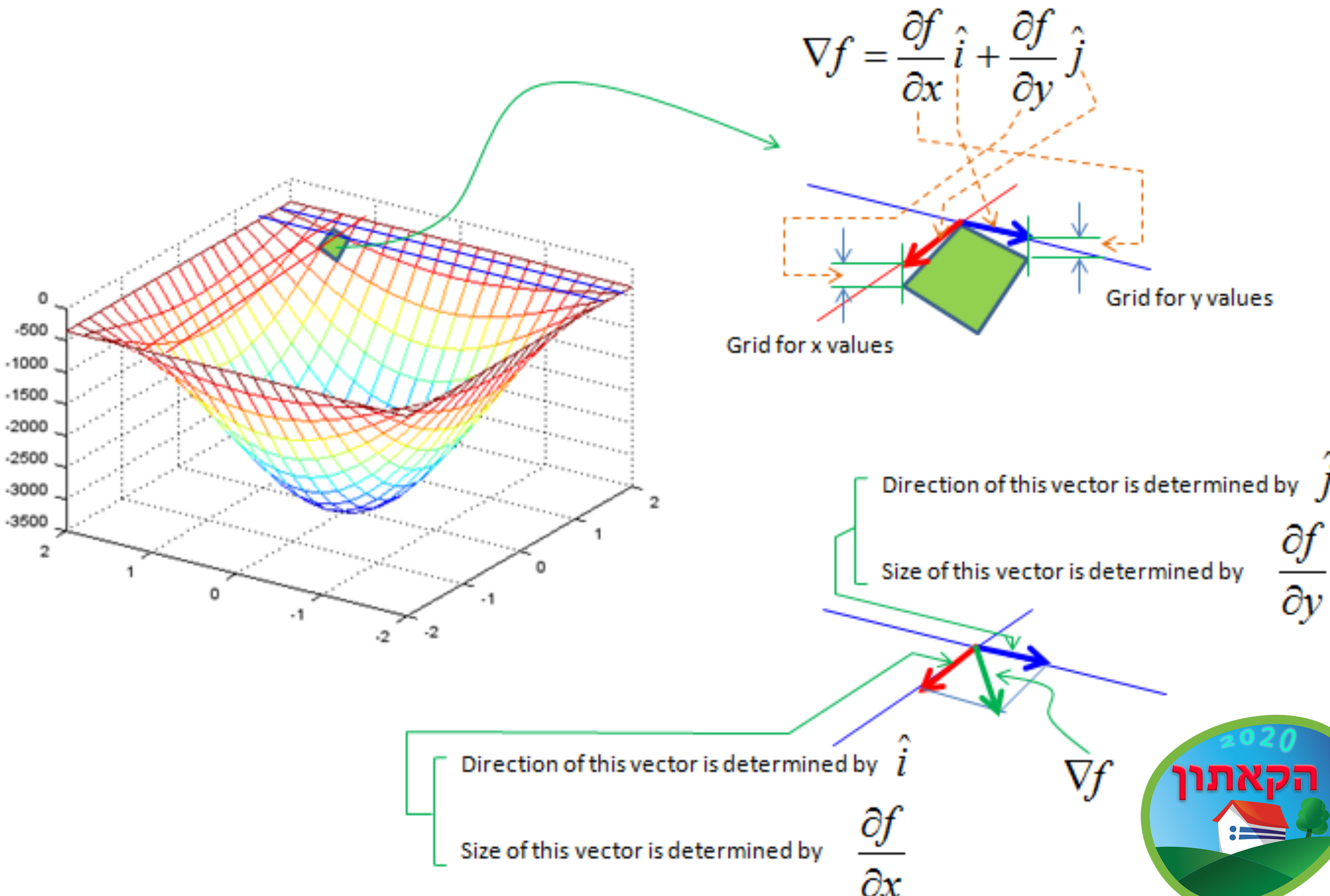
Artificial Neural Network



1. Learning algorithm
2. Neurons
3. Cost function
4. Activation function
5. Weights
6. Models

Live Demo

Gradient descent



Neurons are functions

- Let's start with a complex one!

$$f(x, y) = x + y$$

- Given $x = a, y = b$, how to update x *and* y to make $f(x, y)$ larger?
- Follow gradient directions!

$$f(x, y) = x + y \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = 1 \qquad \frac{\partial f}{\partial y} = 1$$

$$x = a + 0.01 * 1,$$
$$y = b + 0.01 * 1$$
$$f(x, y): a + b \rightarrow \mathrm{a} + \mathrm{b} + 0.02$$

Neurons are functions

- A more complex one!

$$f(x, y) = x * y$$

- Given $x = a, y = b$, how to update x *and* y to make $f(x, y)$ larger?
- Follow gradient directions!

$$f(x,y) = xy \qquad \rightarrow \qquad \frac{\partial f}{\partial x} = y \qquad \frac{\partial f}{\partial y} = x$$

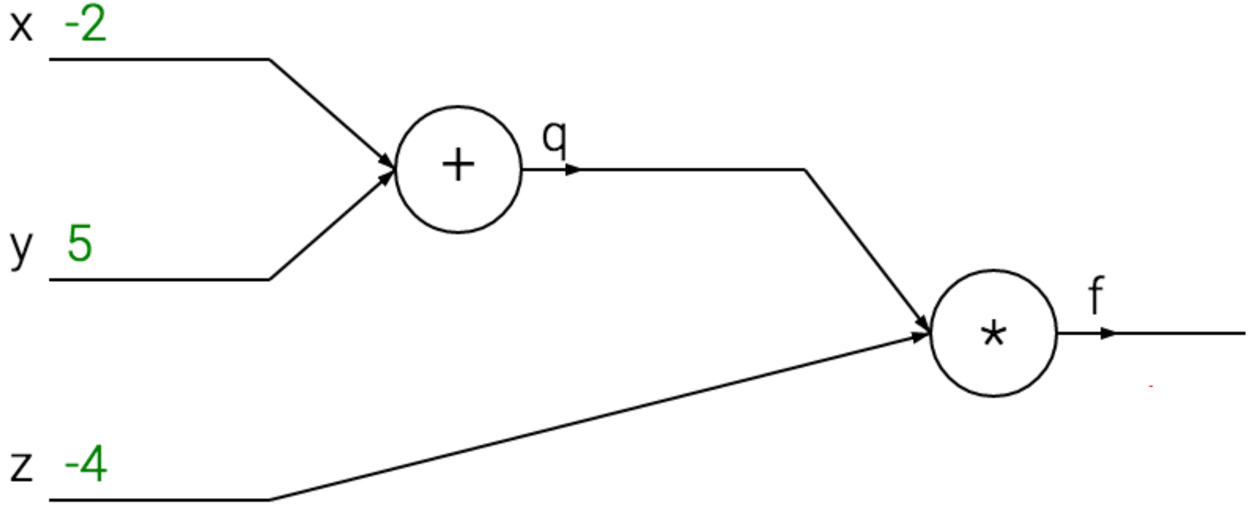
$$x = a + 0.01 * b,$$
$$y = b + 0.01 * a$$
$$f(x, y) \colon a * b \rightarrow (\mathrm{a} + 0.01 * \mathrm{b})(\mathrm{b} + 0.01 * \mathrm{a})$$
$$f(x, y) \colon 4 * (-3) \rightarrow 3.97 * (-2.96)$$

Back-propagation

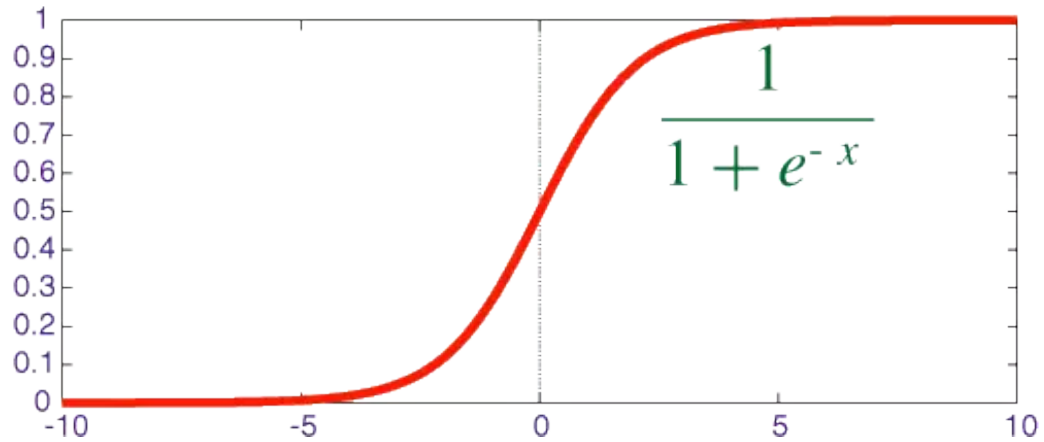
- An extremely complex one!

$$f(x,y,z) = (x+y) * z$$

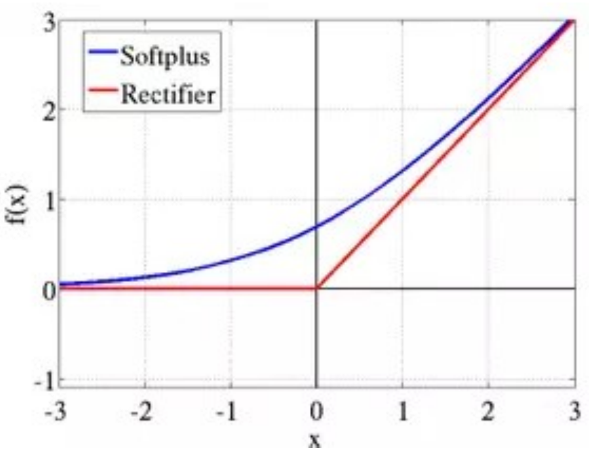
- Let $q(x,y) = (x+y)$, then $f(x,y,z) = q(x,y) * z$
- Chain rule: $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$



Sigmod → ReLU

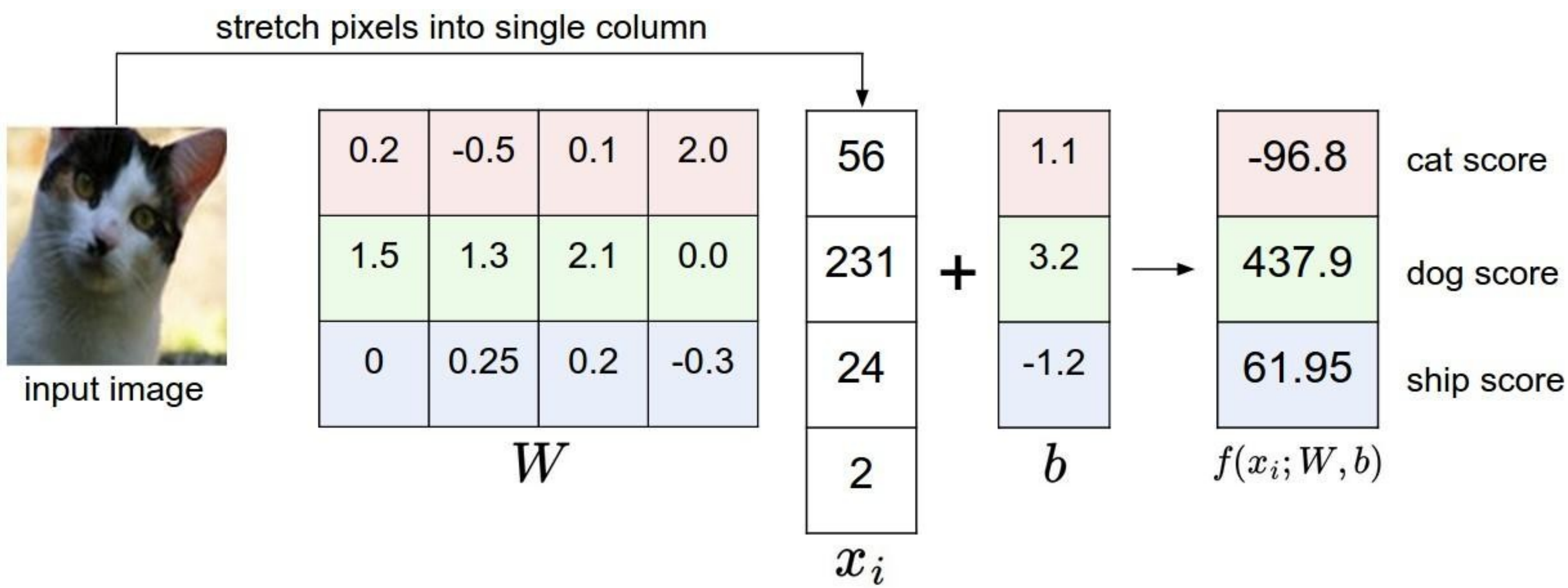


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-10
-5
0
5
10
$\frac{1}{1+e^{-x}}$

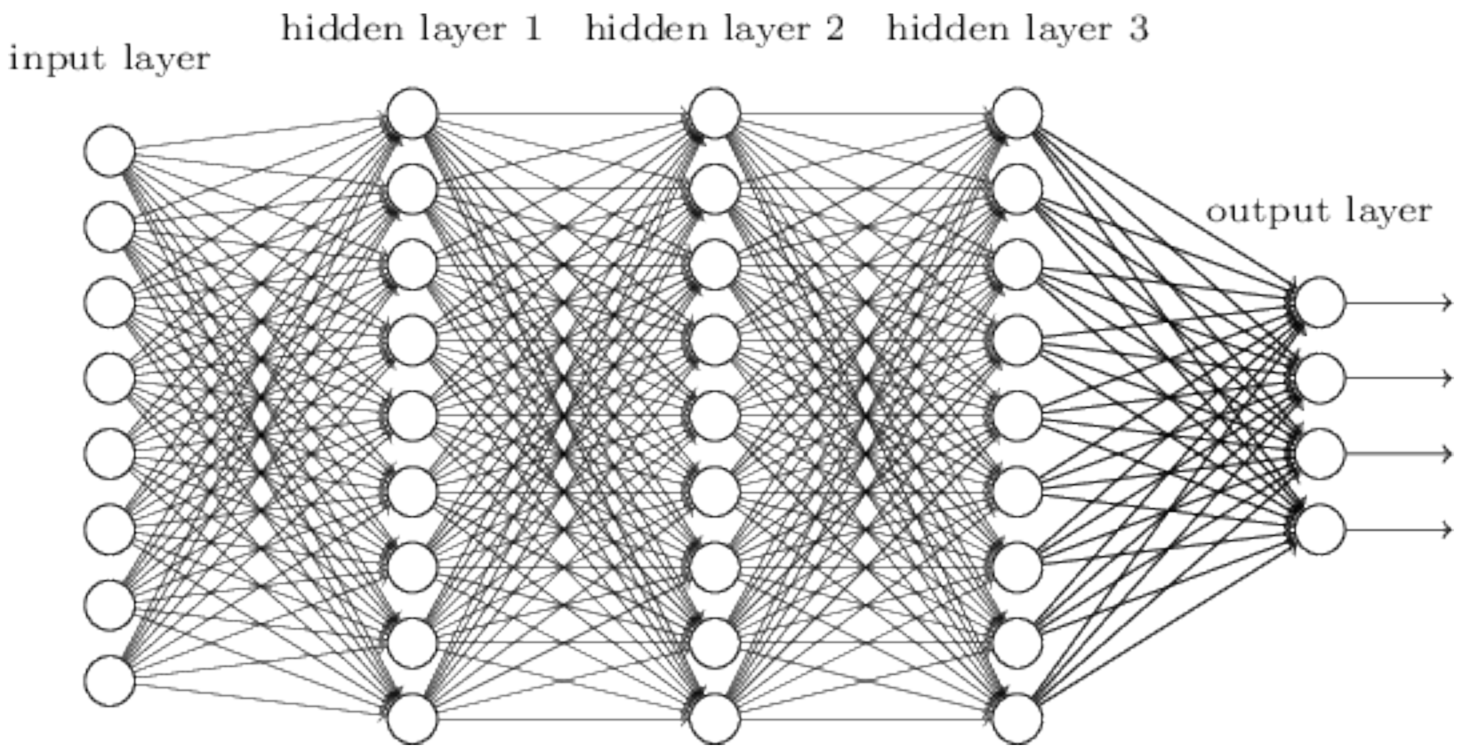


Softplus
Rectifier
f(x)
x
3
2
1
0
-1
-3
-2
-1
0
1
2
3

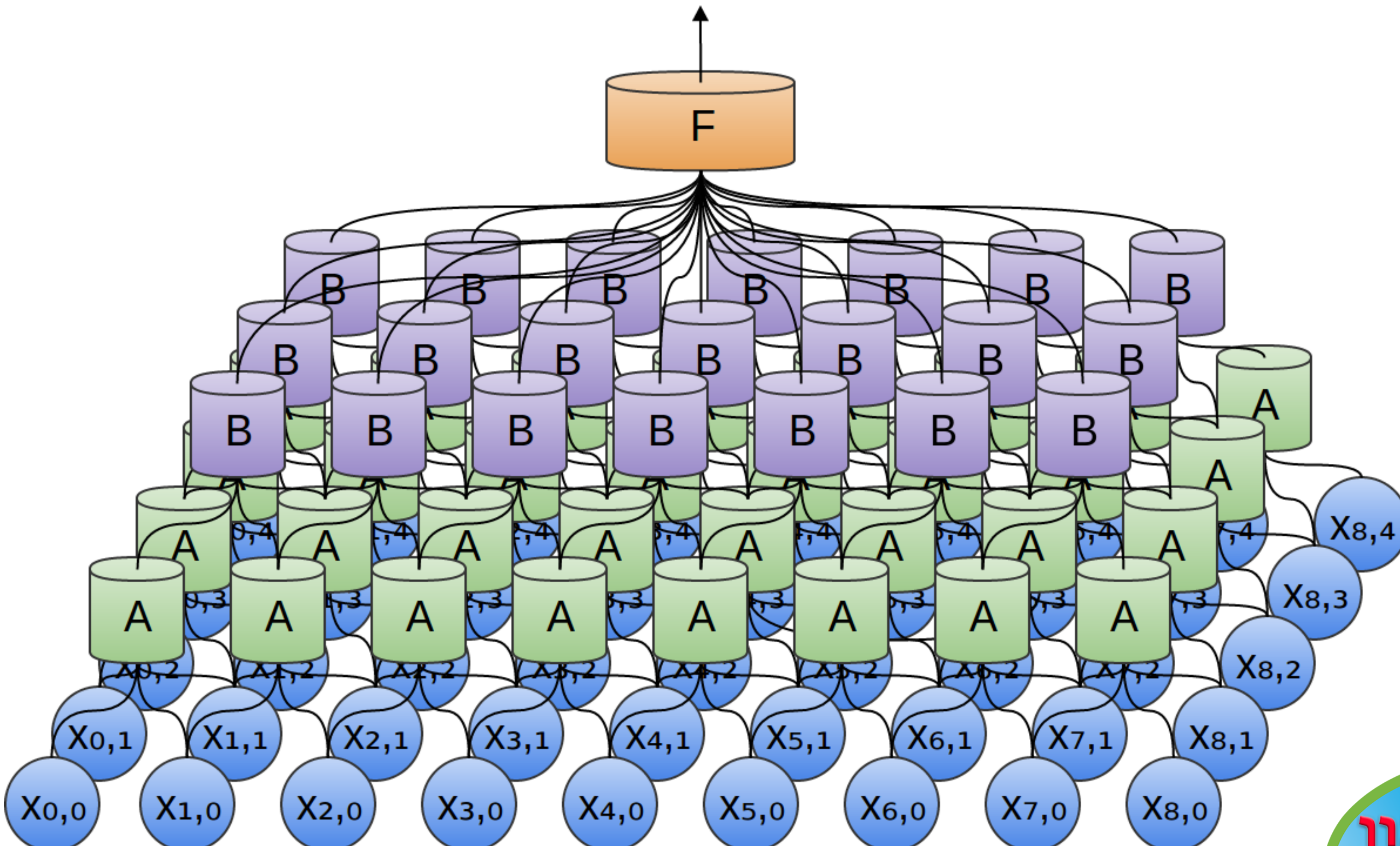
Now, serious stuff, a bit…



Fully Connected Layers



Convolutional Layers



Traditional Recognition Approach

Features are not learned

Input data (pixels) → feature representation (hand-crafted) → Learning Algorithm (e.g., SVM)

Image

Low-level vision features (edges, SIFT, HOG, etc.)

Object detection / classification

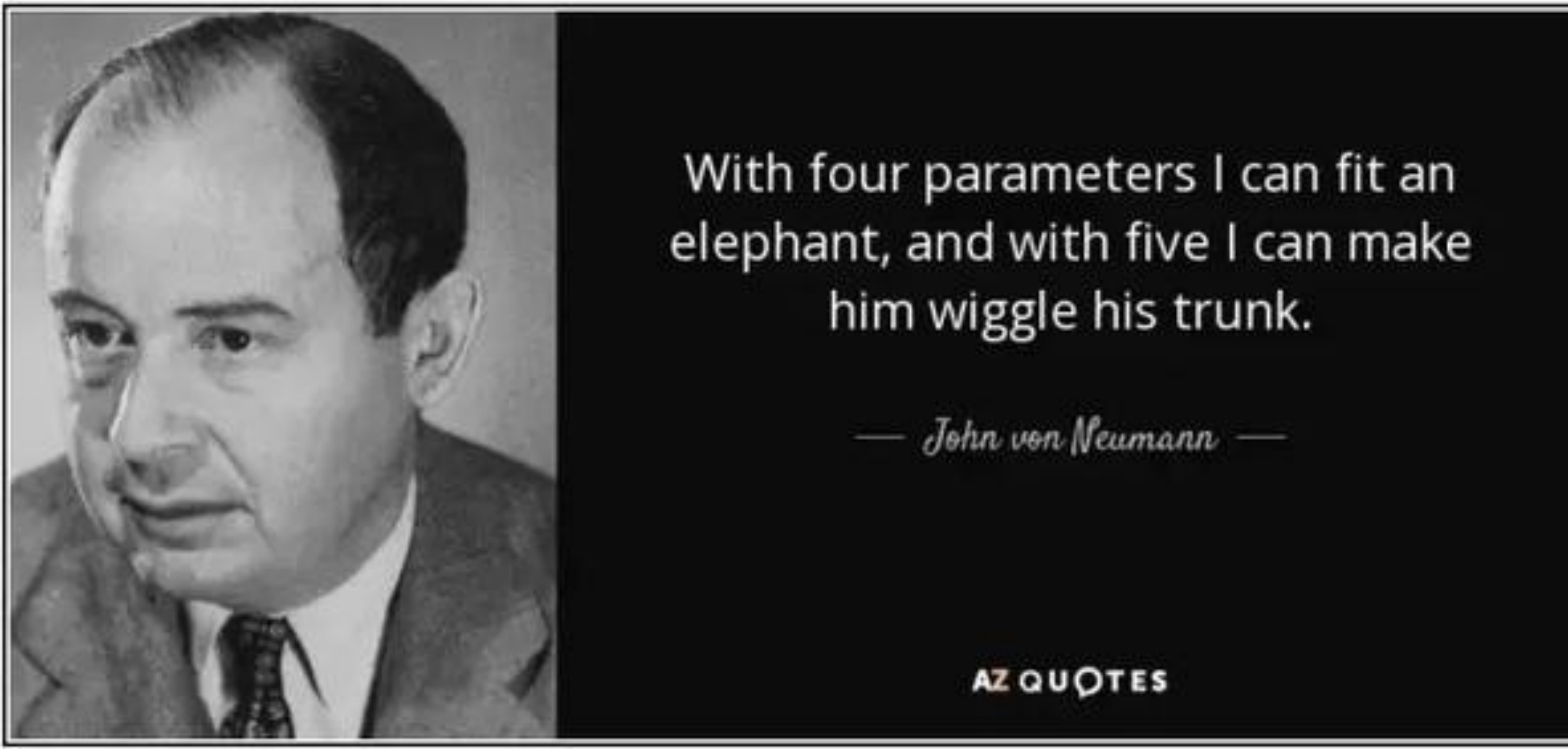
Feature Engineering vs. Learning

- Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work.
- “When working on a machine learning problem, feature engineering is manually designing what the input x's should be.”

-- Shayne Miel

- “Coming up with features is difficult, time-consuming, requires expert knowledge.”

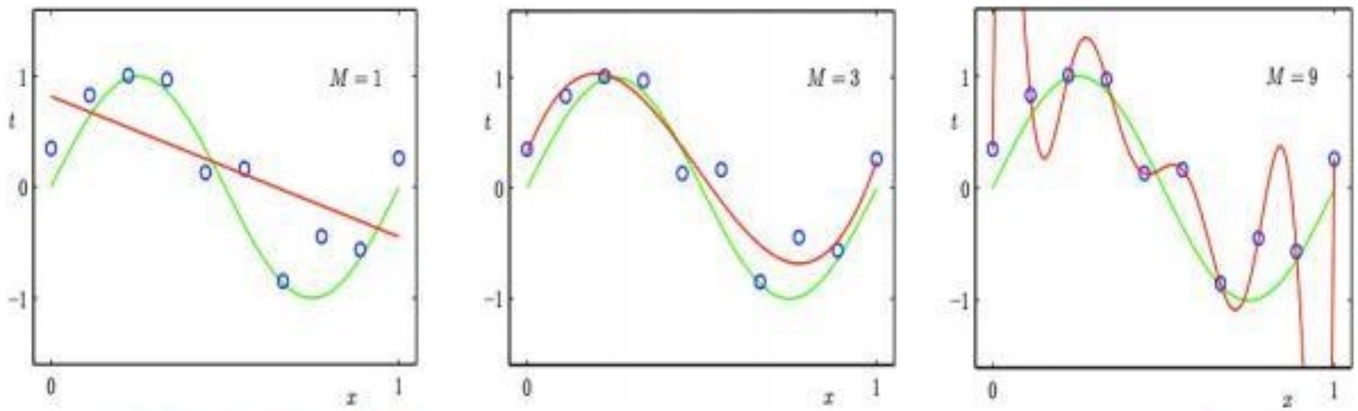
--Andrew Ng


With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.
— John von Neumann —
AZ QUOTES

2020
הקאתון

Under- and Over-fitting examples

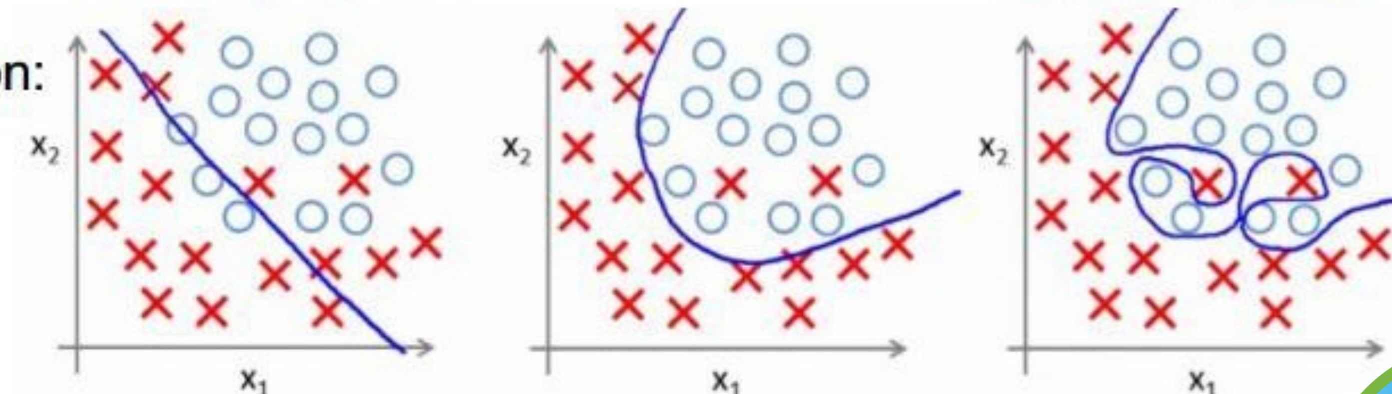
Regression:



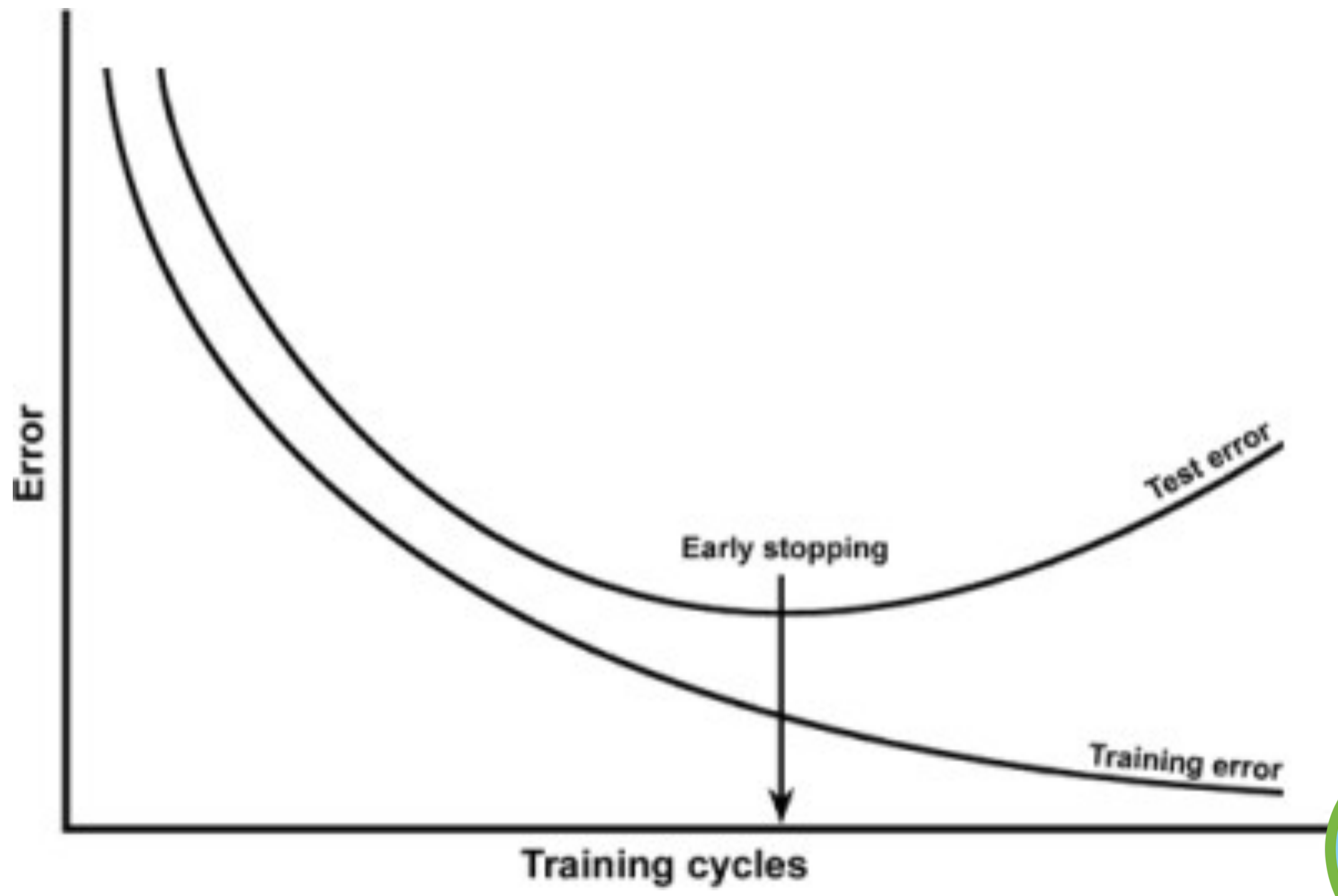
predictor too inflexible: cannot capture pattern

predictor too flexible: fits noise in the data

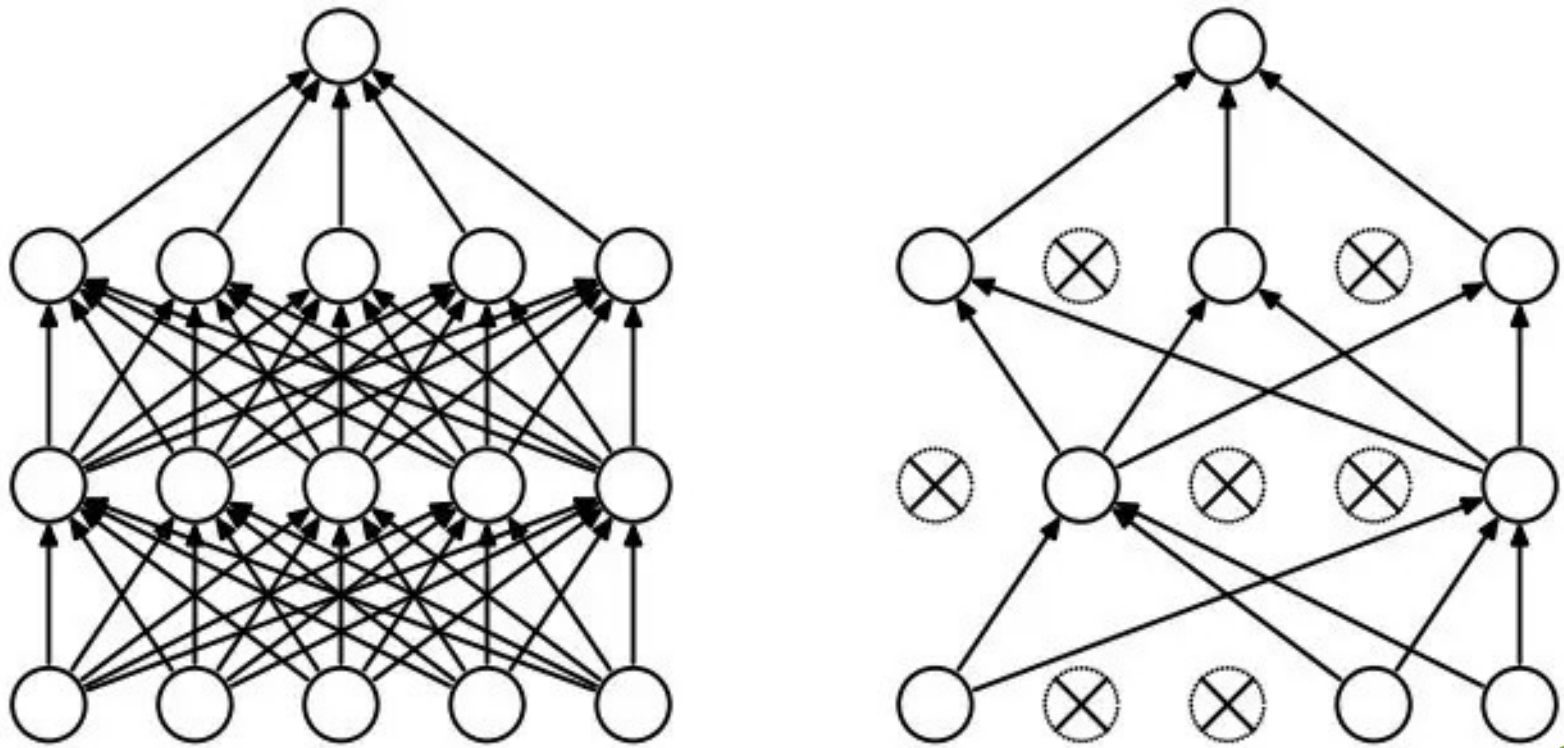
Classification:



How to detect it in training process?



Dropout



A brief history

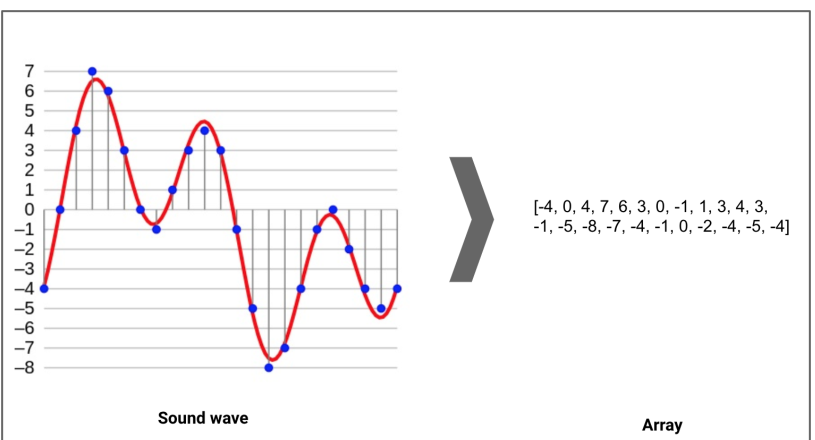
- McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.
- Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.
- Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *Cognitive modeling* 5.3 (1988): 1.
- LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." *Neural computation* 1.4 (1989): 541-551.
- 1993: Nvidia started…
- Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18.7 (2006): 1527-1554.
- Raina, Rajat, Anand Madhavan, and Andrew Y. Ng. "Large-scale deep unsupervised learning using graphics processors." *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009.
- Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database."*Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009.
- 2010: “GPUS ARE ONLY UP TO 14 TIMES FASTER THAN CPUS” SAYS INTEL –Nvidia
- Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." *International Conference on Artificial Intelligence and Statistics*. 2011.
- Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." *arXiv preprint arXiv:1207.0580* (2012).
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

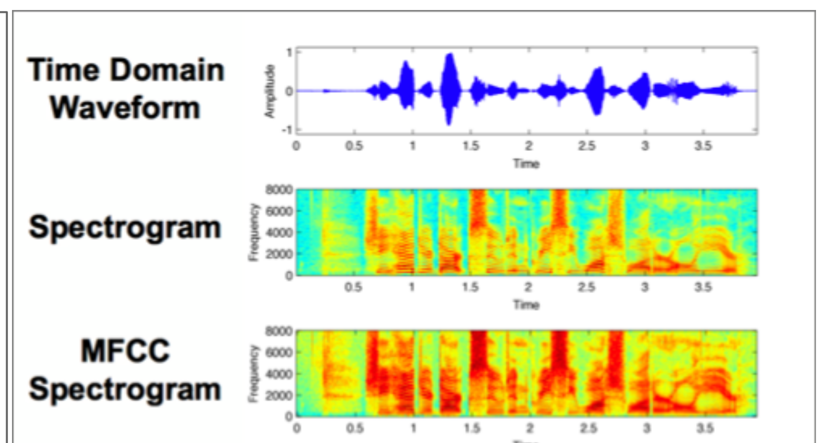
What are we going to do

- Deep Learning techniques to classify sounds
- Preprocess data, feed it to neural network & train

Towards a neural network

- For much of the preprocessing we will be able to use Librosa's load() function
- Create a visual representation of each of the audio samples which will allow us to identify features for classification, using the same techniques used to classify images with high accuracy
- Images will be the MFCC of the signals
- Useful functions: librosa.load, librosa.feature.mfcc, pd.DataFrame





Towards a neural network

- Converting the data and labels then splitting the dataset:
  - Using a csv with filename<->digit, load the files and compose an array of [[MFCC,digit] for file in files]
  - Convert into a Panda dataframe
    - features_df = pd.DataFrame(features, columns=['feature','class_label’])
  - Encode the classification labels
    - from sklearn.preprocessing import LabelEncoder
    - from keras.utils import to_categorical
    - le = LabelEncoder()
    - yy = to_categorical(le.fit_transform(y))
    - x_train, x_test, y_train, y_test = train_test_split(X, yy, test_size=0.2, random_state = 42)

Towards a neural network

- Normalization = make data lie on a scale
- Consider using normalization to speed

Towards a neural network

- We are going to use CNN

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, input_shape=(num_rows, num_columns, num_channels), activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=32, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=64, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))

model.add(Conv2D(filters=128, kernel_size=2, activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(GlobalAveragePooling2D())

model.add(Dense(num_labels, activation='softmax’))

model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."

--Winston Churchill