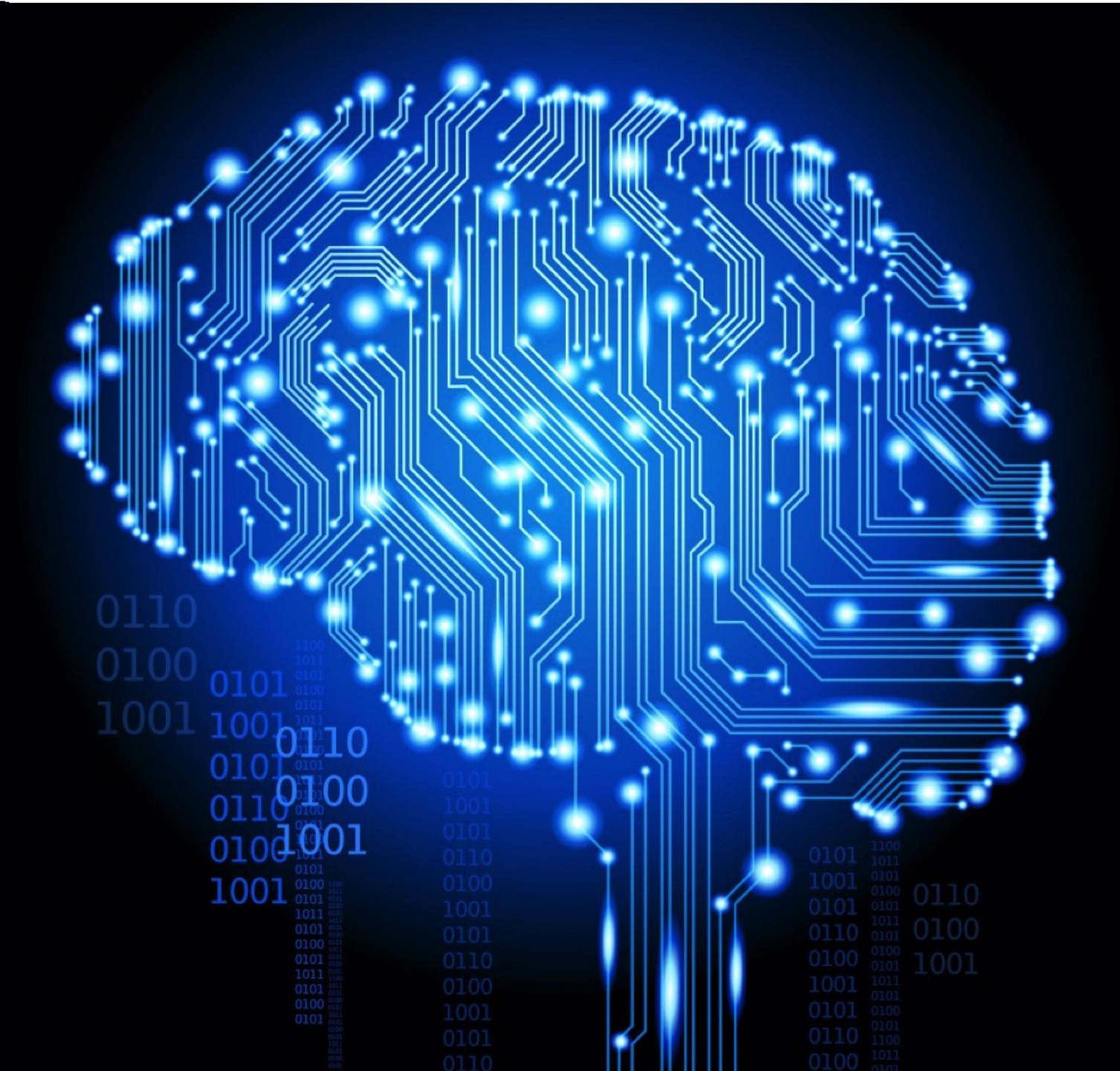
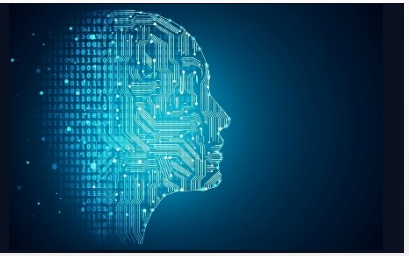
DEEP LEARNING
Part I

Leeor Langer

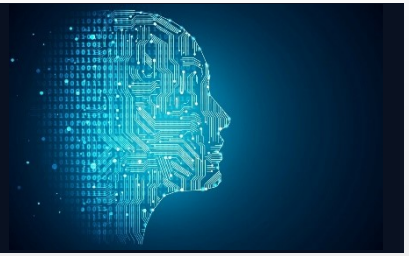


AGENDA



1. Intro: Background and Context
2. Technology trends and use cases
3. What is a deep neural net?
4. Tensorflow using Python environment
5. Build a basic neural net for classification\regression

WHAT IS DEEP LEARNING?

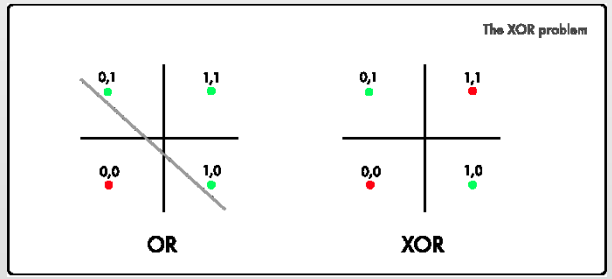


Physics ⟷ Engineering

Statistics ⟷ Deep Learning

-Sridhar Mahadevan

AI TIMELINE



1930s
- Turing Machine (Turing)
- Incompleteness Theorem (Gödel)

1950s
- Perceptron (Rosenblatt)

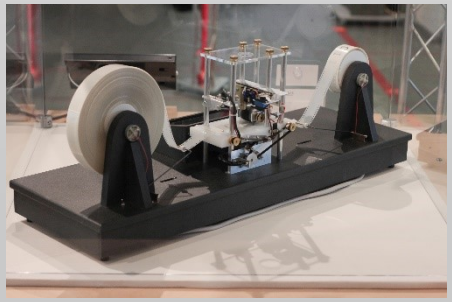
1960s
- Bayesian Methods (Solomonoff)

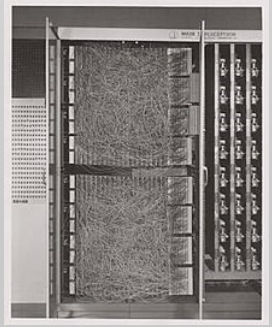
1970s
- XOR problem

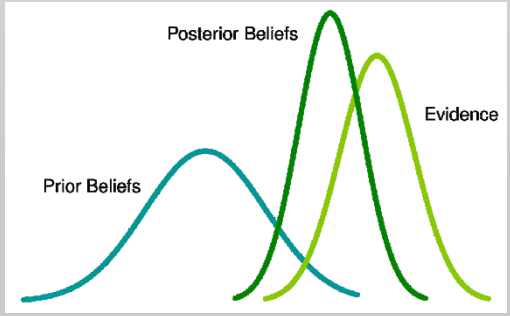
1980s
- Multi Layer Perceptron
- Backpropagation (Hinton)

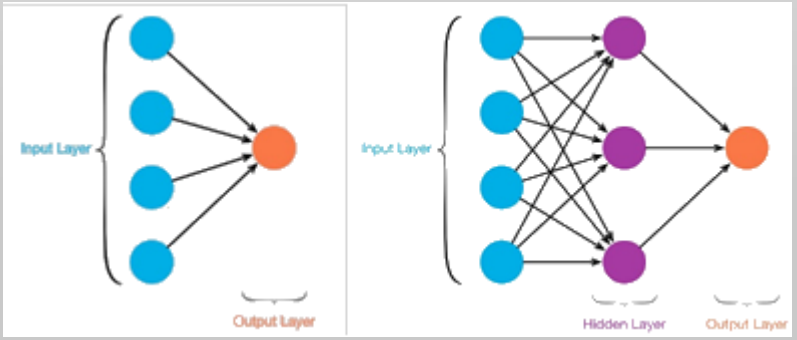
1990s
- SVM
- Random Forest
- (Vapnik, Breiman)

2000s
- Deep Neural Networks
- (Lecun, Benjio, Hinton)

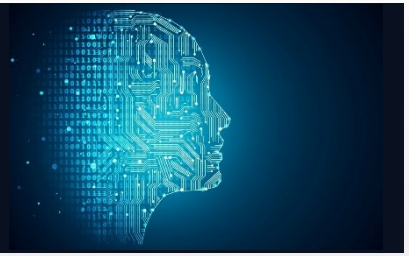








WHAT IS DEEP LEARNING?

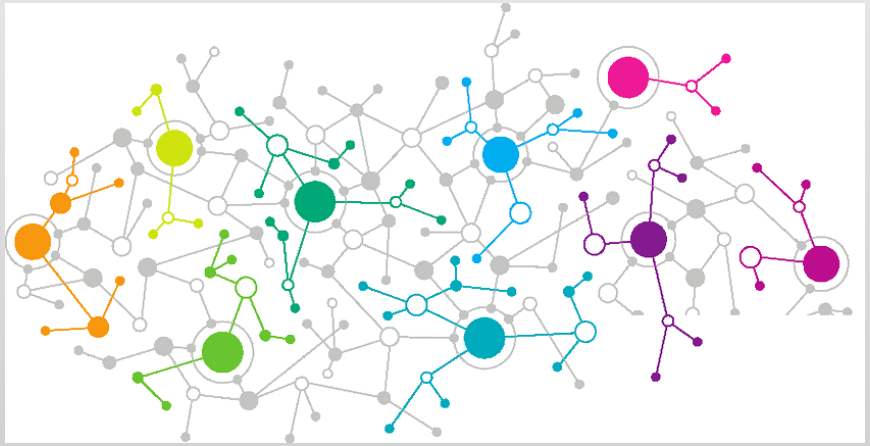


“There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown. The statistical community has been committed to the almost exclusive use of data models… If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.“

- The Two Cultures, Leo Breiman

THE BIG PICTURE

Complex models

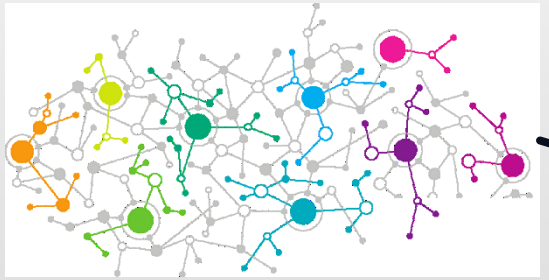


Massive Compute Power

Big Data

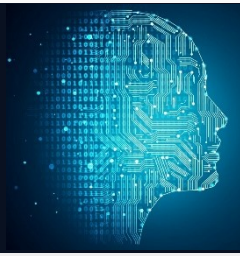
Deep Learning = Complex models + Massive Compute Power + Big Data

THE BIG PICTURE: ROCKET ANALOGY



Rocket Engine = Complex Model
Big Data = Fuel
-Andrew NG

THE BIG PICTURE



Sorting Lego Blocks

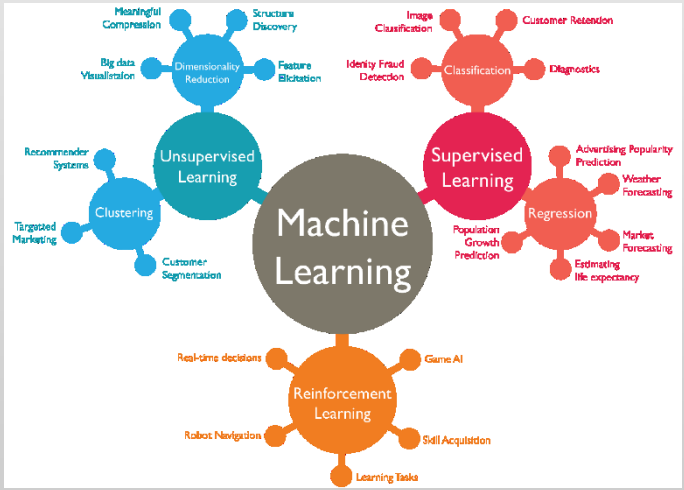
Simple example of a child sorting Lego blocks illustrates the differences between the three machine learning styles

Unsupervised Learning
Child has to sort blocks by color, shape or both with no instructions

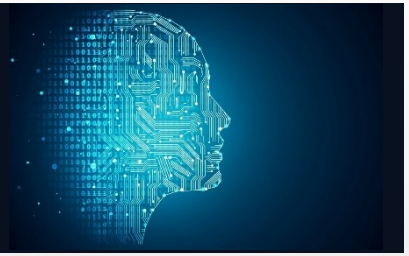
Supervised Learning
Child has to sort the colored blocks by matching the colors of the block with the colors of the bag

Reinforcement Learning
Child gets feedback from Mom when he does something right or something wrong

source pwc via @mikequindazzi



THE BIG PICTURE

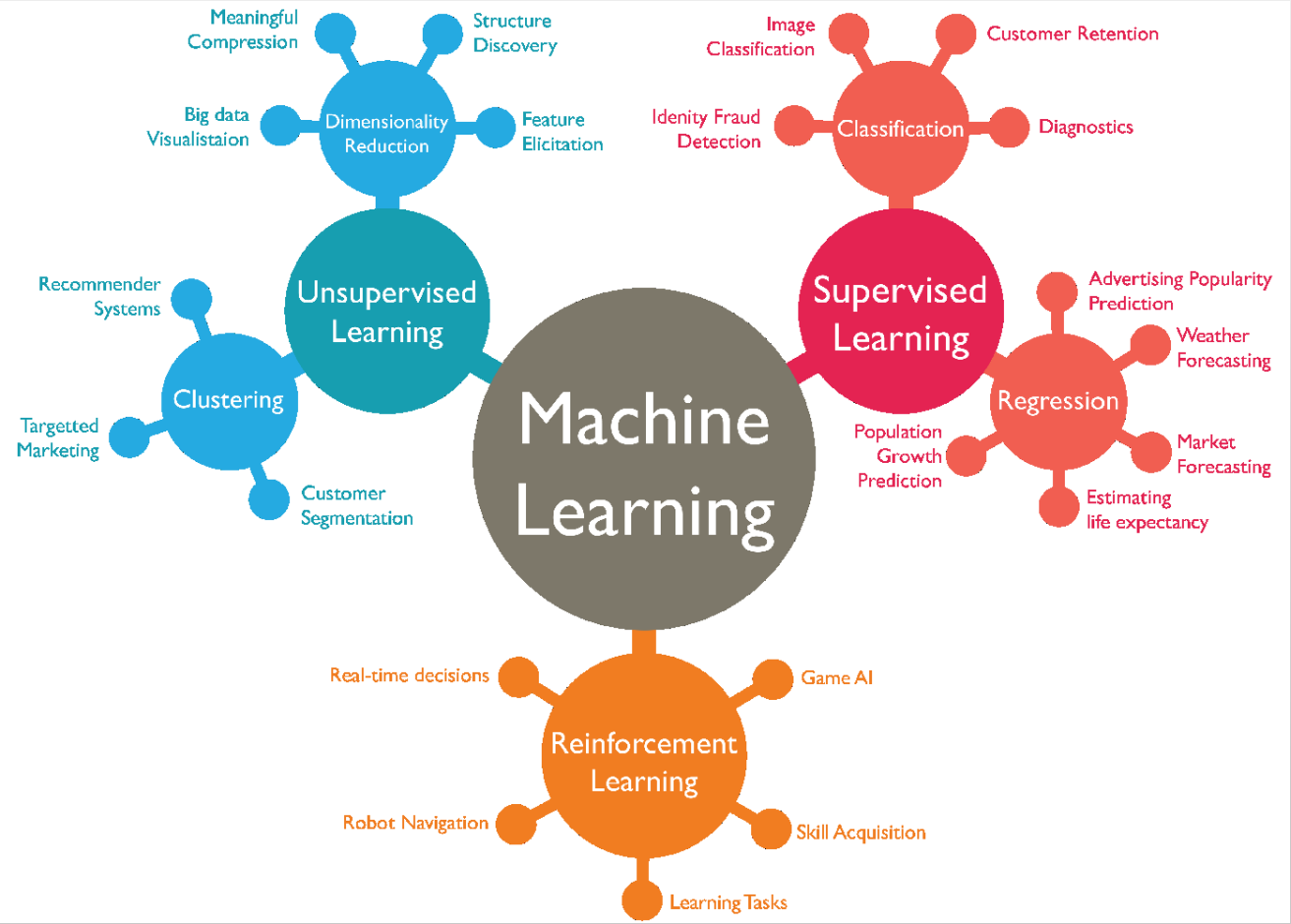


"Is it just one big Sears Catalog?"

– Gerry Fodor

1. Unsupervised Learning: $\mathrm{D} = \{X\}_{i=1}^{N}$
2. Supervised Learning: $\mathrm{D} = \{X, y\}_{i=1}^{N}$
3. Reinforcement Learning:

$$S = \{s\}_{i=1}^{N}, A = \{a\}_{i=1}^{M}, P(s_{t+1}|s_t, a)$$



THE BIG PICTURE

"Is it just one big Sears Catalog?"

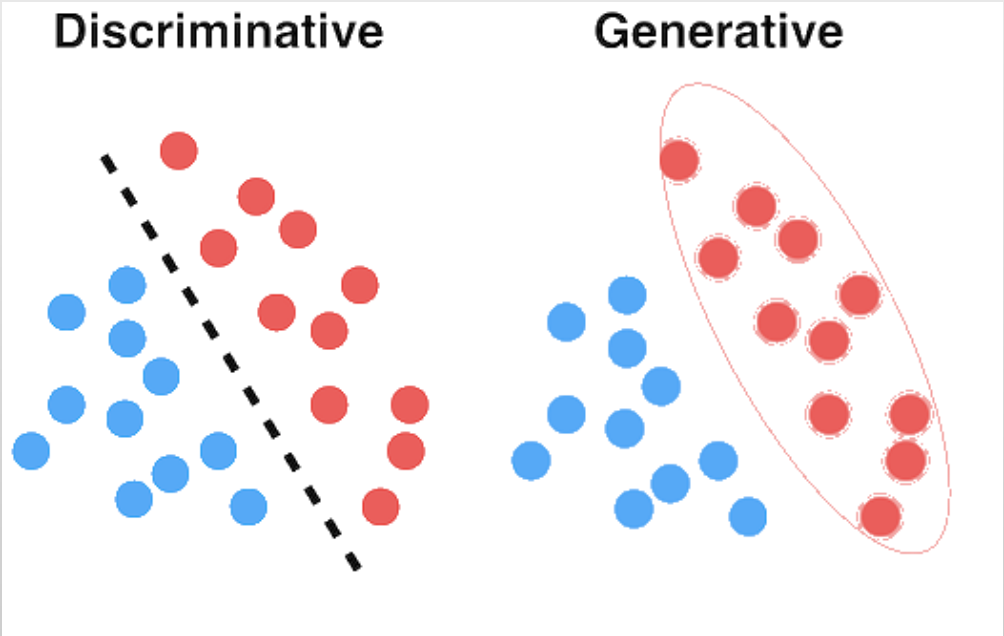
– Gerry Fodor

1. Discriminative models: $P(y|x)$

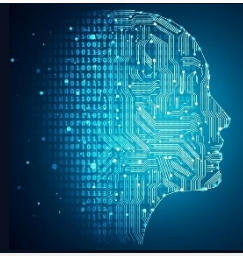
   (example: image classification)

2. Generative models: $P(x, y)$

   (example: art generation)

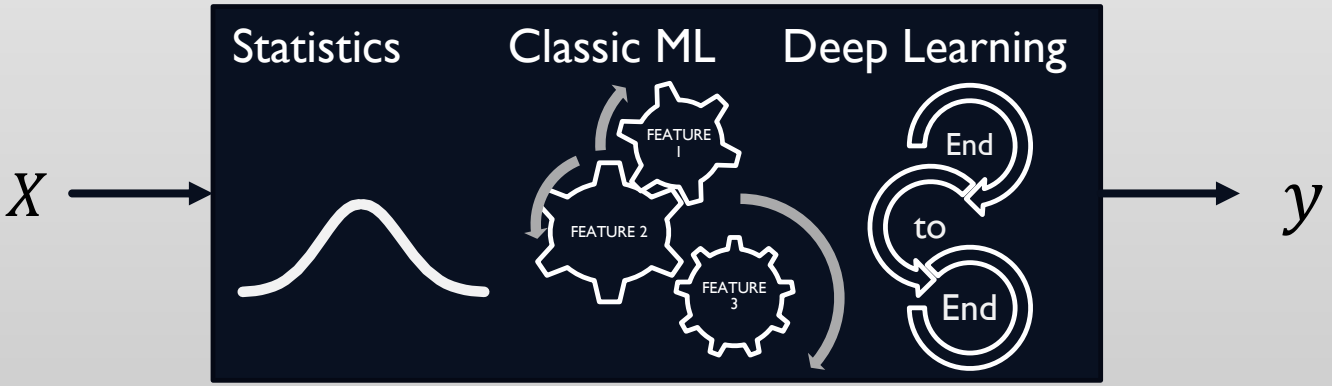


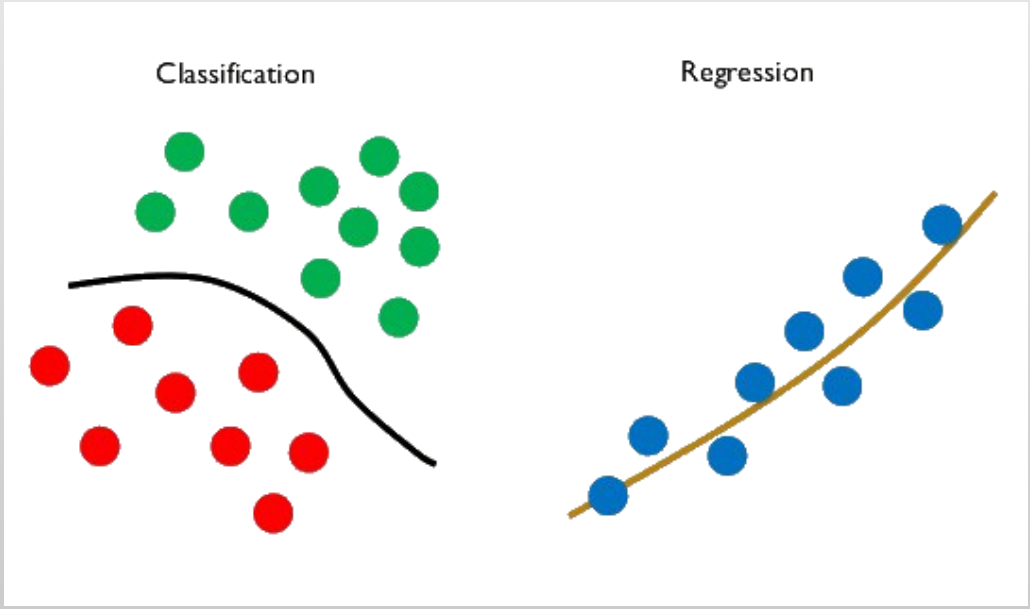
THE BIG PICTURE



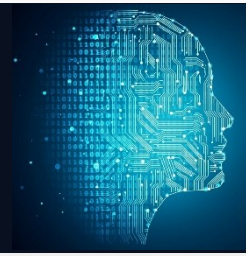
Statistics vs Classic ML vs Deep Learning

Model:

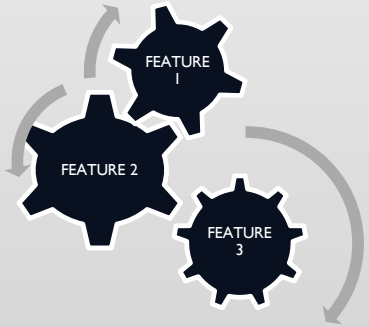




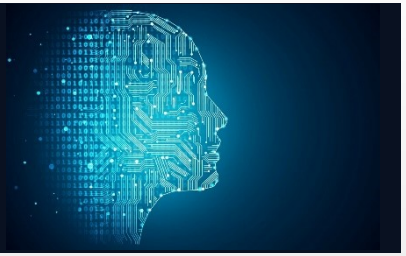
THE BIG PICTURE

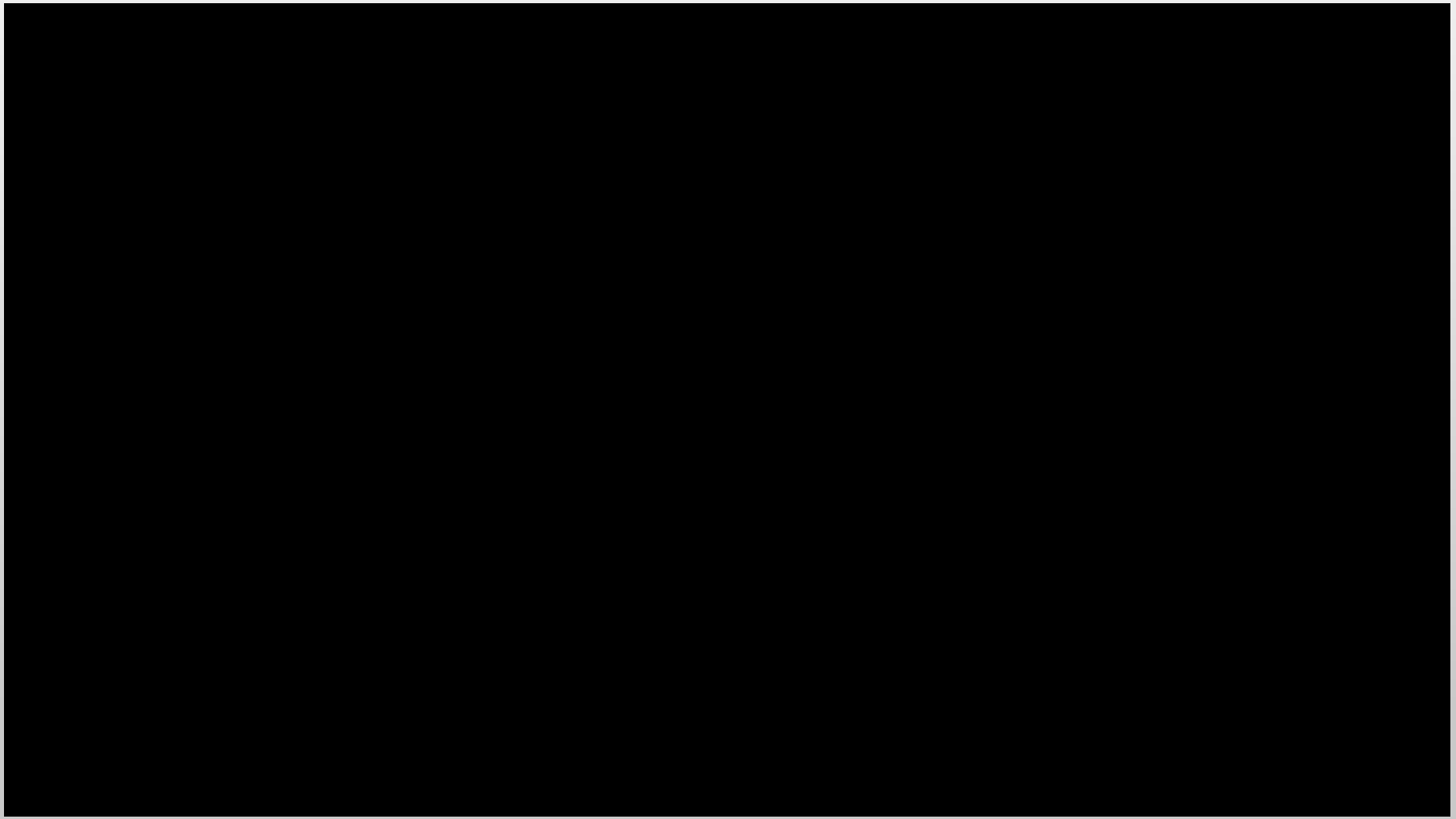


Feature Engineering:

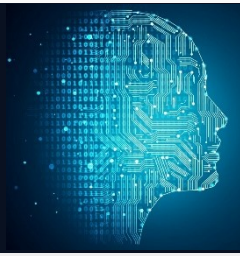


USE CASES: MUSIC CLASSIFICATION

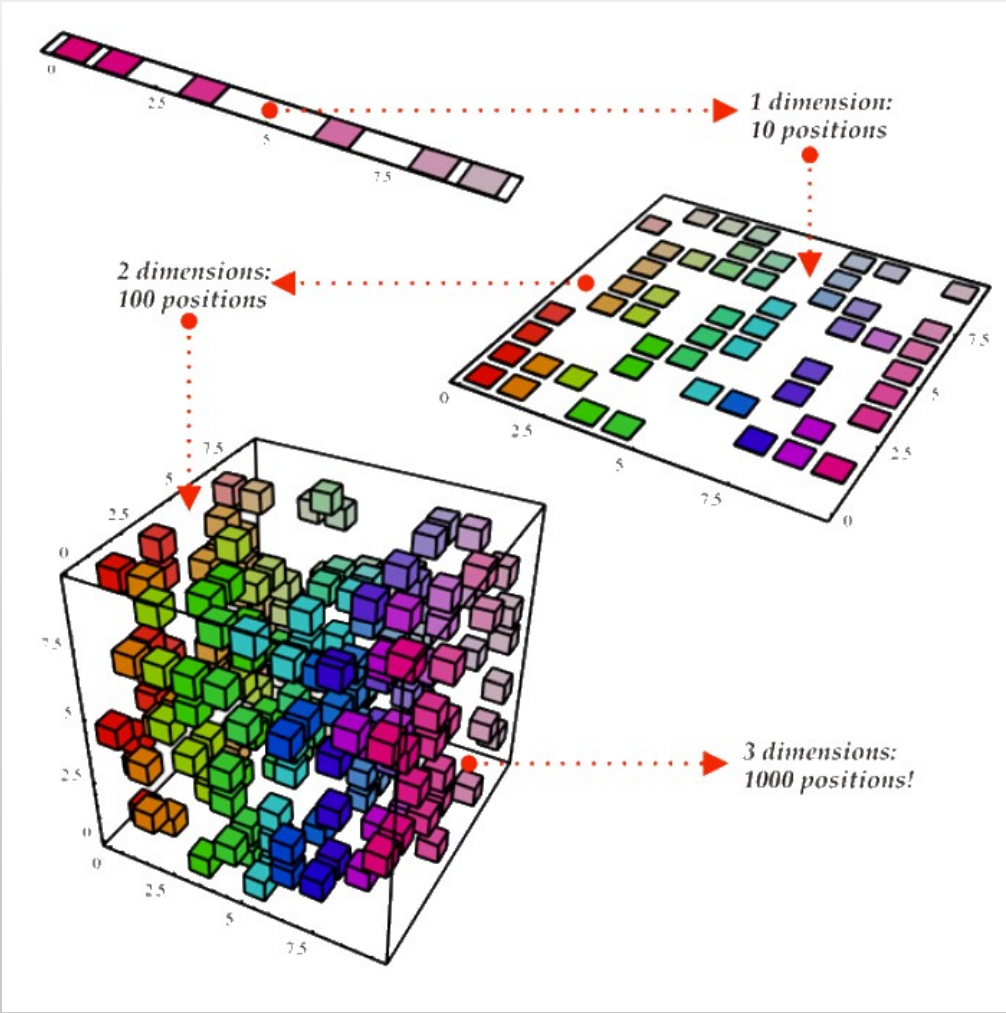




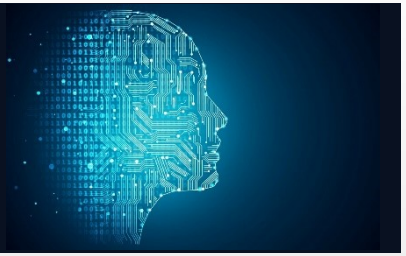
EMBEDDING SPACE

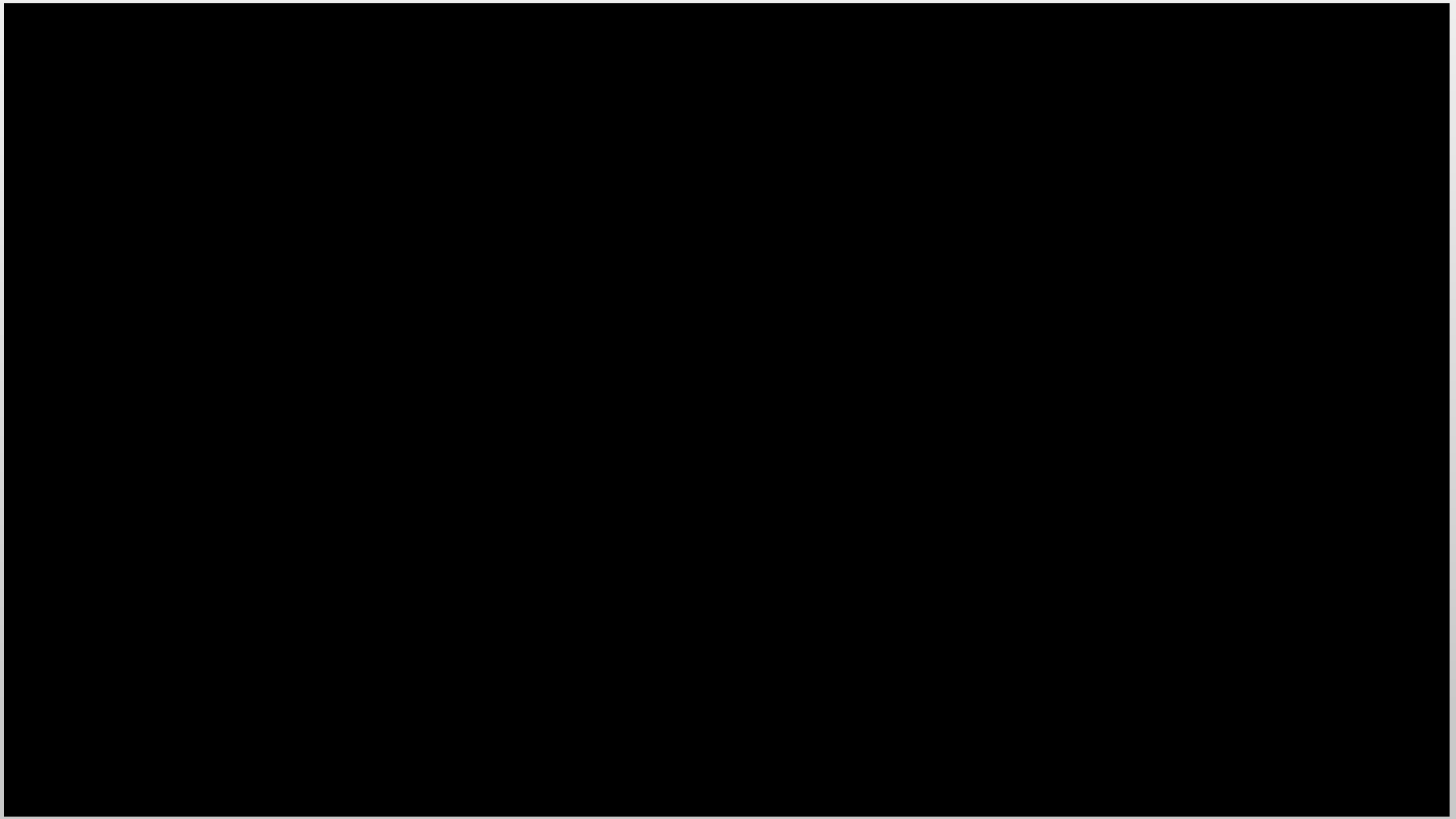


- Dimensionality Reduction
- Manifold Embedding
- Feature Extraction

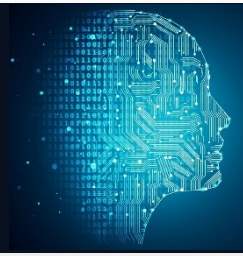


USE CASES: DATA VISUALIZATION





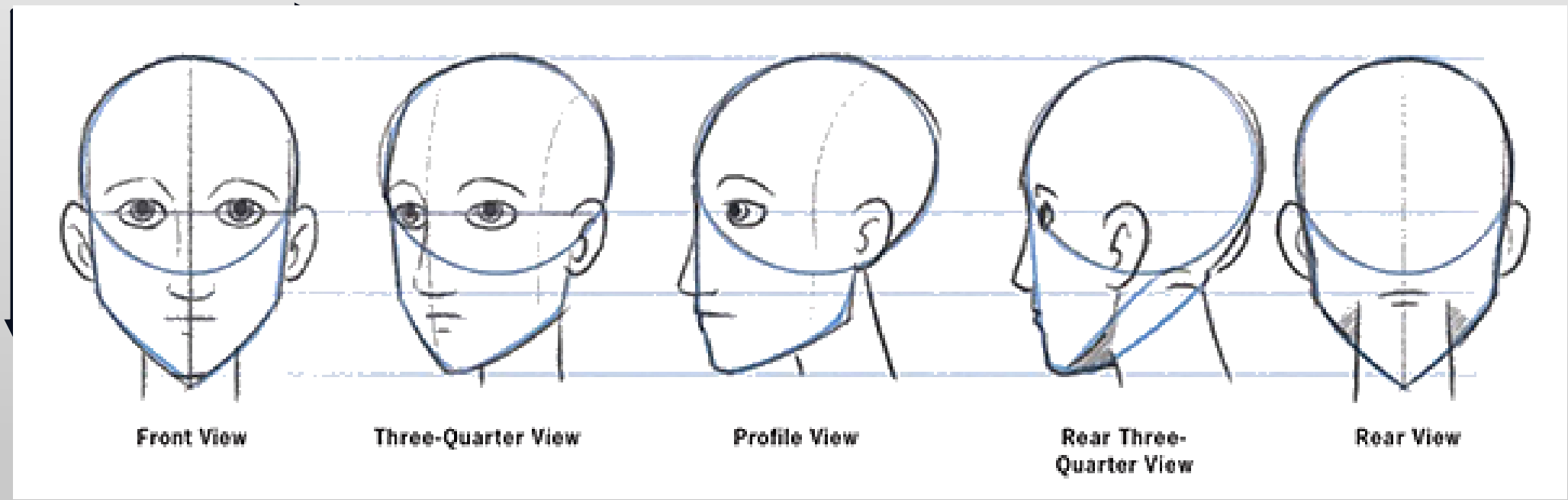
EMBEDDING SPACE



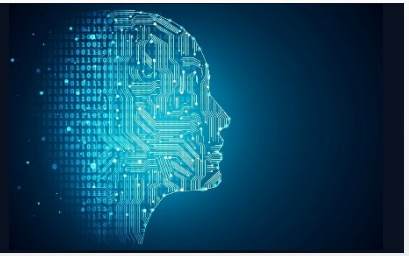
What is the dimension of the problem? (series of images)

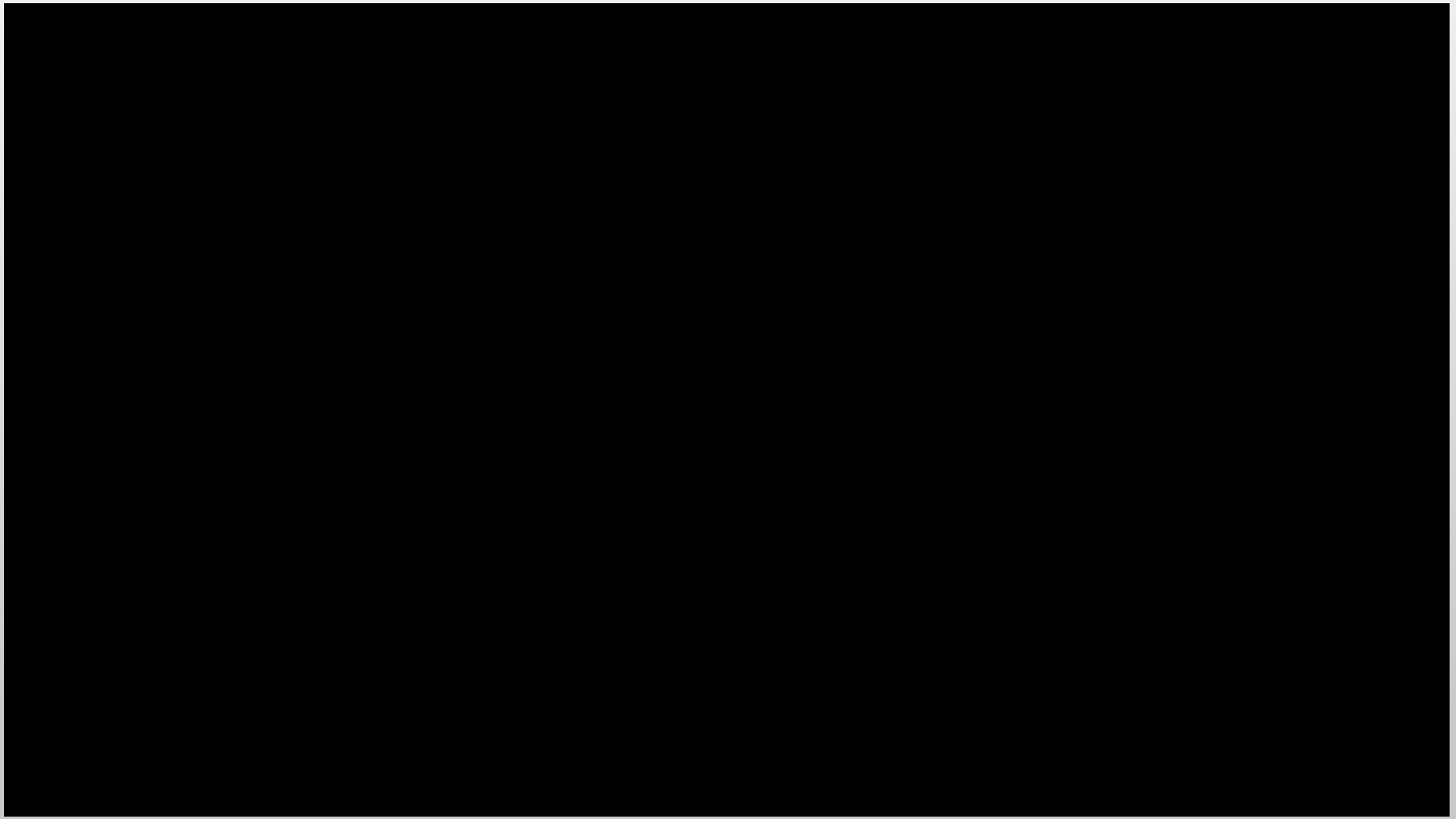
100 pixels

100 pixels

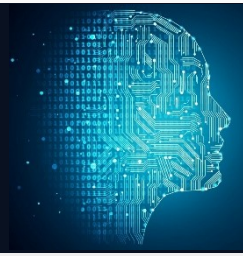


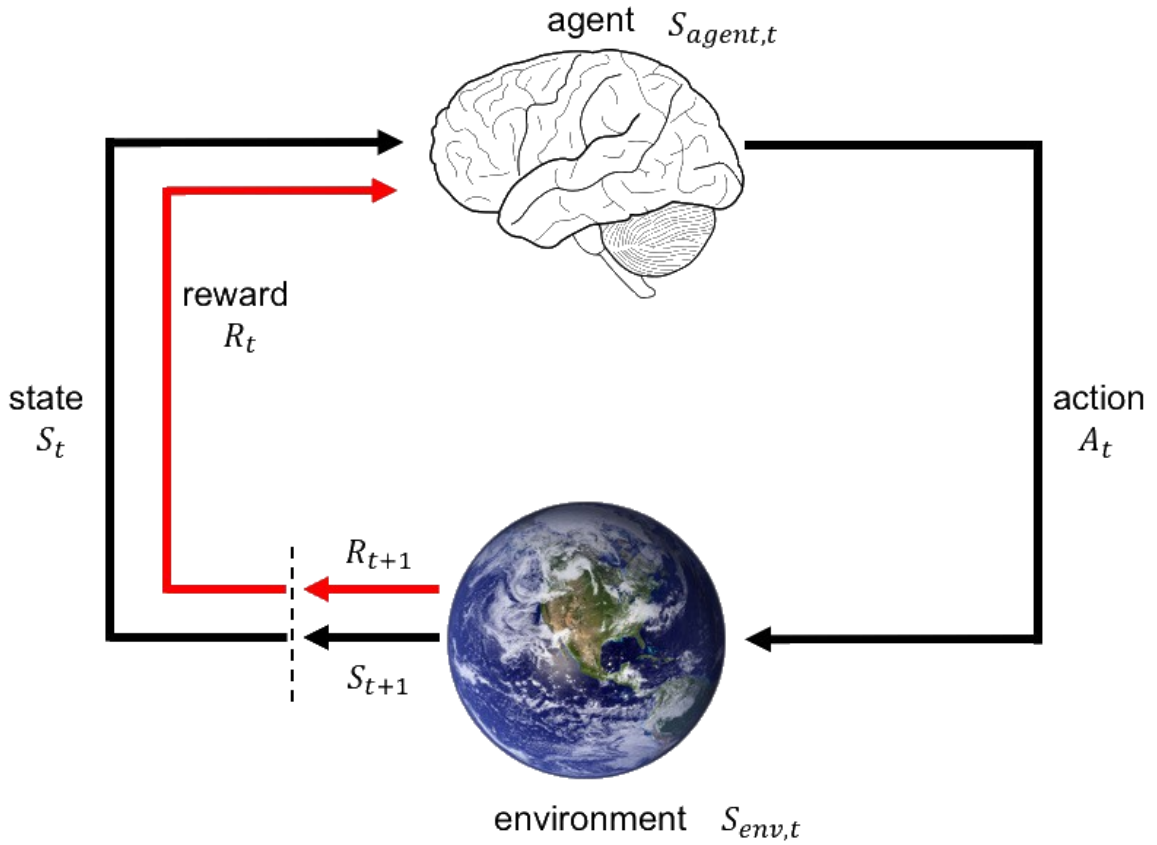
USE CASES: INTELLIGENT AGENT



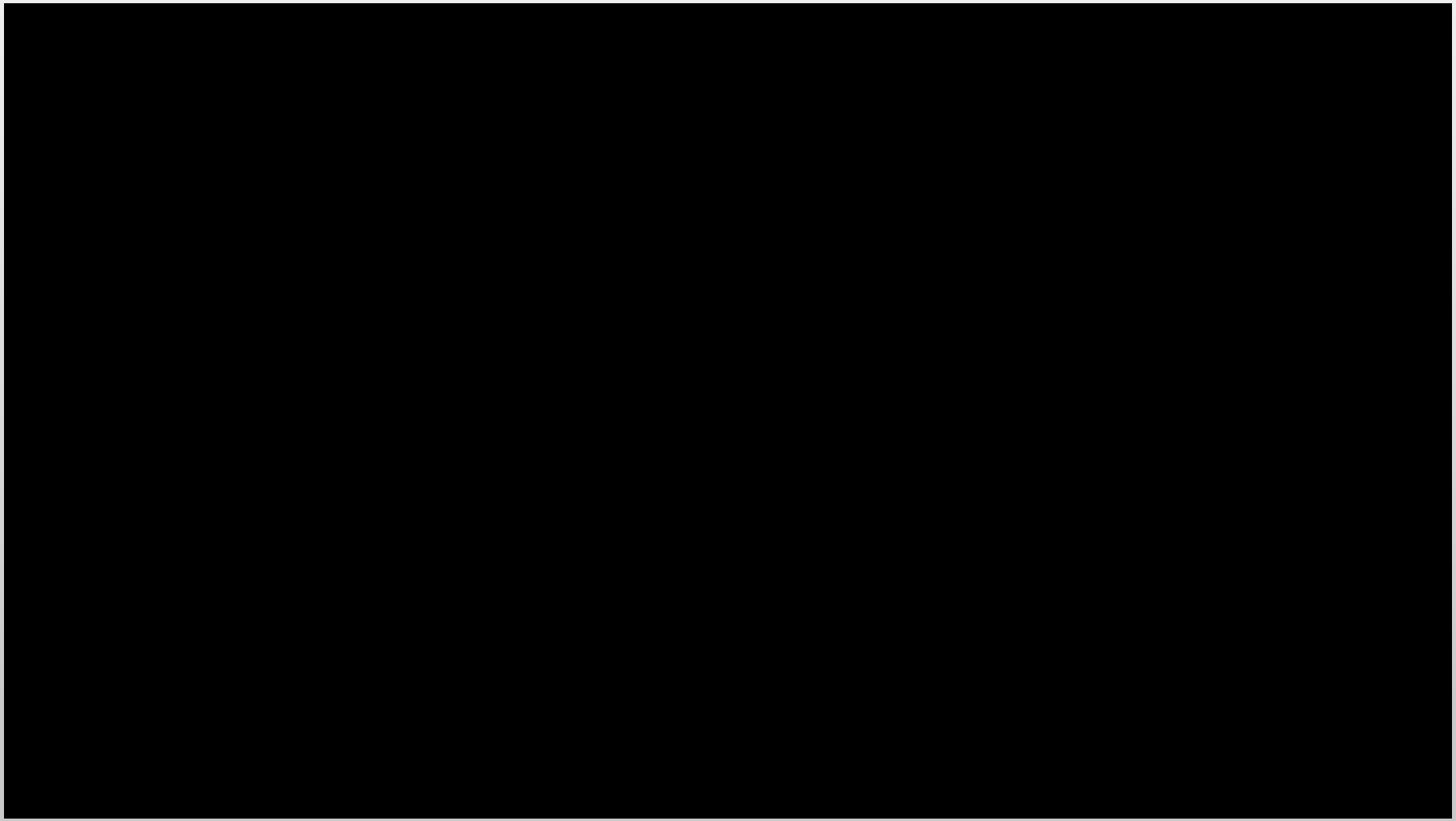


EMBEDDING SPACE - DECISIONS IN REAL TIME!



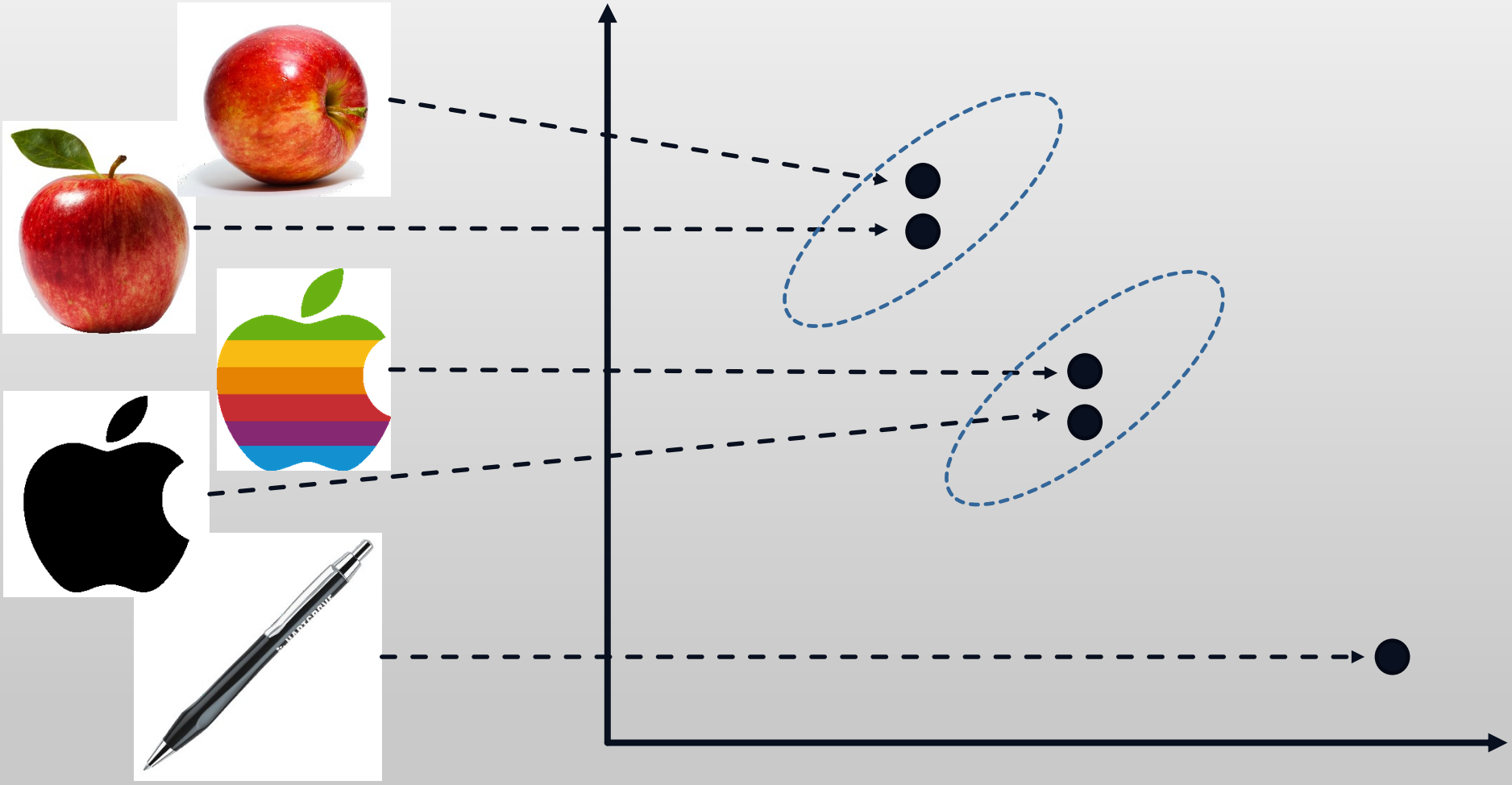


USE CASES: AUTONOMOUS VEHICLE - DARPA CHALLENGE

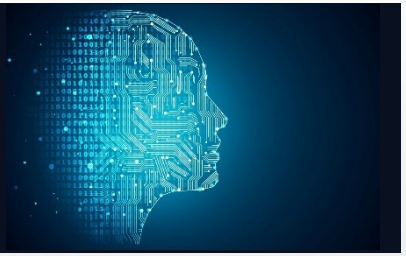


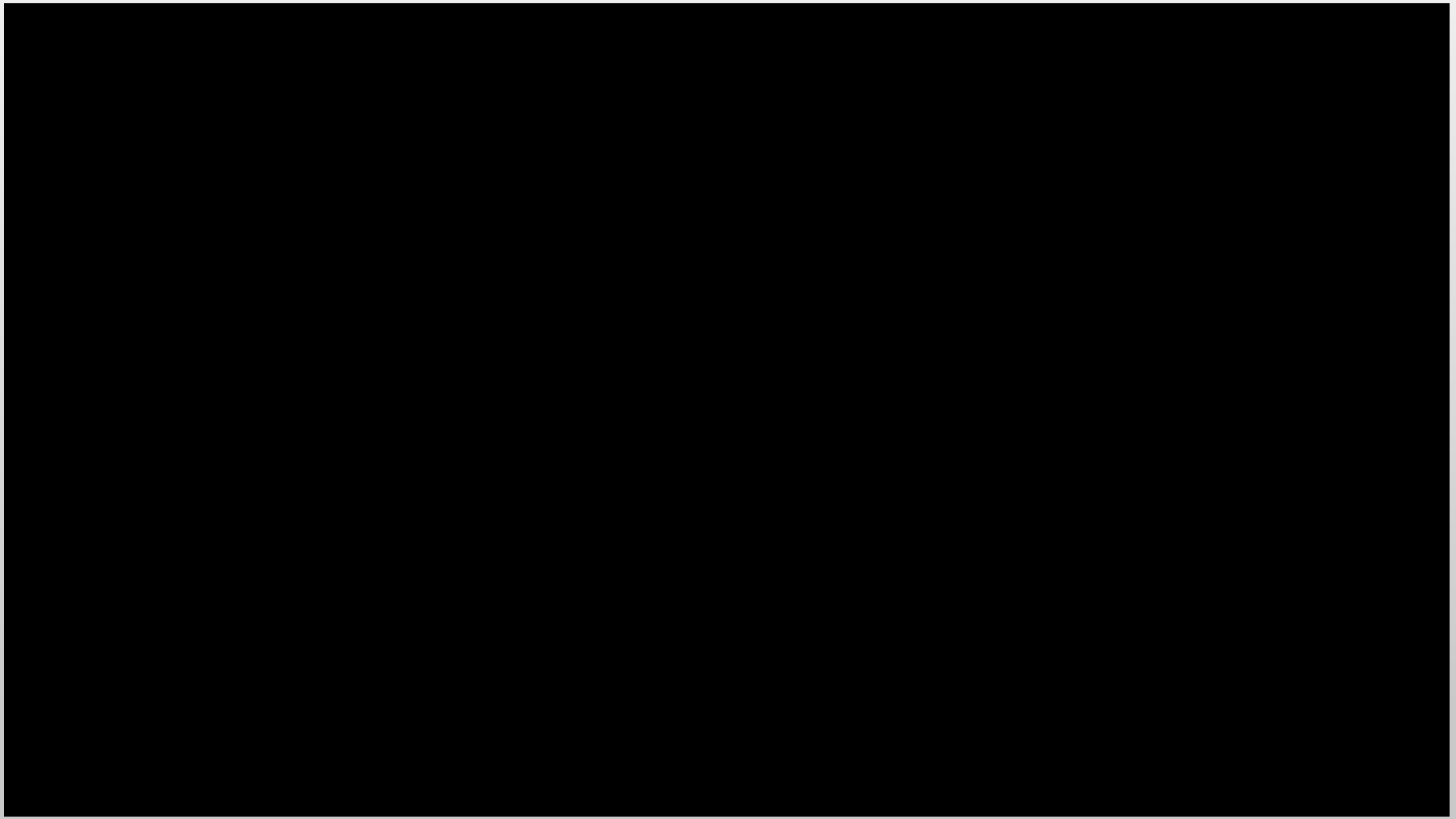
USE CASES: AUTONOMOUS VEHICLE

SIMILARITY AND EMBEDDING ARE IMPORTANT

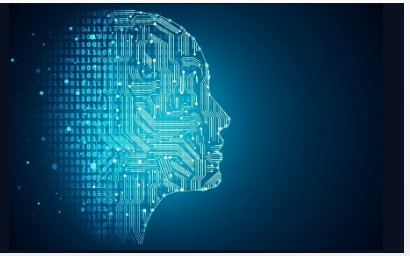


USE CASES: ART GENERATION





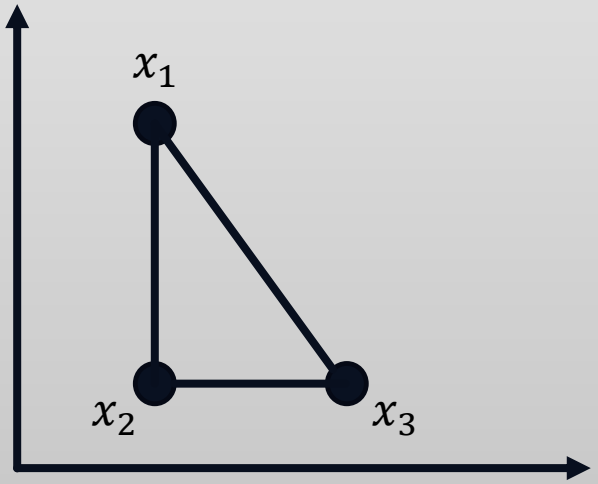
SIMILARITY AND EMBEDDING ARE IMPORTANT

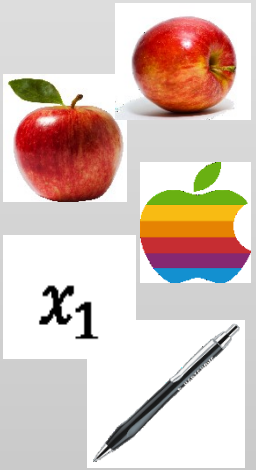


A metric space (X, d) is a set for which distances between all members of the set are defined.

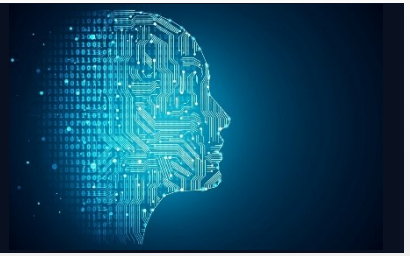
d is a function such that for any $x_1, x_2, x_3 \in X$ the following holds:

1. $d(x_1, x_2) \geq 0$ (non-negativity)
2. $d(x_1, x_2) = 0 \leftrightarrow x_1 = x_2$ (discernable)
3. $d(x_1, x_2) = d(x_2, x_1)$ (symmetric)
4. $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$





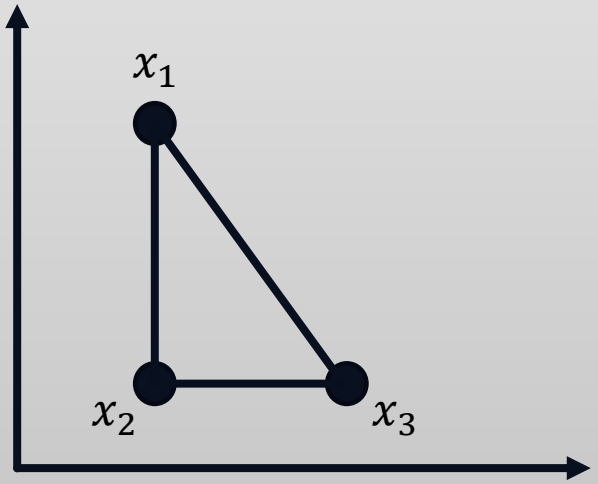
SIMILARITY AND EMBEDDING ARE IMPORTANT



A pseudo metric space (X, d) does not require (2).

1. $d(x_1, x_2) \geq 0$ (non-negativity)
2. ~~$d(x_1, x_2) = 0 \leftrightarrow x_1 = x_2$ (discernable)~~
3. $d(x_1, x_2) = d(x_2, x_1)$ (symmetric)
4. $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$

$\rightarrow d(x_1, x_2) = 0 \; for \; x_1 \neq x_2$

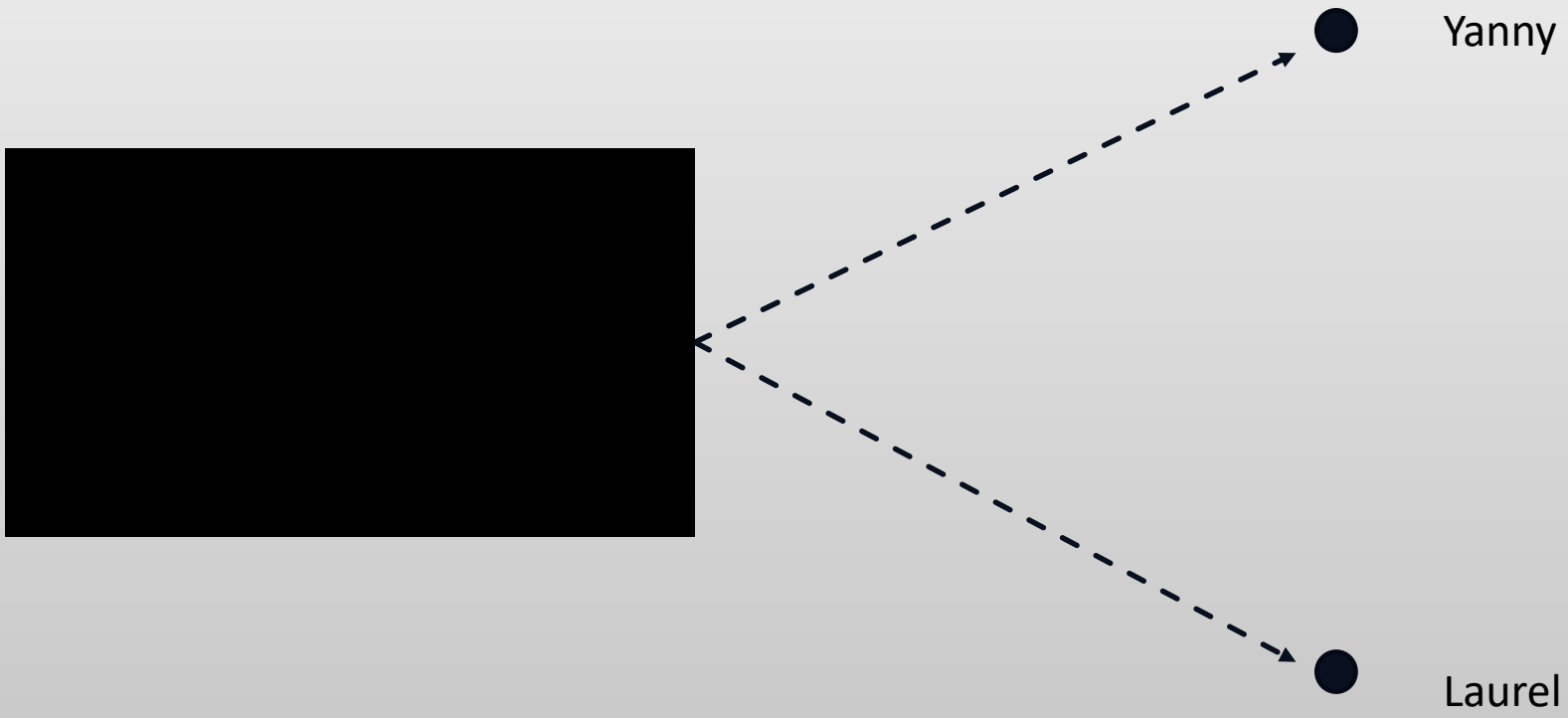


SIMILARITY AND EMBEDDING ARE IMPORTANT

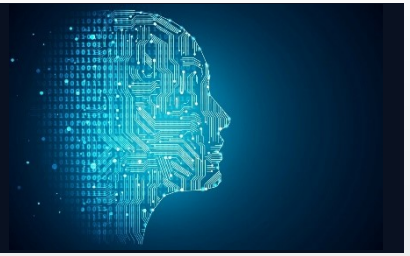
White and Gold

Blue and Black

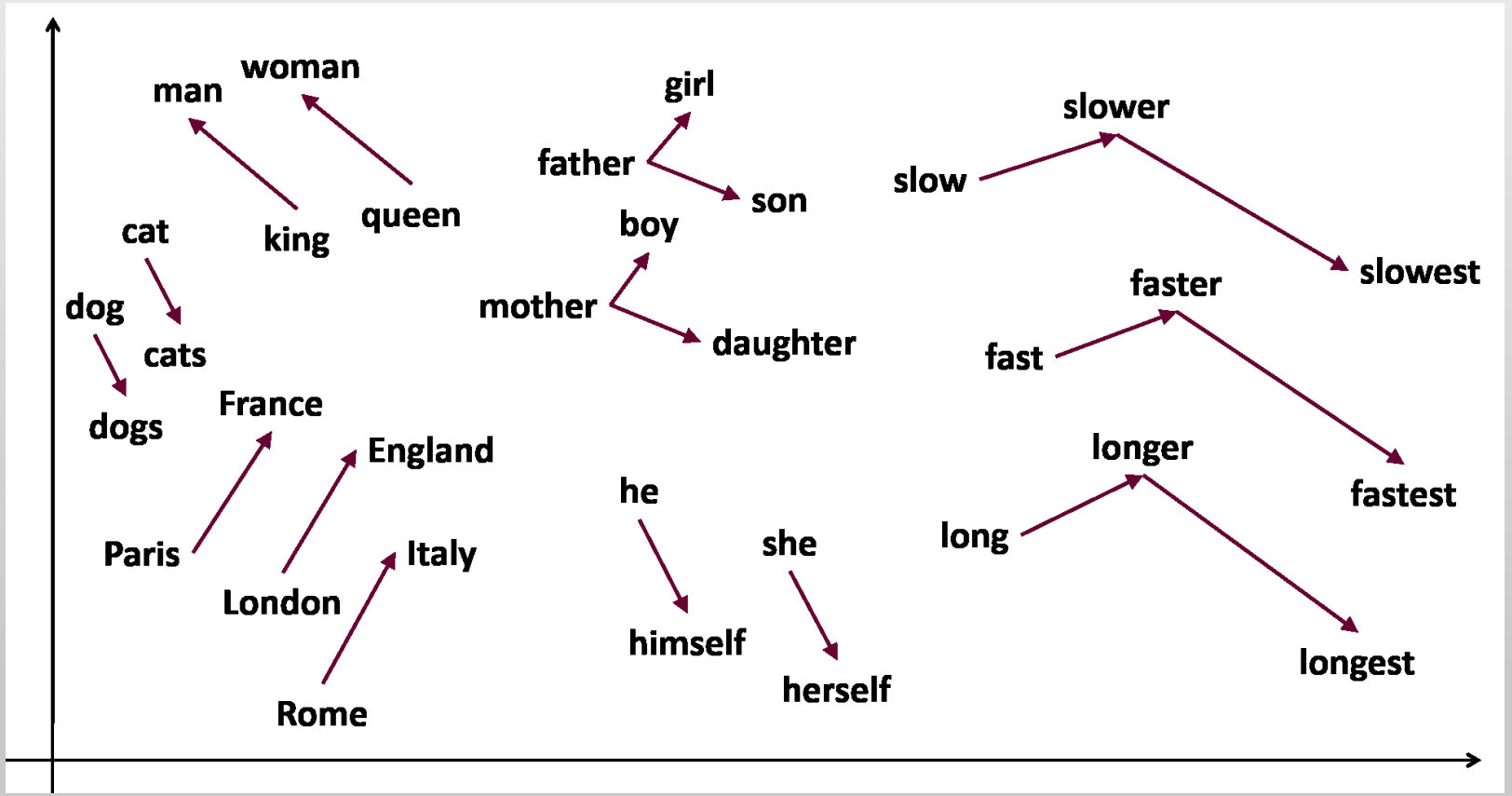
SIMILARITY AND EMBEDDING ARE IMPORTANT



SIMILARITY AND EMBEDDING ARE IMPORTANT



Embedding space in natural language processing (NLP)

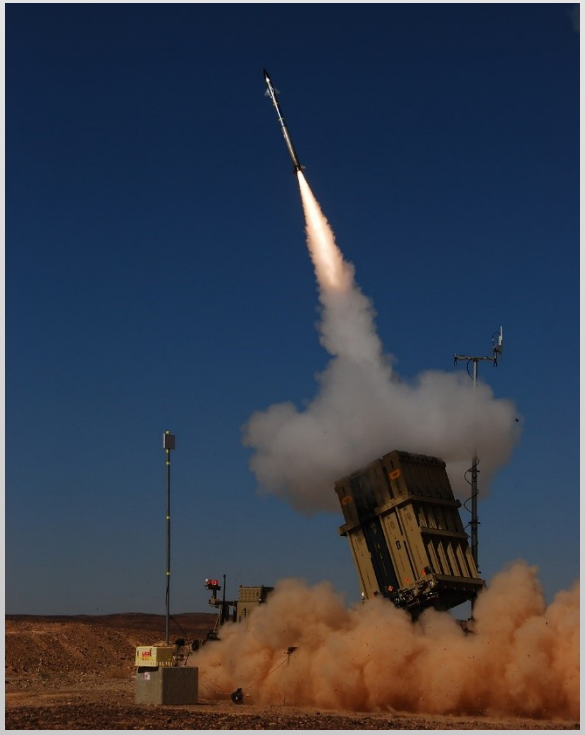


TECHNOLOGY TRENDS – COMPUTER VISION

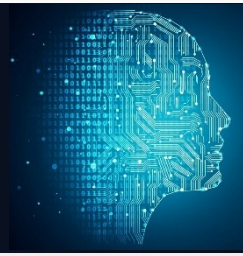
IphoneX Face ID

Mobileye

Iron Dome



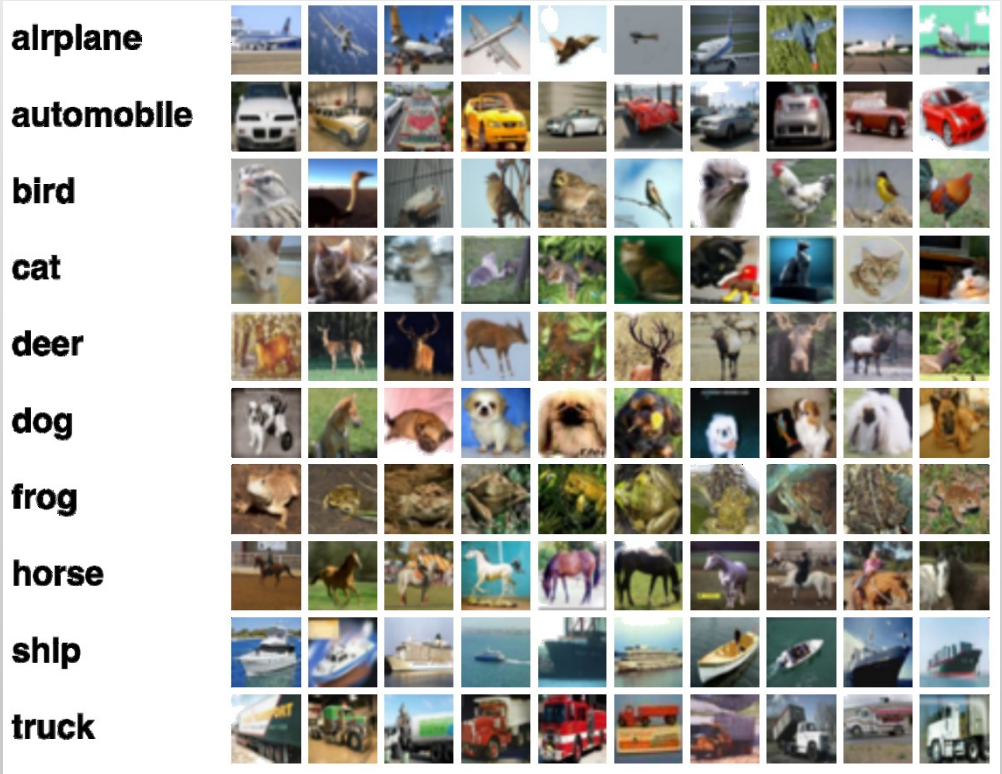
TECHNOLOGY TRENDS – COMPUTER VISION



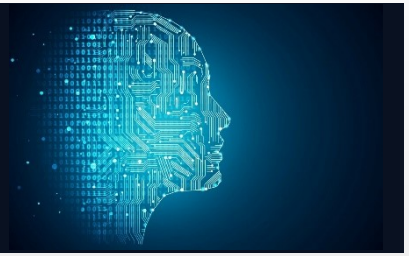
ImageNet Challenge 2012

- ~14 million labeled images
- Images gathered from the internet
- Human labels via Amazon Turk
- 1.2 million test images, 1000 classes (categories)

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*



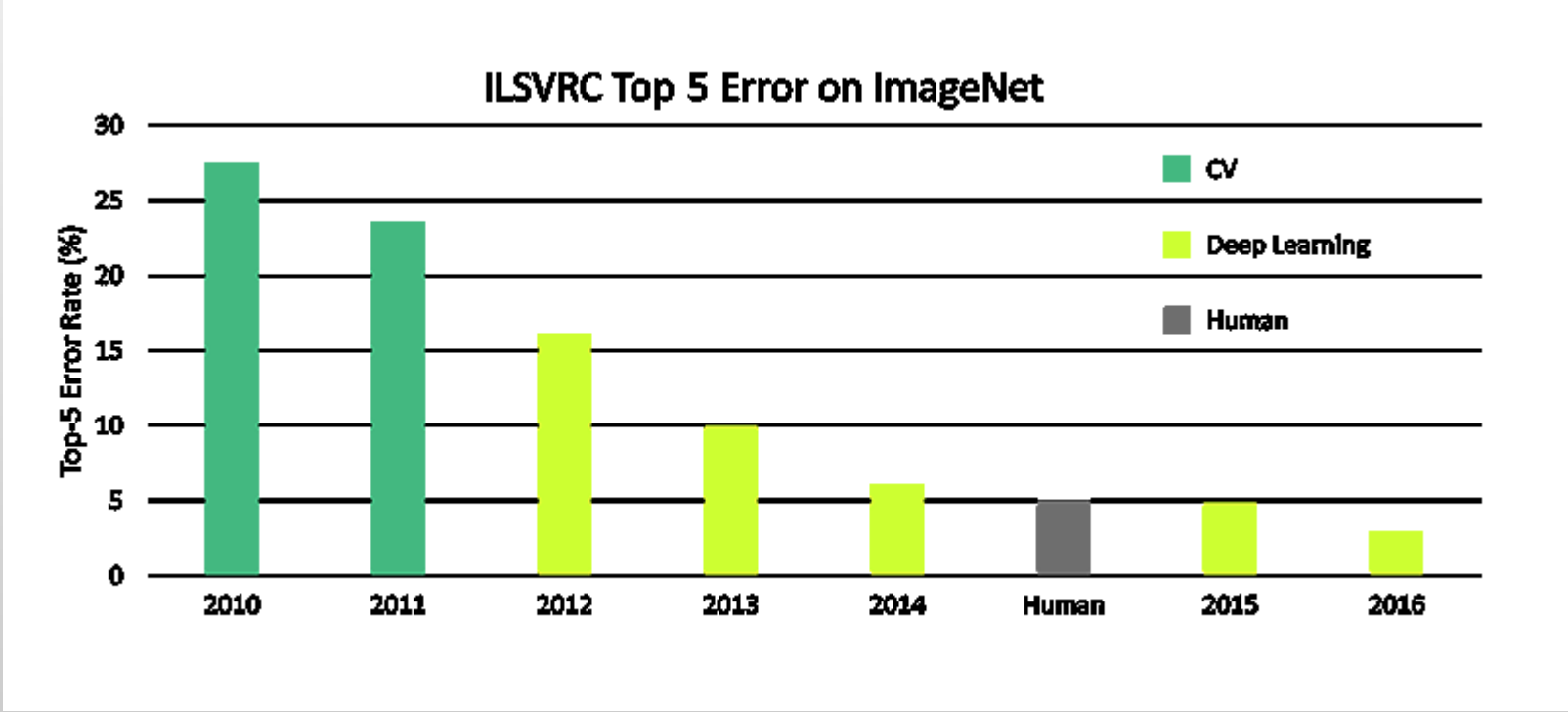
TECHNOLOGY TRENDS – COMPUTER VISION



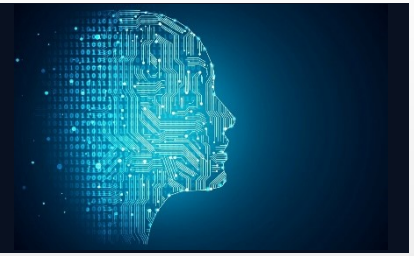
ImageNet Challenge 2012

- ~14 million labeled images
- Images gathered from the internet
- Human labels via Amazon Turk
- 1.2 million test images, 1000 classes (categories)

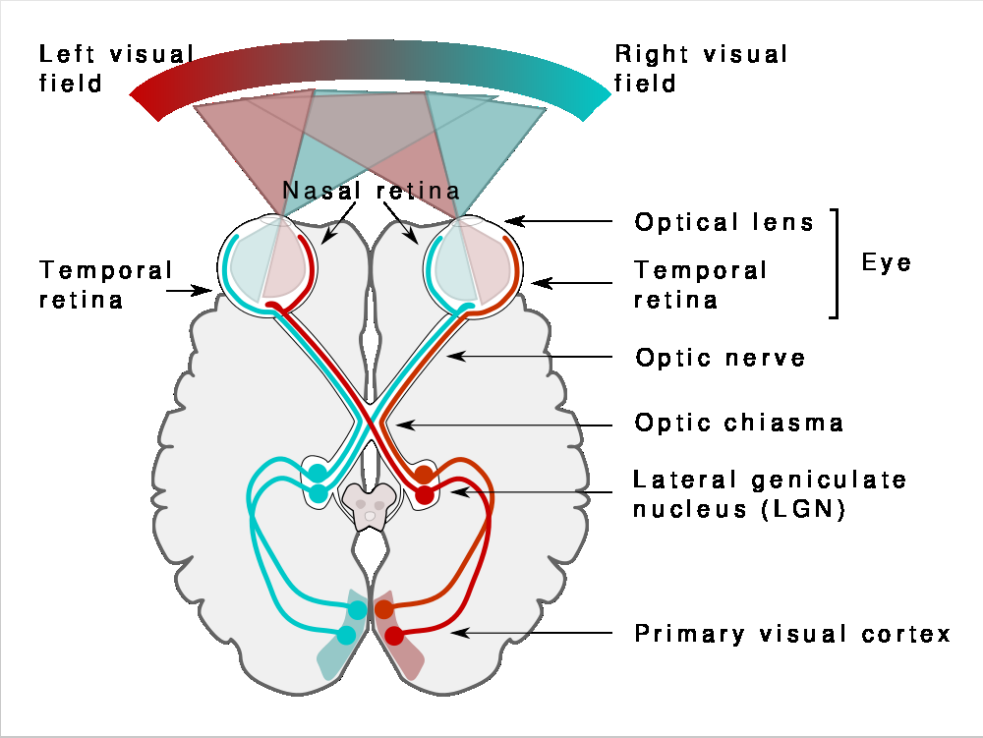
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton: *ImageNet Classification with Deep Convolutional Neural Networks*

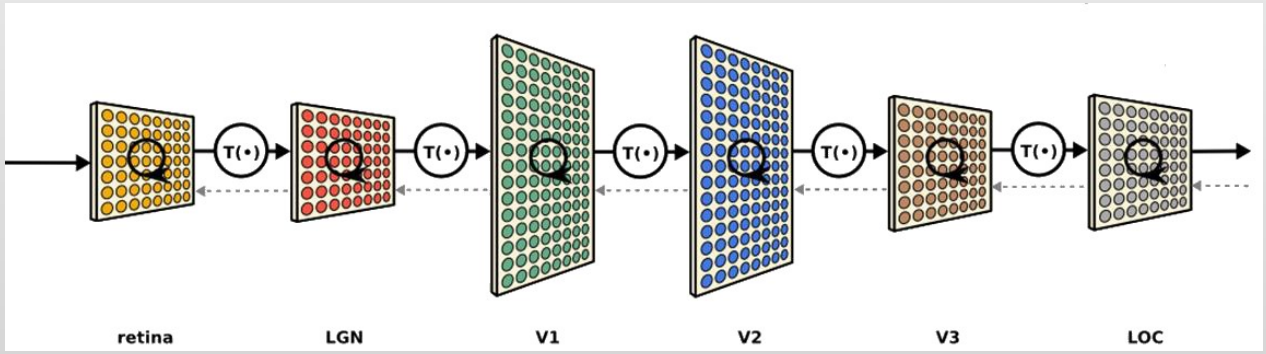


TECHNOLOGY TRENDS – COMPUTER VISION

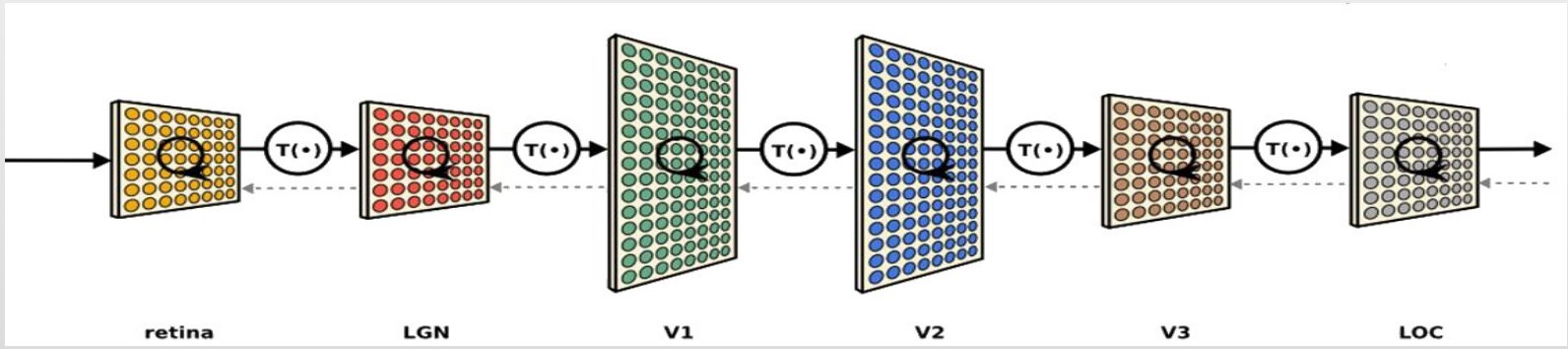


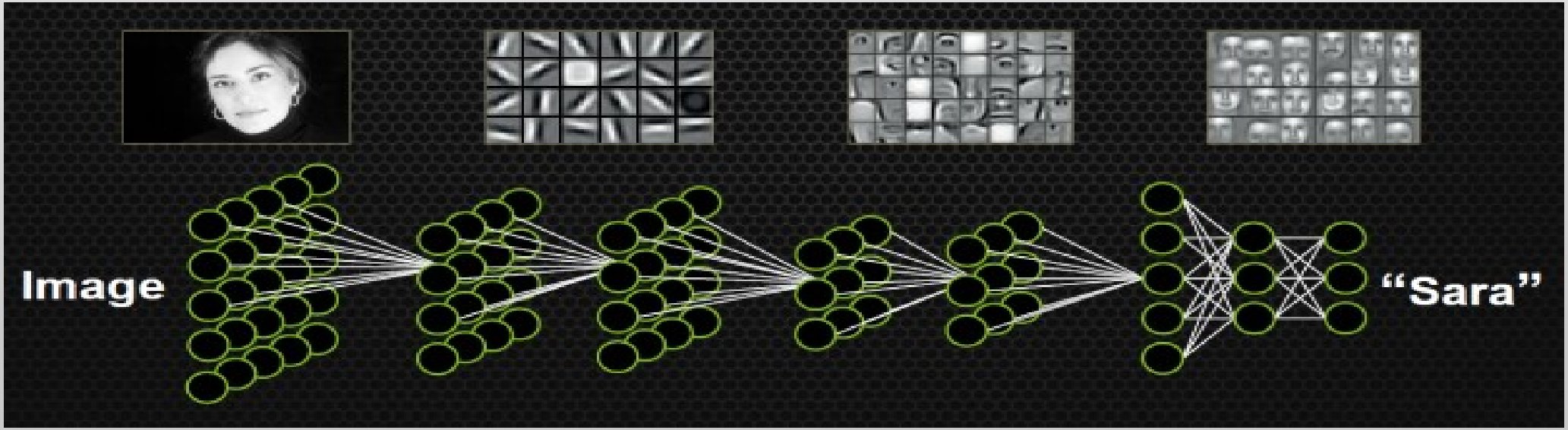
Convolutional Neural Networks, inspired by the human mind:





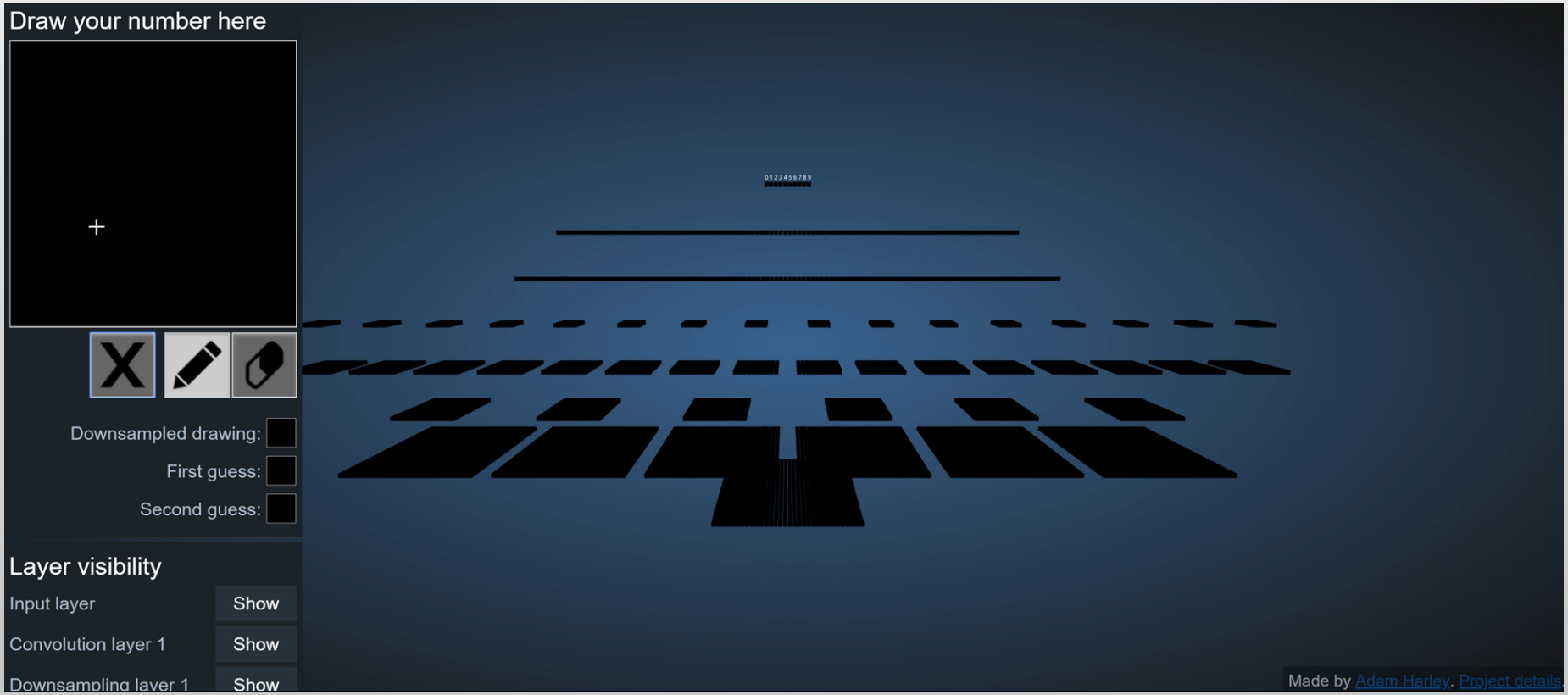
TECHNOLOGY TRENDS – COMPUTER VISION

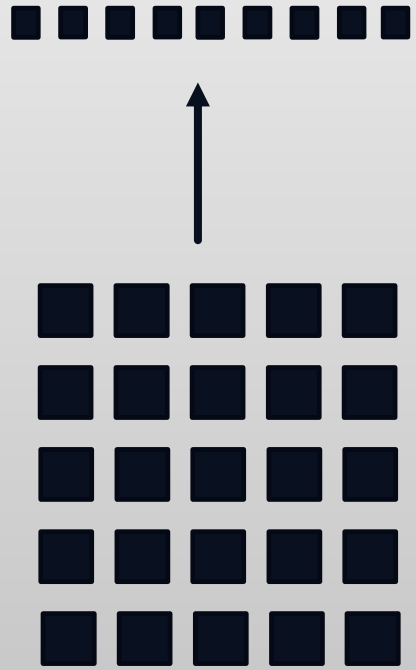




TECHNOLOGY TRENDS – COMPUTER VISION

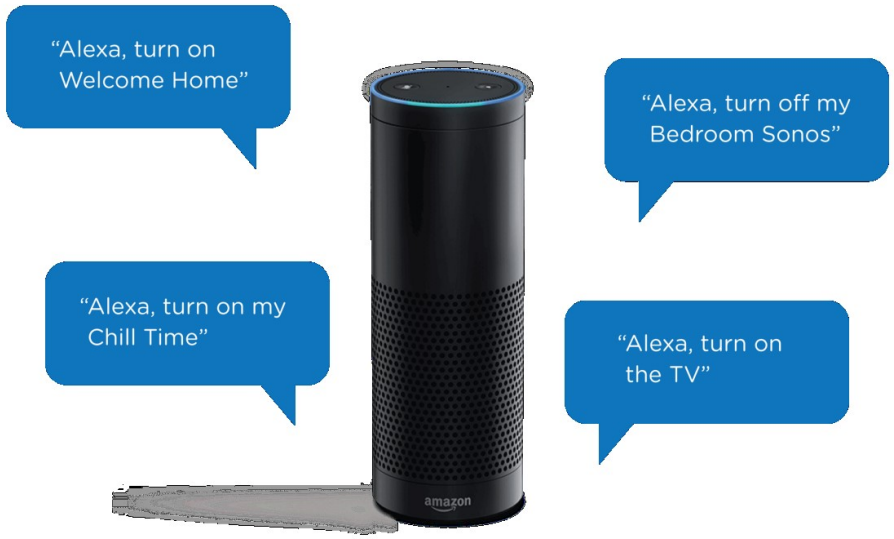
ConvNet Visualization (link)





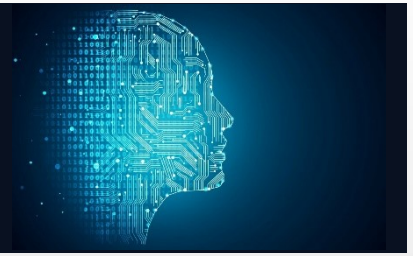
TECHNOLOGY TRENDS – VOICE RECOGNITION

Amazon Alexa



Google Translate

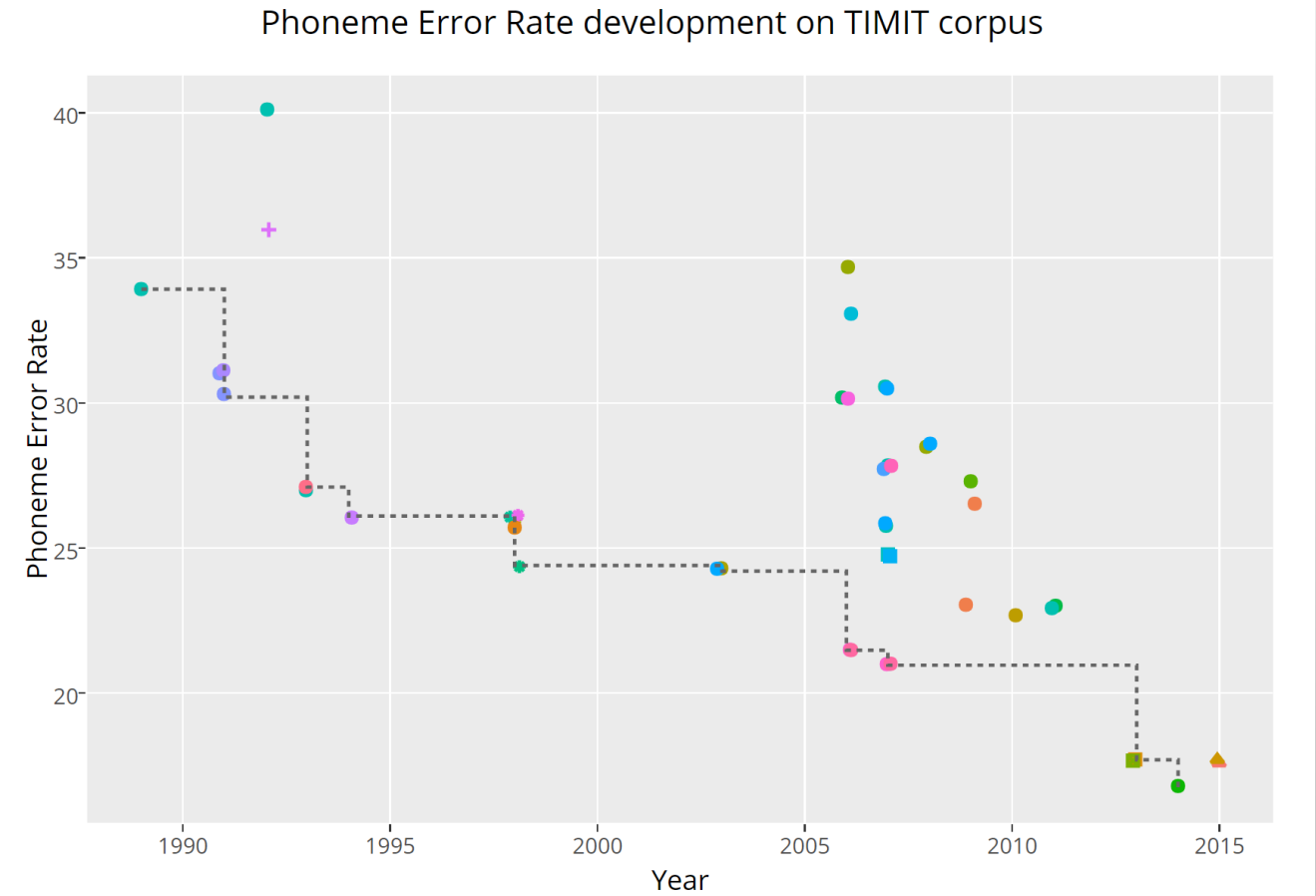
TECHNOLOGY TRENDS – VOICE RECOGNITION



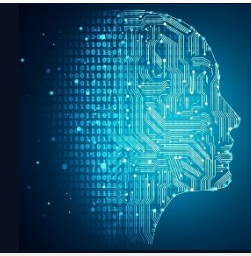
TIMIT (Texas Instruments Massachusetts Institute of Technology) Challenge 2013

- Transcribed corpus of English texts, speakers are from different dialects and sexes.
- Commissioned by DARPA

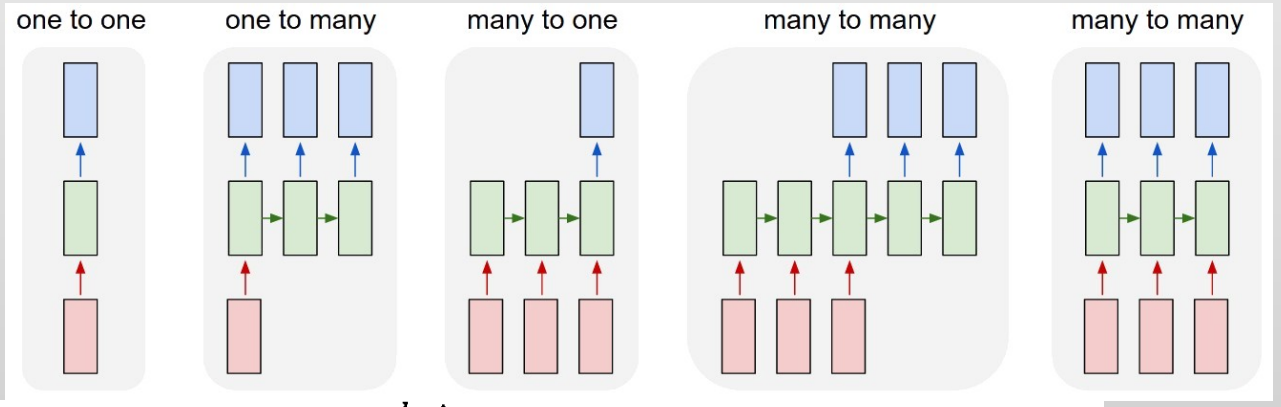
Sepp Hochreiter; Jürgen Schmidhuber (1997): *Long short-term memory*

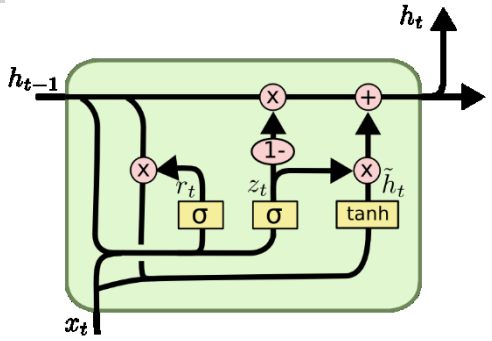


TECHNOLOGY TRENDS – VOICE RECOGNITION



Recurrent Neural Networks, inspired by the human mind:





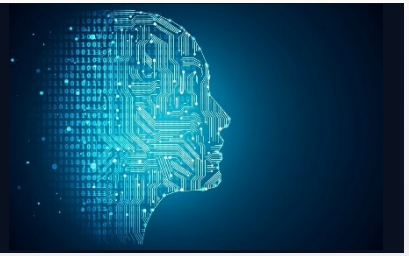
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

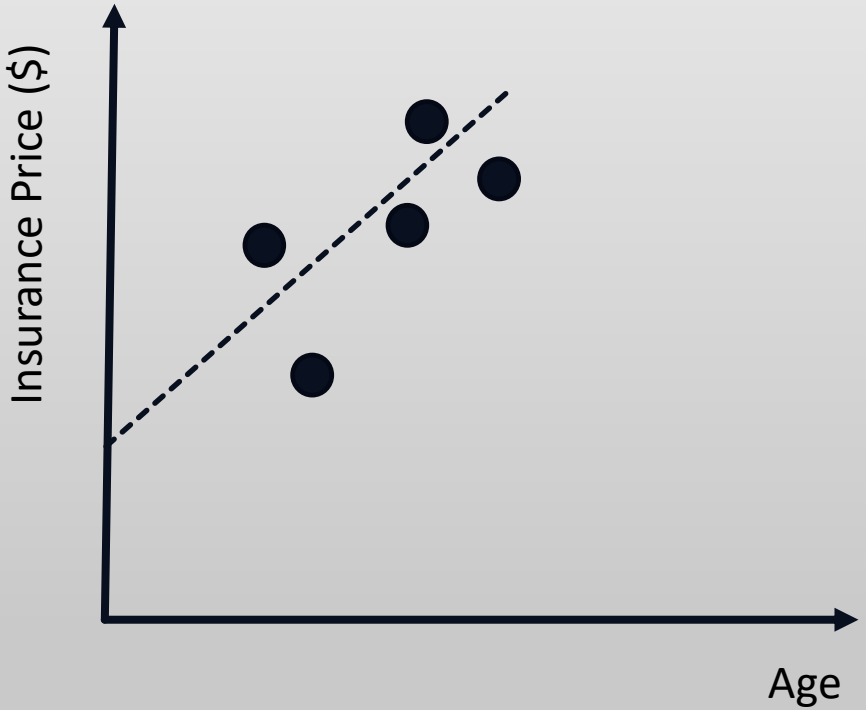
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

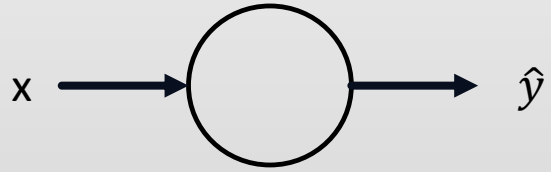
WHAT IS A NEURAL NET?



Define a model:

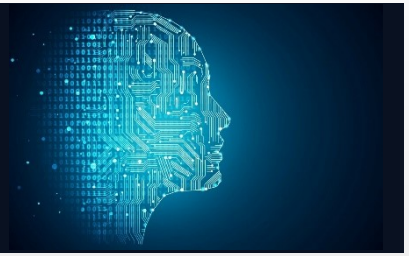
- Linear Regressor





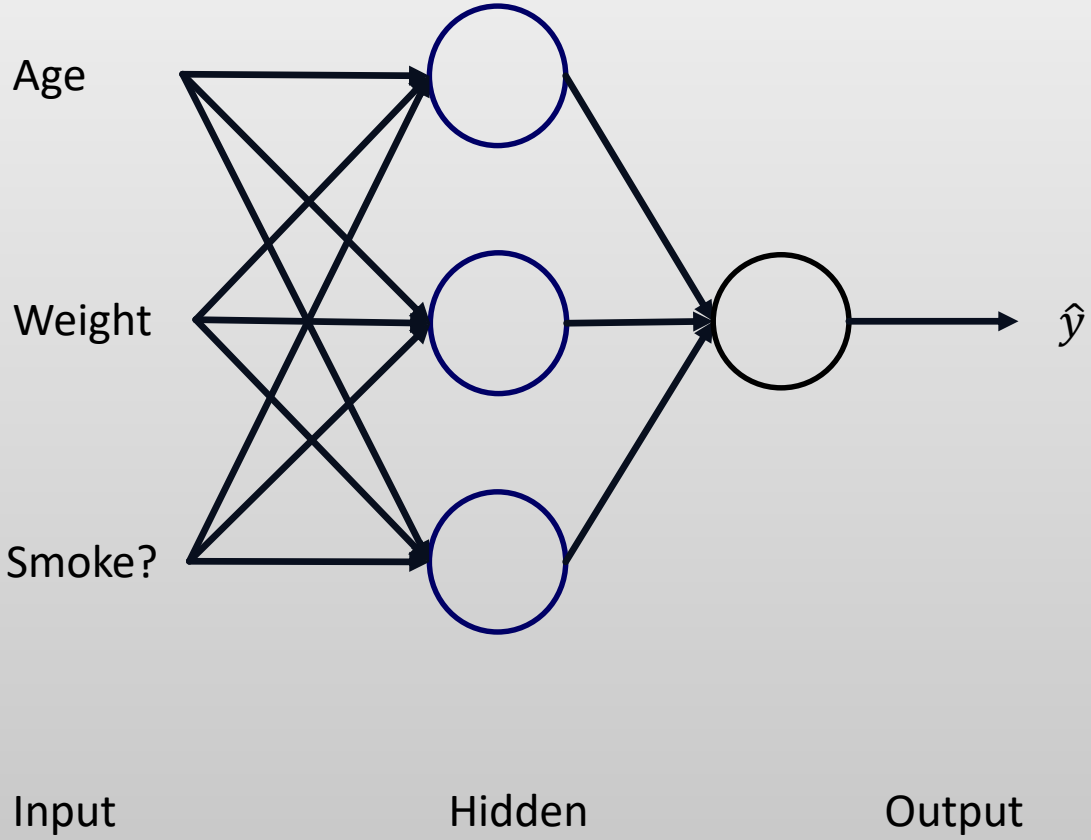
$$\hat{y} = a \cdot x + b$$

WHAT IS A NEURAL NET?



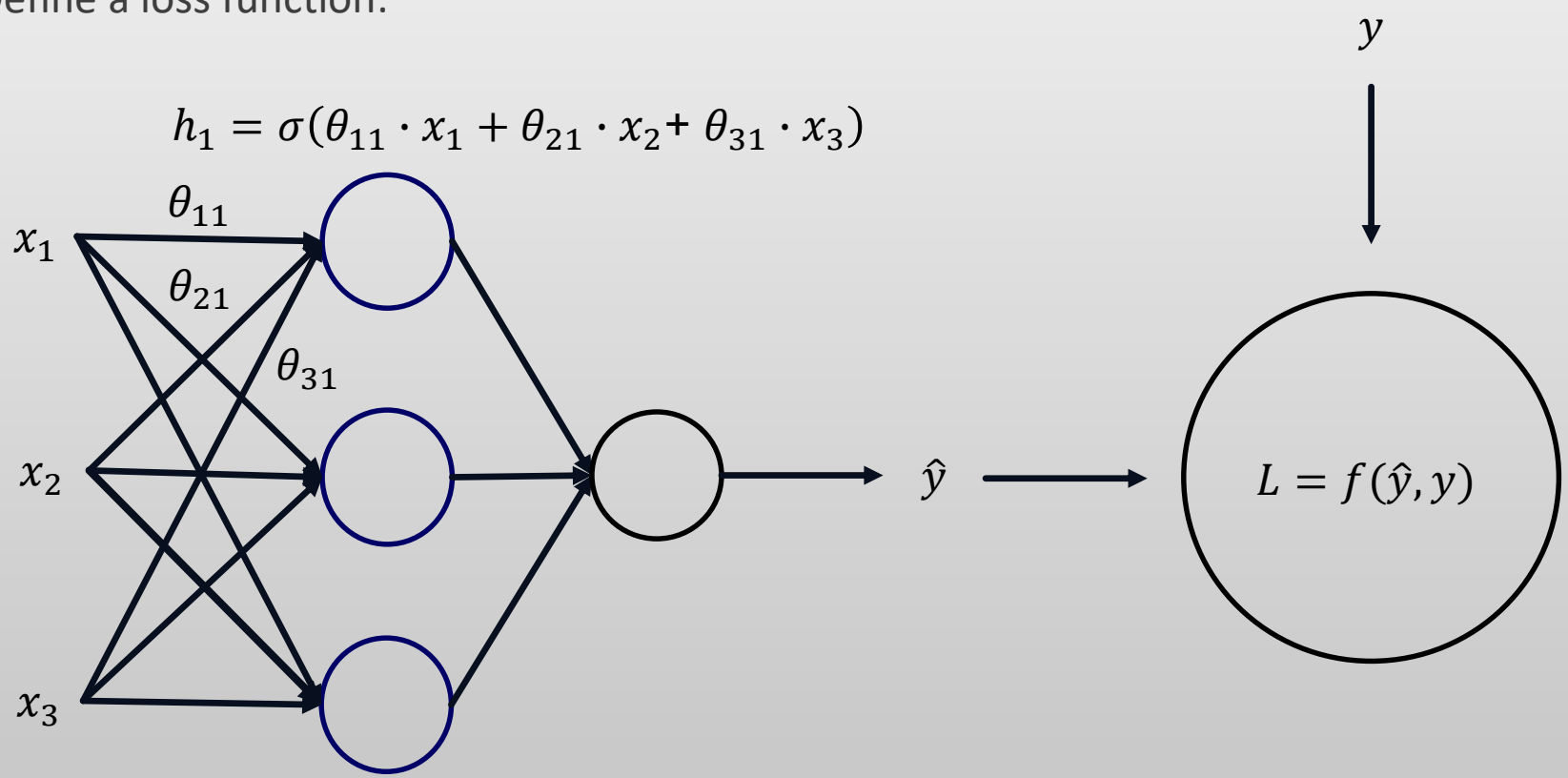
Define a model:

- Neural net – A specific type of computational graph
- A collection of edges and nodes
- A generalization of the linear regressor
- Inspired by the brain?
- “Just curve fitting” (Judea Pearl) ?

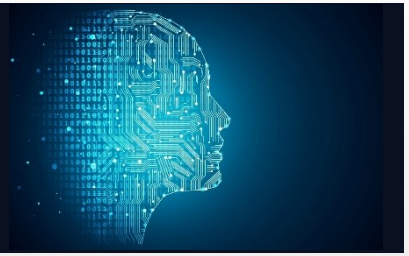


TRAINING A MODEL

Define a loss function:

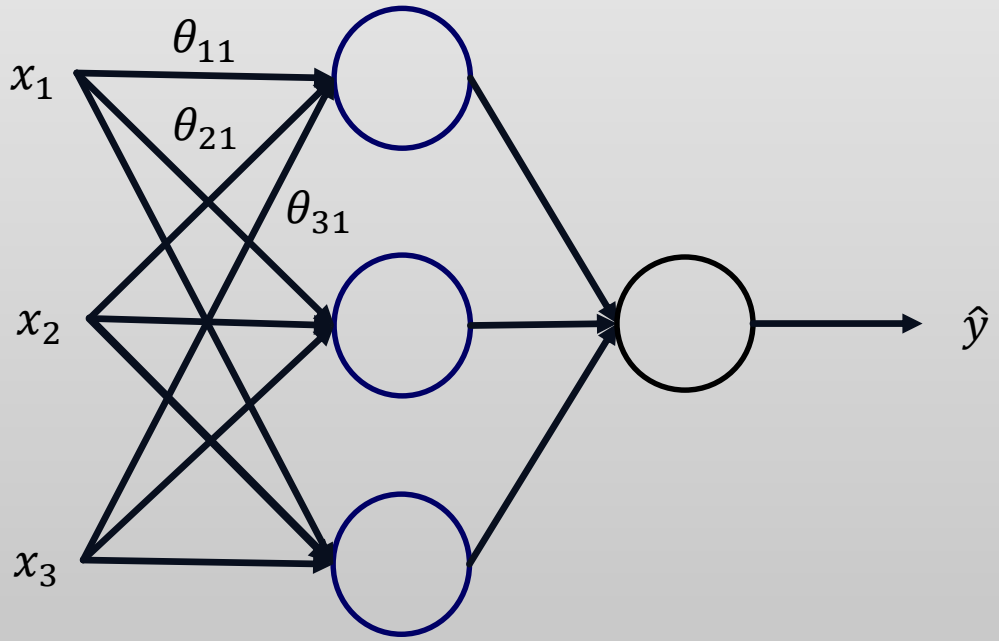


TRAINING A MODEL



Define a loss function and apply batch gradient descent (the below architecture is called ***fully connected***):

$$h_1 = \sigma(\theta_{11} \cdot x_1 + \theta_{21} \cdot x_2 + \theta_{31} \cdot x_3)$$



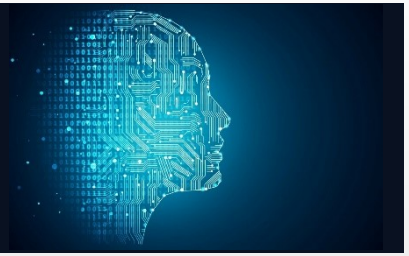
- Initialize weights randomly
- Choose a batch of M labeled examples ($M \ll N$) , $\mathrm{D} = \{X, y\}_{i=1}^{N}$
- Calculate the loss function

$$L(\theta) = \frac{1}{M}\sum_{i=0}^{M} C(\hat{y}(x_i, \theta), y_i(x_i)) = \frac{1}{2 \cdot M}\sum_{i=0}^{M} (\hat{y}(x_i, \theta) - y_i(x_i))^2$$

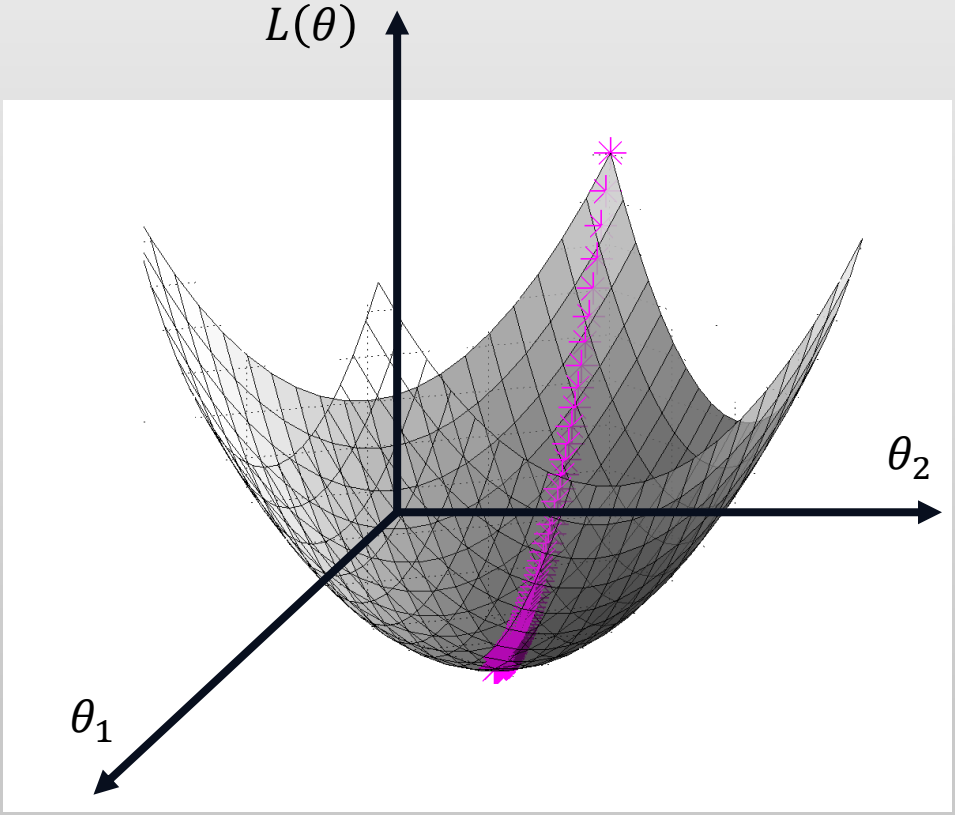
- Update the model parameters by derivation

$$\frac{\partial L}{\partial \theta} = \frac{1}{M}\sum_{i=0}^{M} C(\hat{y}(x_i, \theta), y_i(x_i)) \cdot \frac{\partial \hat{y}}{\partial \theta}$$

TRAINING A MODEL



Define a loss function and apply batch gradient descent:



- Backpropagation is derived from the chain rule in calculus:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

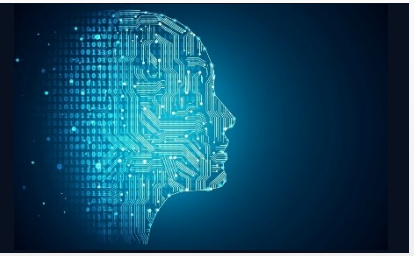
$$(f \circ g)'(x) = f'(g(x)) \cdot g'(x)$$

- In order to minimize the loss function, we will follow the following rule:

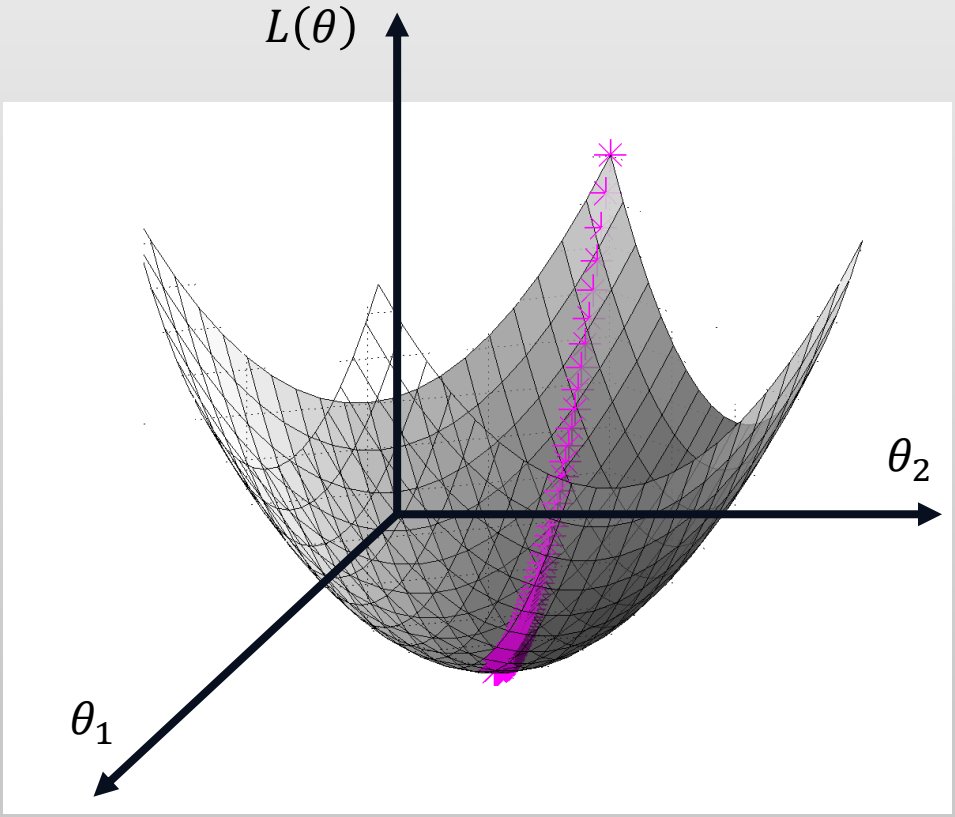
$$\theta^{t+1} := \theta^t - \alpha \cdot \frac{\partial L}{\partial \theta}$$

→ Neural Nets can be trained via backprop only for feedforward model architectures!

TRAINING A MODEL

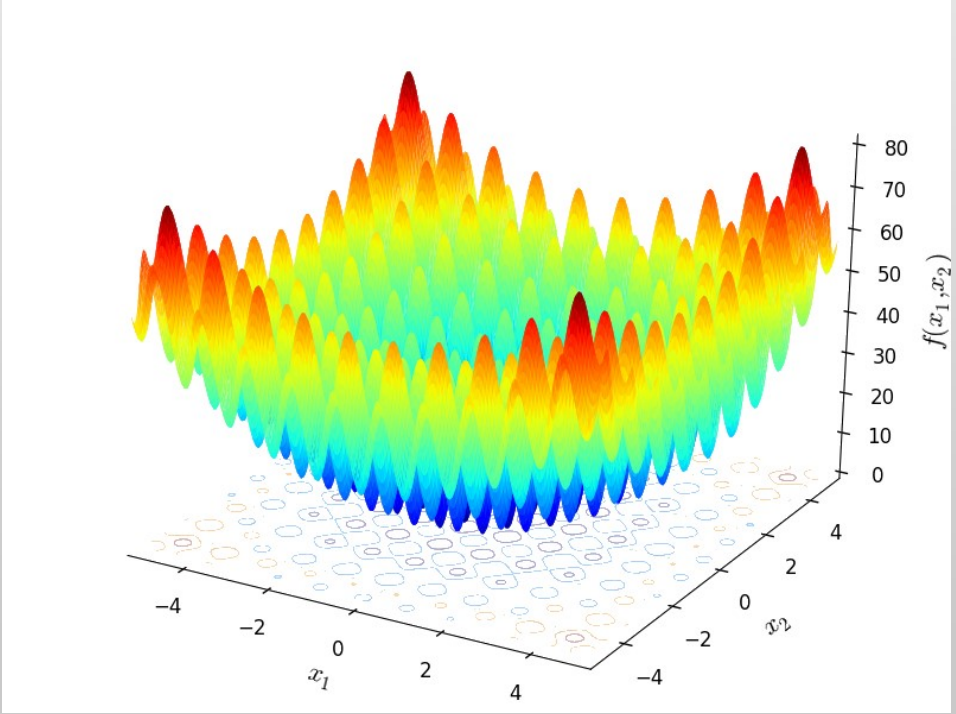


Define a loss function and apply batch gradient descent:

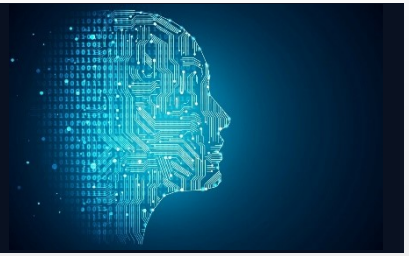


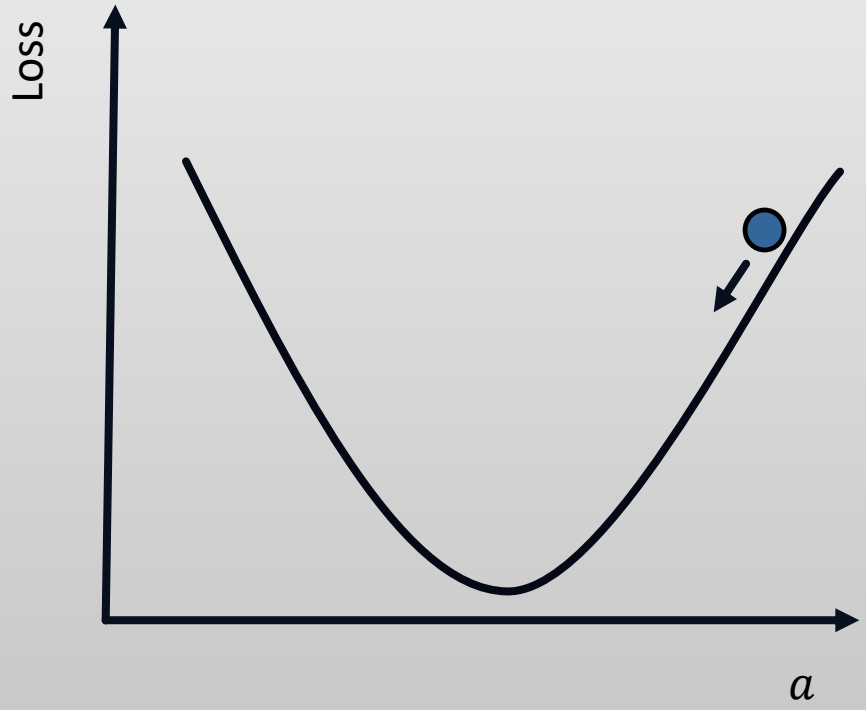
- Deep Neural Net:

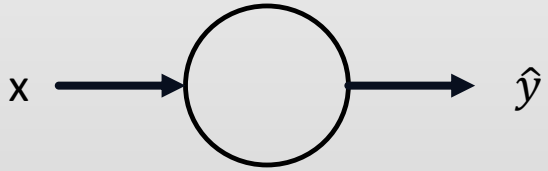
$$(f \circ g \circ h \ldots.)'(x) = f' \cdot g' \cdot \mathrm{h}' \cdot \ldots.$$



TRAINING A MODEL





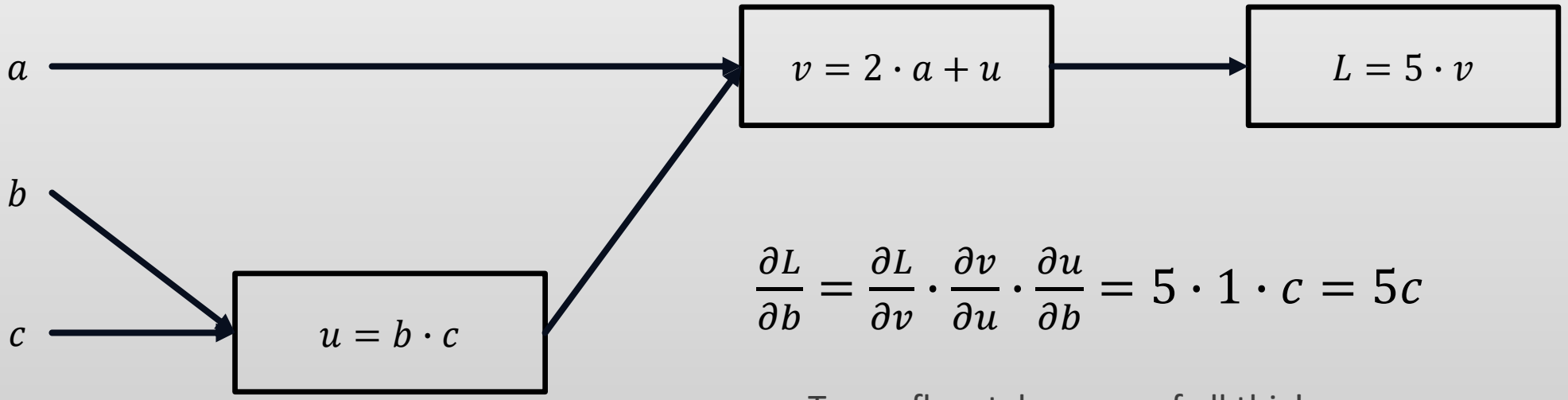


$$\hat{y} = a \cdot x + b$$

$$a^{t+1} := a^t - \alpha \cdot \frac{\partial L}{\partial a}$$

$$b^{t+1} := b^t - \alpha \cdot \frac{\partial L}{\partial b}$$

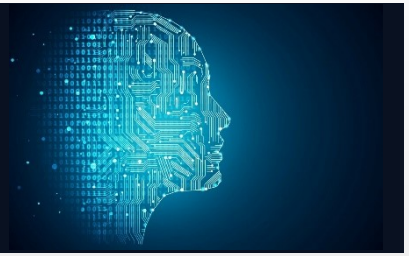
TRAINING A MODEL – EXAMPLE



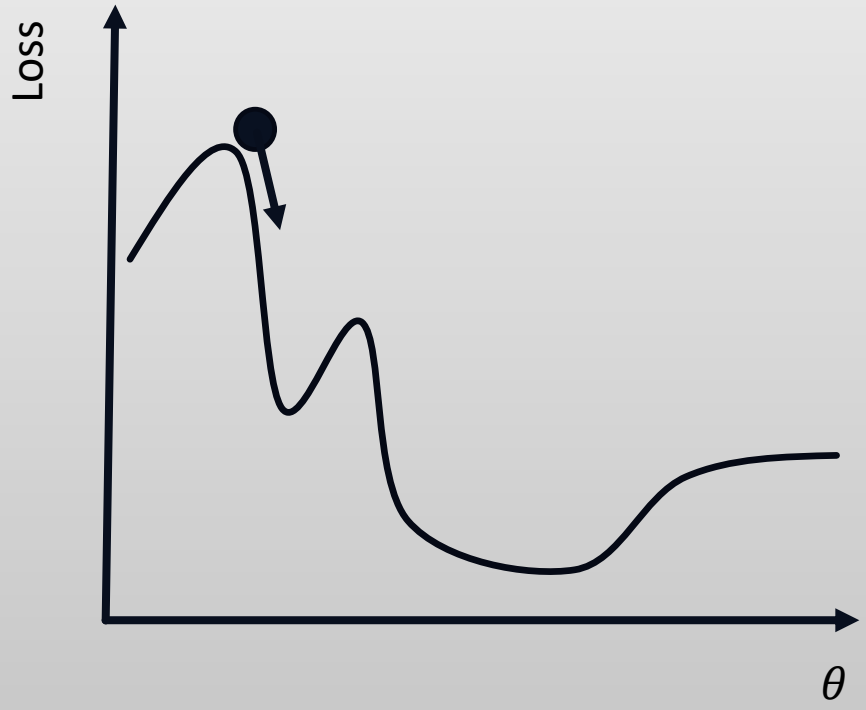
$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial v} \cdot \frac{\partial v}{\partial u} \cdot \frac{\partial u}{\partial b} = 5 \cdot 1 \cdot c = 5c$$

- Tensorflow takes care of all this!
- The user defines the computation graph, loss etc. Tensorflow computes the optimal graph parameters.

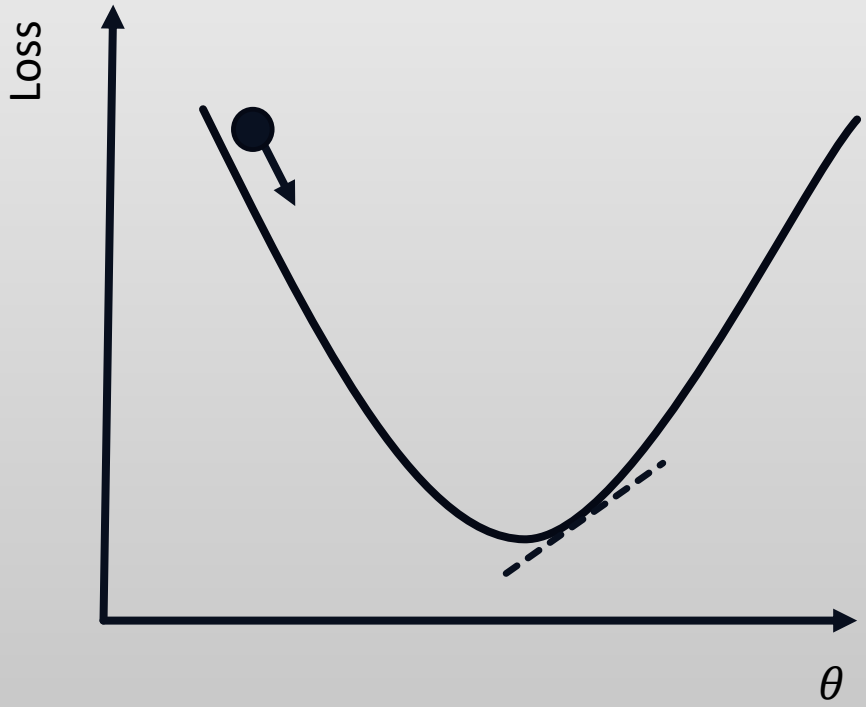
TRAINING A MODEL – OPTIMIZATION



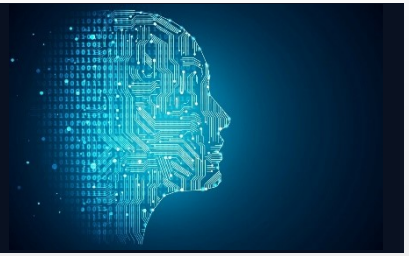
"Real Life":



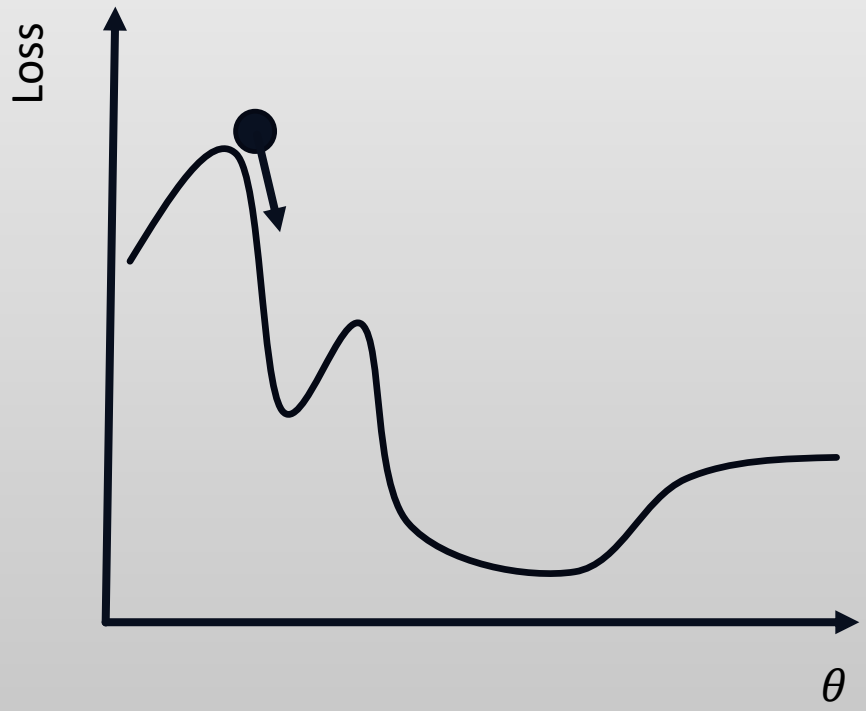
Convex Optimization:



TRAINING A MODEL – OPTIMIZATION



Momentum:



- Stochastic Gradient Descent:

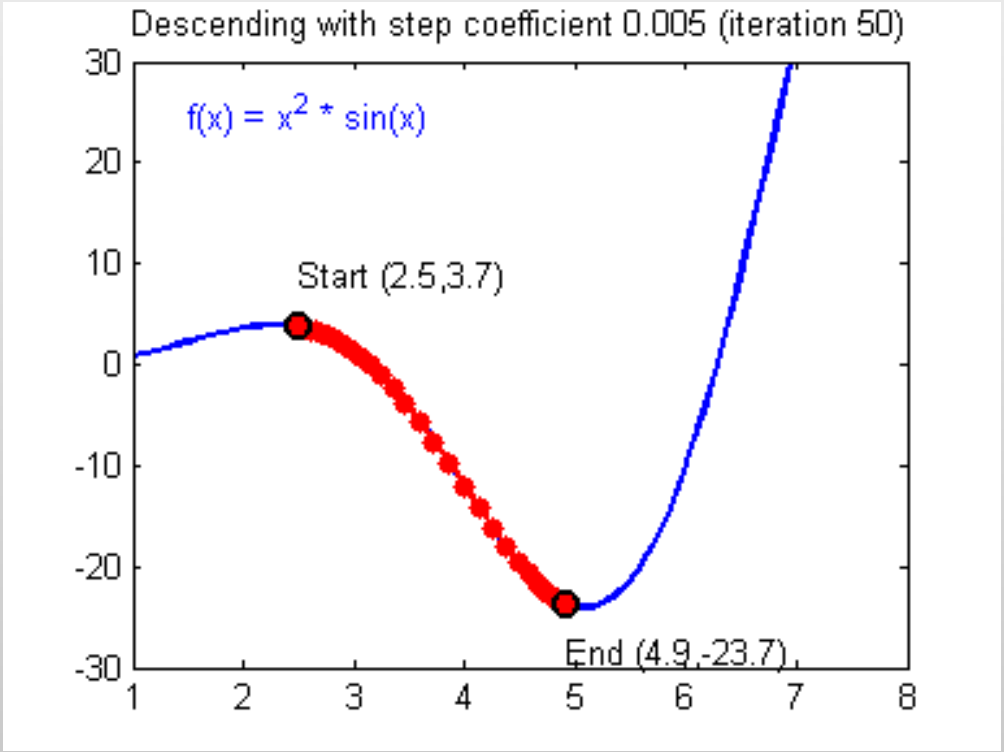
$$\theta^{t+1} \coloneqq \theta^t - \alpha \cdot \nabla_\theta \mathrm{L}$$

- Momentum:

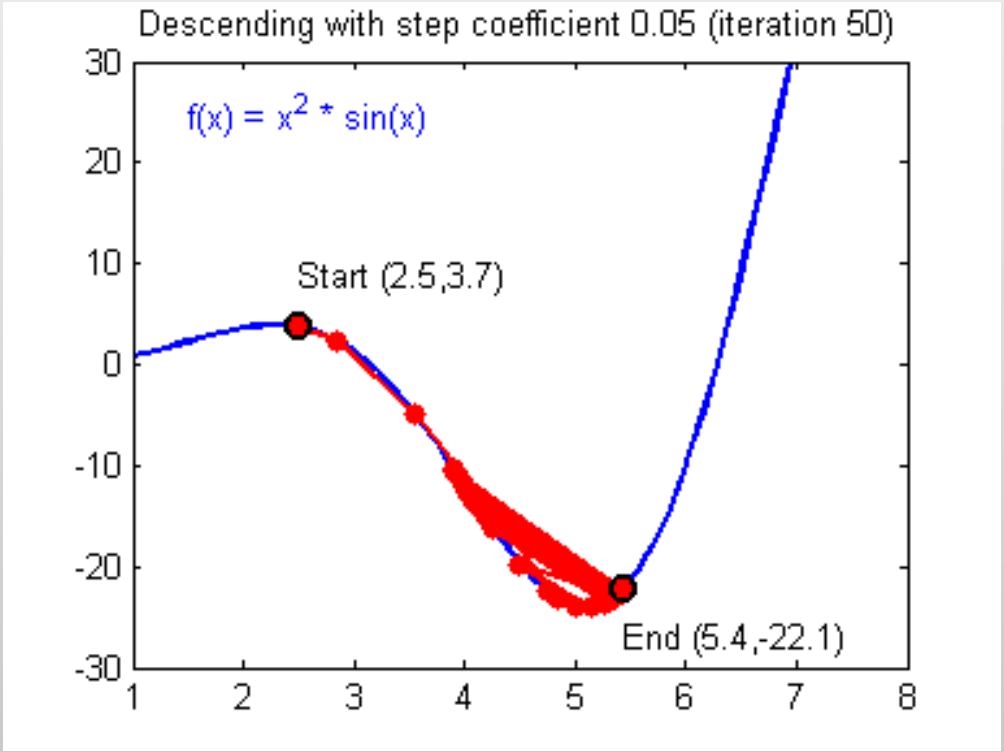
$$v^{t+1} \coloneqq \gamma \cdot v^t + \alpha \cdot \nabla_\theta \mathrm{L}$$
$$\theta^{t+1} \coloneqq \theta^t - v^{t+1}$$

- Many different optimization algorithms, a very active research field.

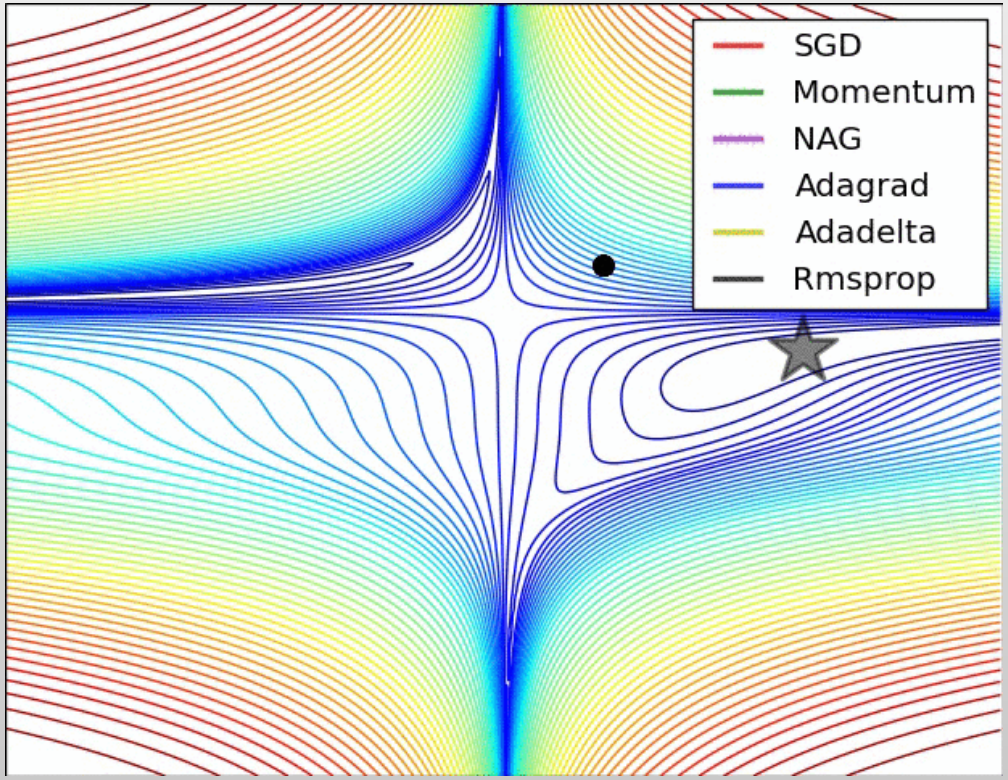
TRAINING A MODEL – OPTIMIZATION

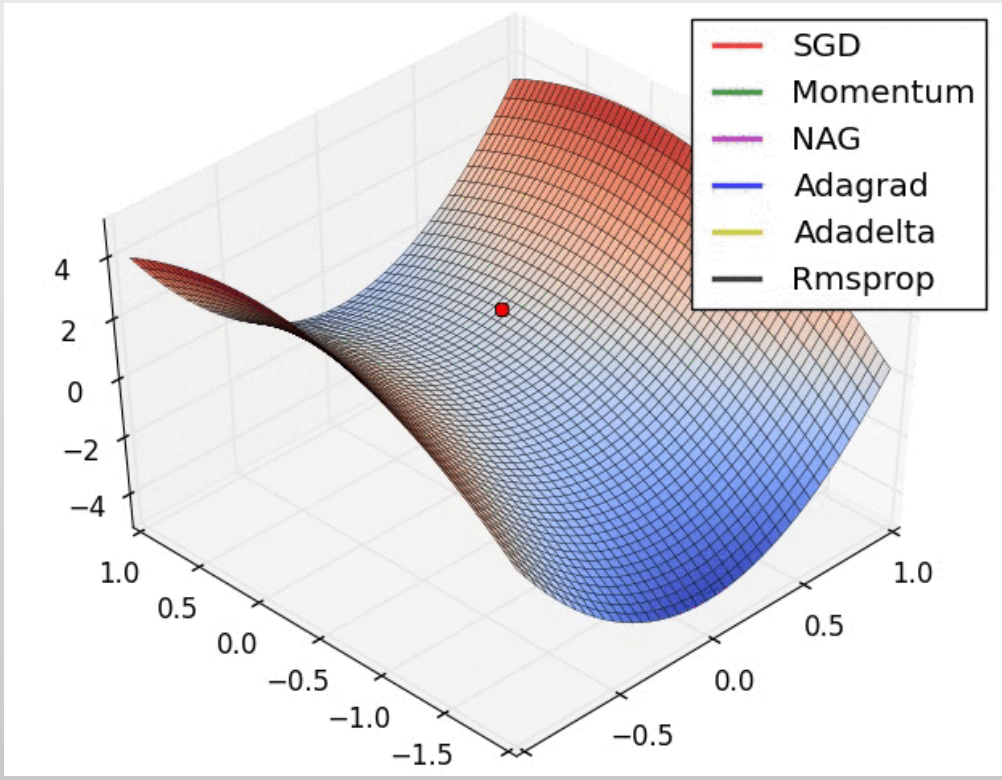


TRAINING A MODEL – OPTIMIZATION

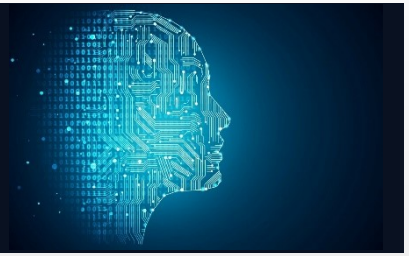


TRAINING A MODEL – OPTIMIZATION

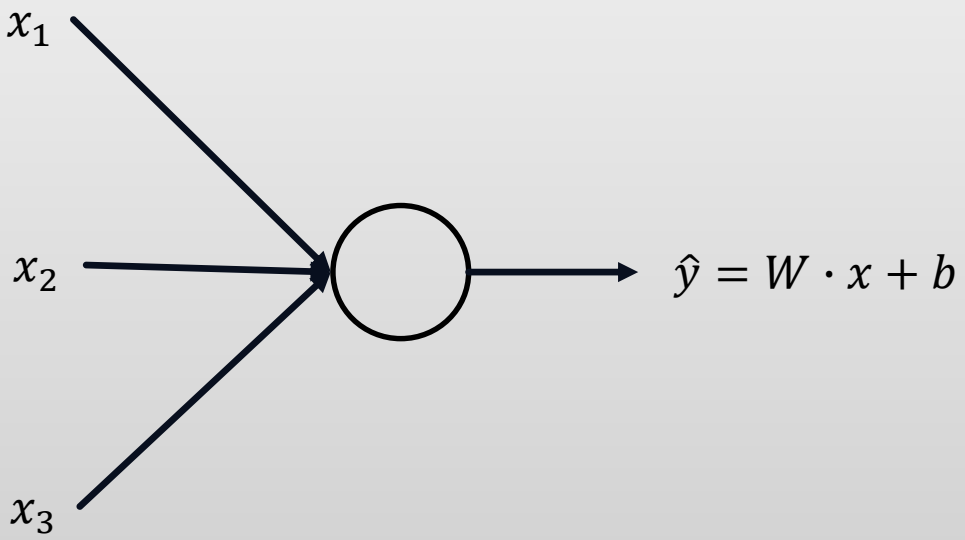


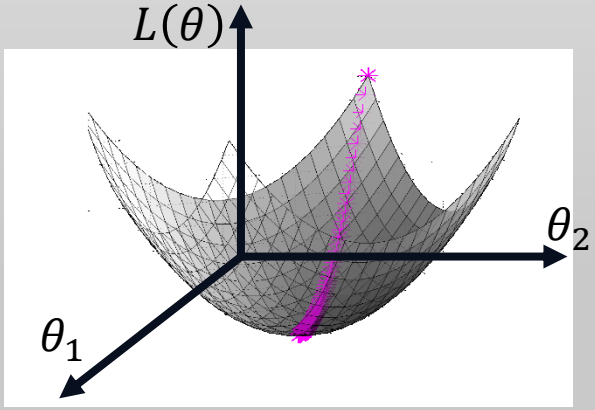


WHEN IS A NN CONVEX?

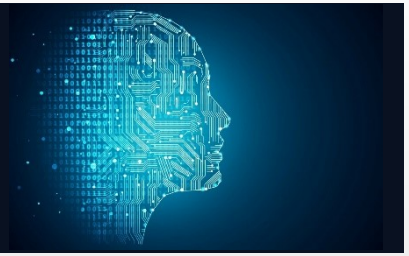


- Is a linear regressor a convex problem?
  - Yes, but it may not be computable!
  - Closed form algebraic solution
  - $S_{opt} = \left(A^T A\right)^{-1} A^T$
  - This may not be computable for large problems

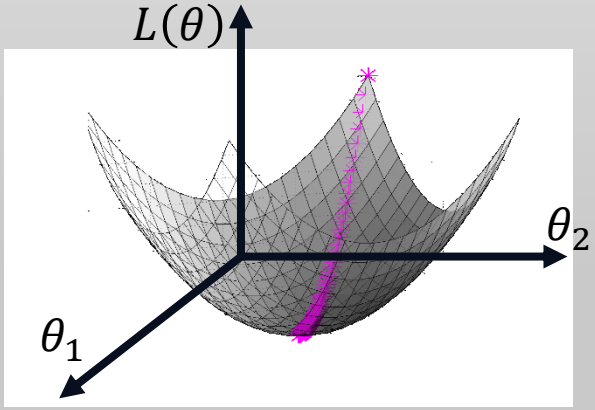


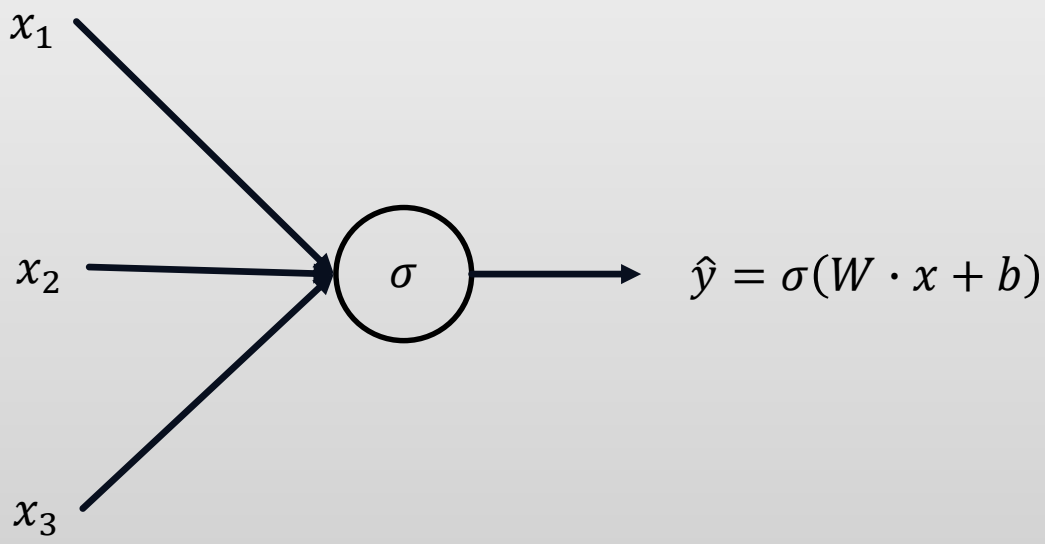


WHEN IS A NN CONVEX?

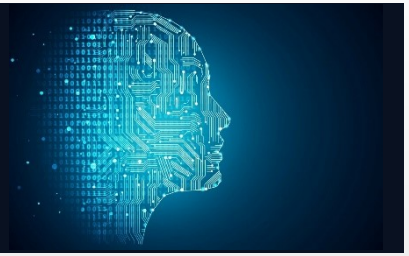


- Is a logistic regressor a convex problem?
  - Yes, but no **known method** to find optimal solution
  - Can be solved numerically
  - Numerical solutions may or may not be stable!
  - complete separation vs quasi-complete separation (link)

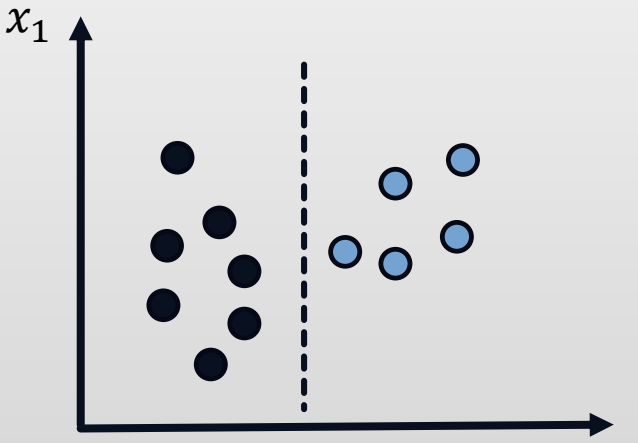


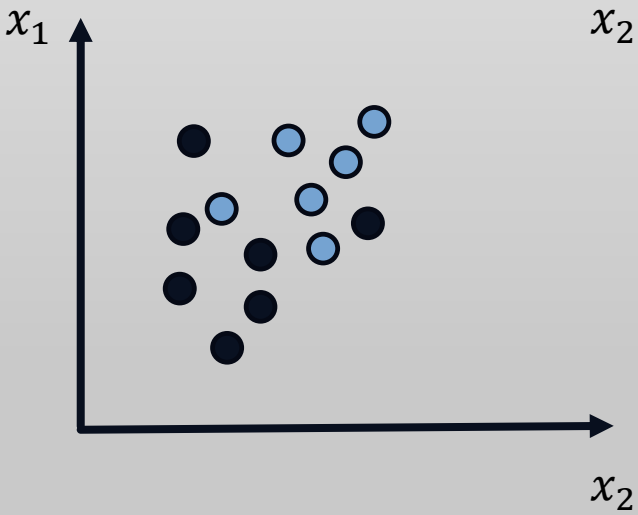


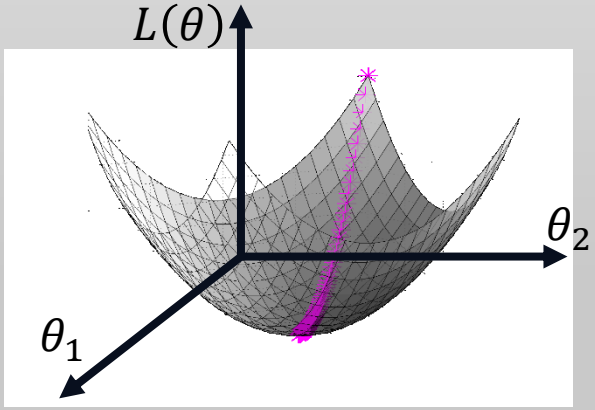
WHEN IS A NN CONVEX?



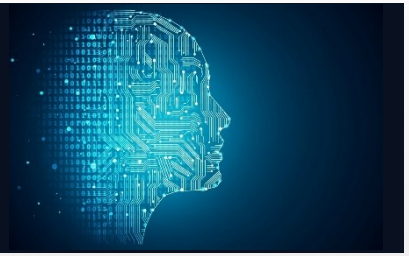
- complete separation vs quasi-complete separation:



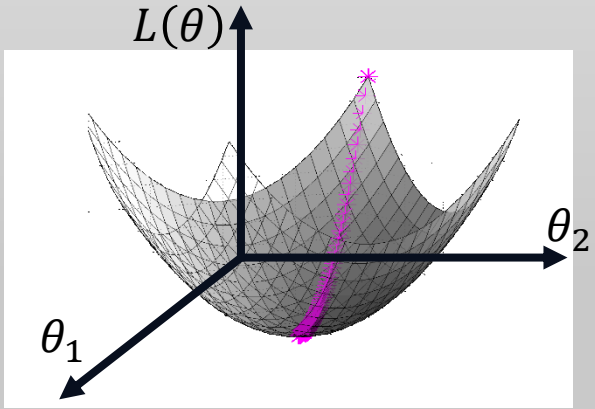


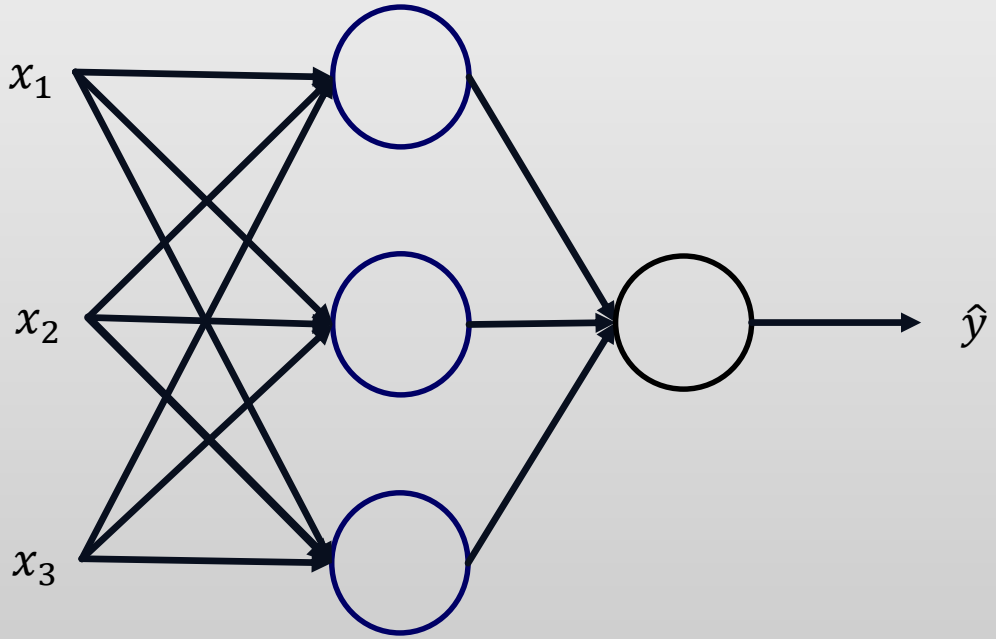


WHEN IS A NN CONVEX?

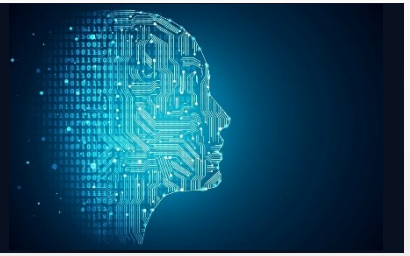


- Can an MLP (Multi Layer Perceptron) have a convex solution?
  - What happens if we switch between 2 blue nodes?

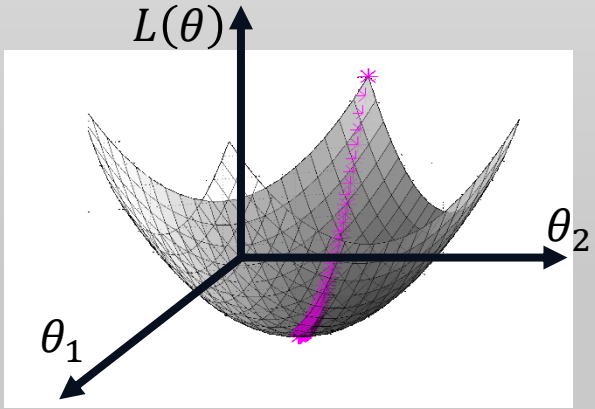


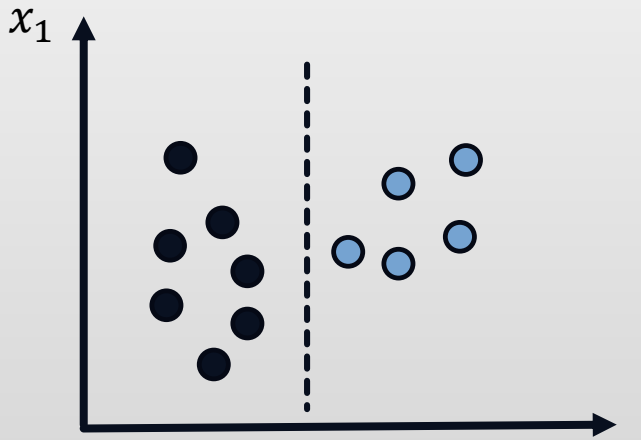


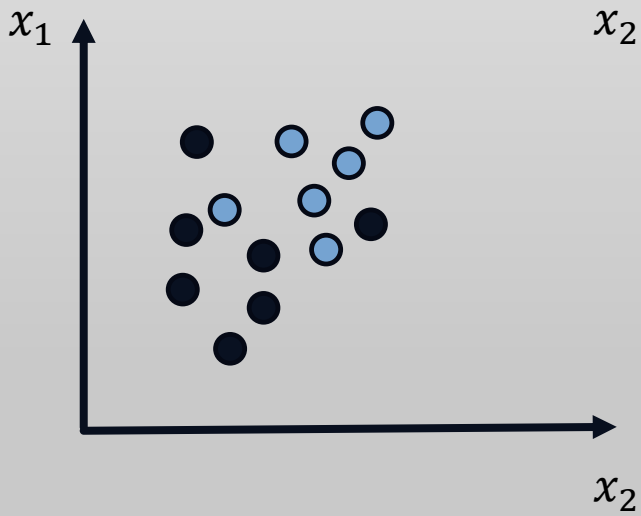
WHEN IS A NN CONVEX?



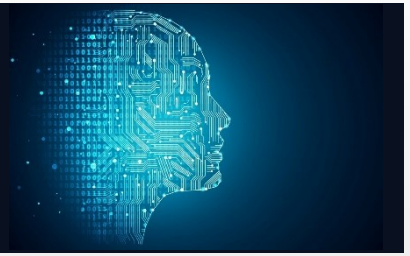
- Conclusions:
  - Almost all “real world” problems require an approximation, we do not have an optimal (and computable!) solution
  - Todays methods almost always rely on gradient descent based methods



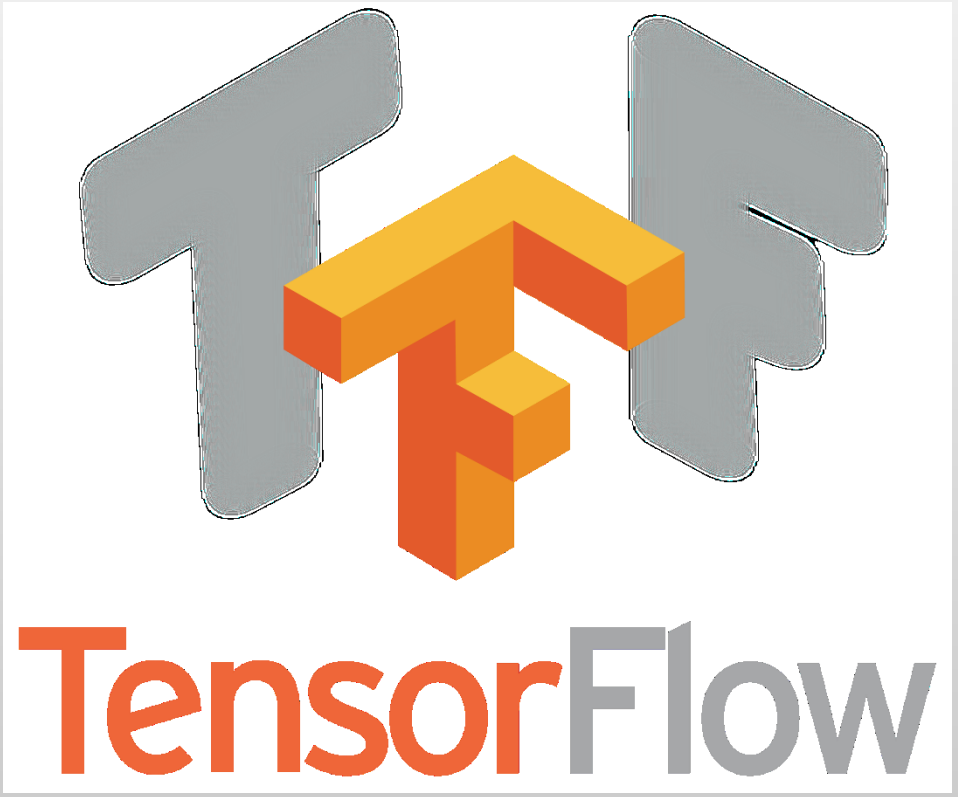




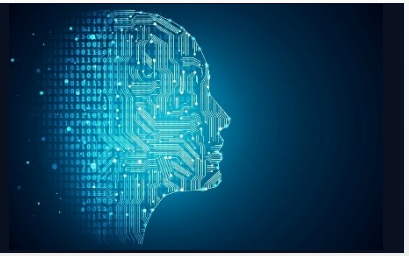
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Linear Regression example in Python



WHY TENSORFLOW?

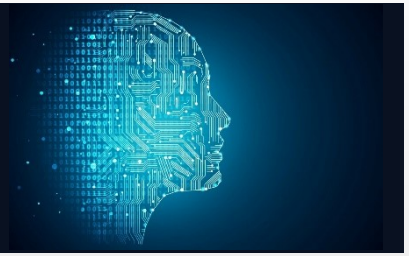


- **Why use Tensorflow?** We have so many great model in sklearn, xgboost etc.
- Traditional ML introduces multiple elements as one complete algorithm (with many parameters)

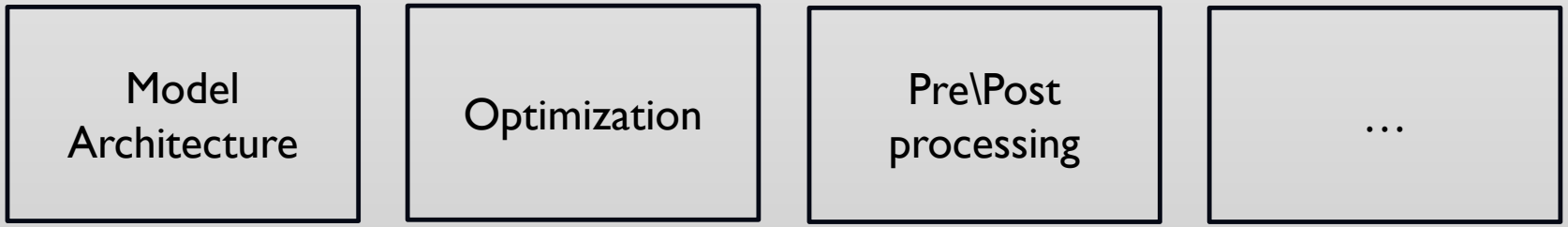
Model Architecture	Optimization	Pre\Post processing	…

- Example: *Random Forest*
  - Decision trees with “split” parameters
  - Gini-index + brute force optimization
  - Post-pruning

WHY TENSORFLOW?



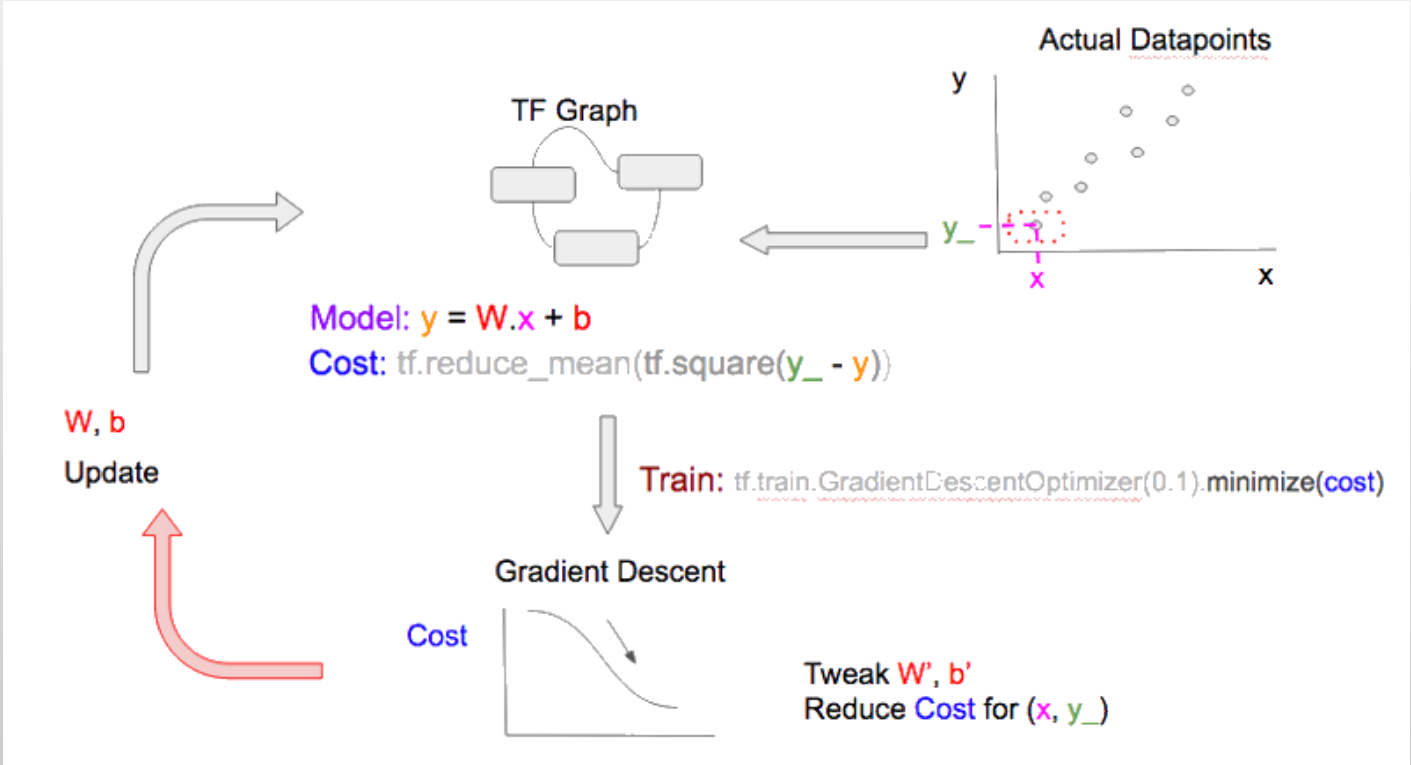
- **Why use Tensorflow?** We have so many great model in sklearn, xgboost etc.
- Traditional ML introduces multiple elements as one complete algorithm (with many parameters)



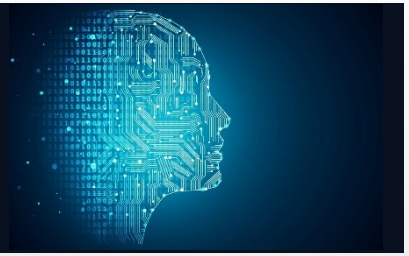
- Tensorflow allows us to pick and choose state-of-the-art elements and combine into a custom algorithm
- This enables **customized models** for specific problems. More specifically, it allows us to introduce **domain knowledge** into the problem.

TENSORFLOW IN PYTHON

- Key concepts in deep learning simulations
  - **Reading files**
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation



TENSORFLOW IN PYTHON

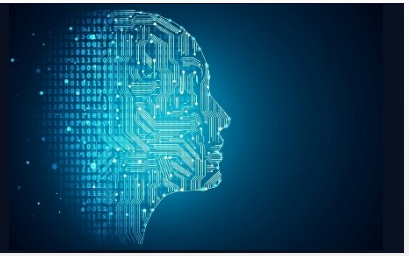


- Key concepts in deep learning simulations
  - **Reading files**
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

```
Import numpy as np
# Data is in matrix format
data=np.loadtxt(path, delimiter…)
```

Works for data in matrix form …

TENSORFLOW IN PYTHON

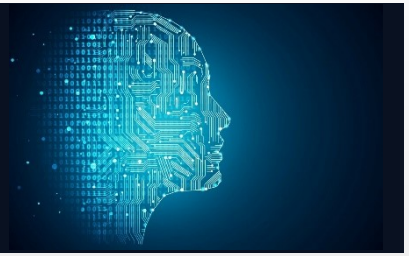


- Key concepts in deep learning simulations
  - Reading files
  - **Variables: Model Variables**
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

```
Import tensorflow as tf

# Data is in matrix format

variable=tf.get_variable('my_var')
```

- Variables usually describe nodes in a graph
- Placeholders are the interface between Tensorflow variables and Python scripts
- Tensorflow is a c++ library with and interface to various languages. The separation allows ease-of-use when debugging and also speed\efficiency when training or running inference

TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - **Variables: Model Variables**
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

Import tensorflow as tf

Data is in matrix format

variable=tf.get_variable('my_var')

- Supports some useful features:
  - Initializer
  - Regularizer

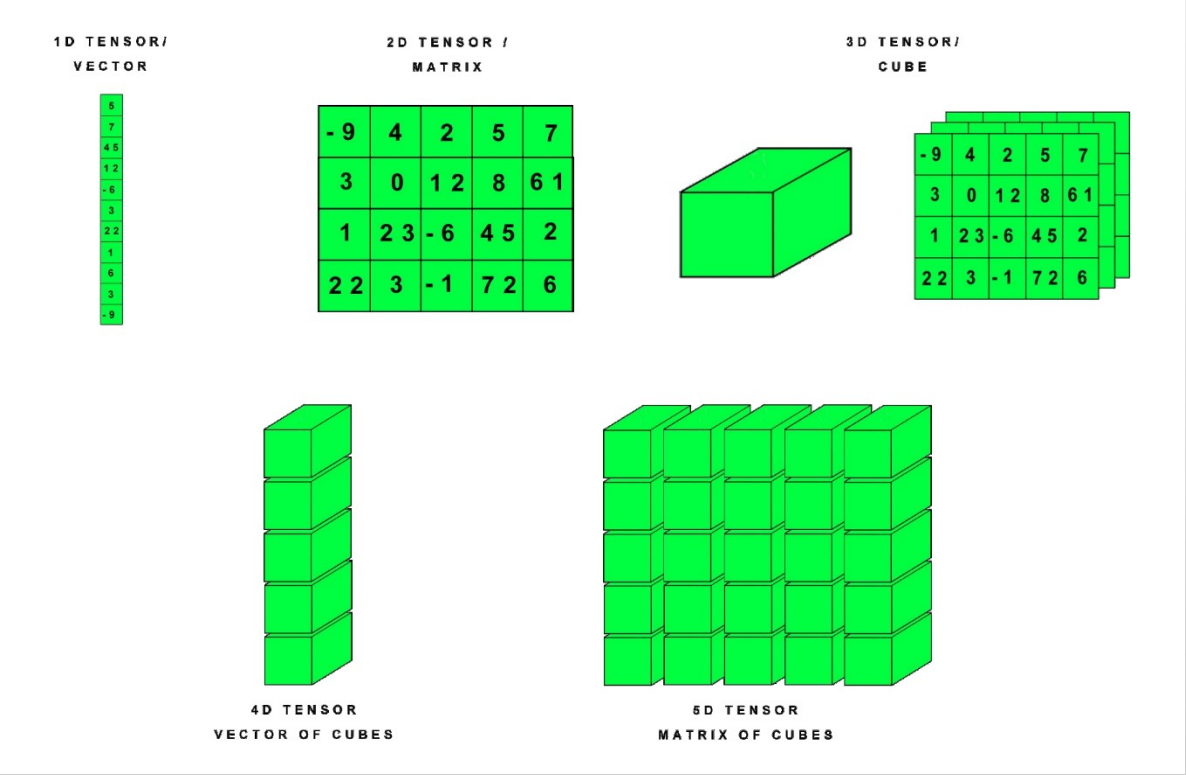
TENSORFLOW IN PYTHON

- Key concepts in deep learning simulations
  - Reading files
  - **Variables: Placeholders**
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

- Variables and Placeholders:

```
x_in = tf.placeholder(tf.float, shape=(2,N))
```

TENSORFLOW IN PYTHON



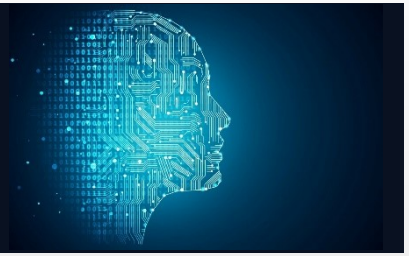
- Tensors are generalization of vectors\matrices
- Tensors are the datatype used in Tensorflow
- A color image, for example, is a 3d tensor

"How do you imagine a 5d surface? Look at a 3d surface and say 5…"

-Geoff Hinton

- Idea originates from the Riemann curvature tensor

TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - **Variables: feed dictionary**
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

```
sess.run([my_variable0, my_variable1],
        feed_dict={my_placeholder0: data0,
                   my_placeholder1: data1})
```

- Feed_dict
  - Placeholders are keys
  - Data are values
  - Session is the tensorflow mechanism which evaluates the tensor ops and updates the model parameters
  - This methodology is designed for speed (such as cloud)

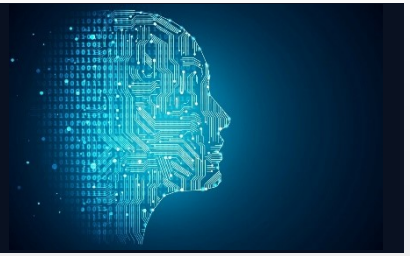
EXERCISE 1B – LINEAR REGRESSION

- basic graph concepts: session

```
# Import `tensorflow`
import tensorflow as tf
# Initialize two constants
x1 = tf.constant([1,2,3,4])
x2 = tf.constant([5,6,7,8])
# Multiply
result = tf.multiply(x1, x2)
# Print the result
print(result)
```

→Tensor("Mul:0", shape=(4,), dtype=int32)

EXERCISE 1B – LINEAR REGRESSION



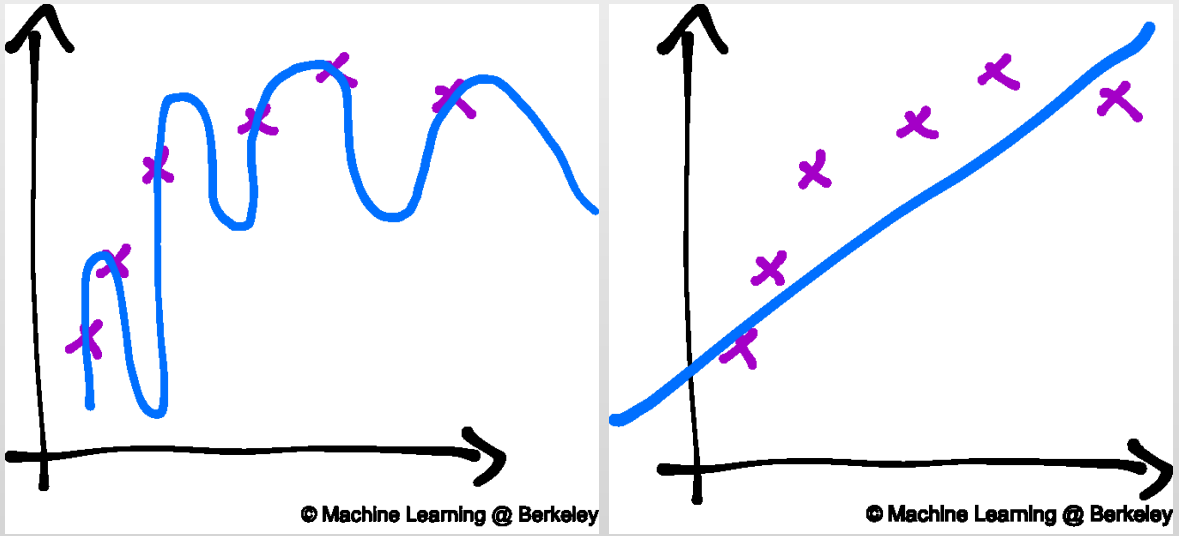
- basic graph concepts: session

```
# Import `tensorflow`
import tensorflow as tf
# Initialize two constants
x1 = tf.constant([1,2,3,4])
x2 = tf.constant([5,6,7,8])
# Multiply
result = tf.multiply(x1, x2)
# Intialize the Session
sess = tf.Session()
# Print the result
print(sess.run(result))
# Close the session
sess.close()
```

→[5 12 21 32]

TENSORFLOW IN PYTHON

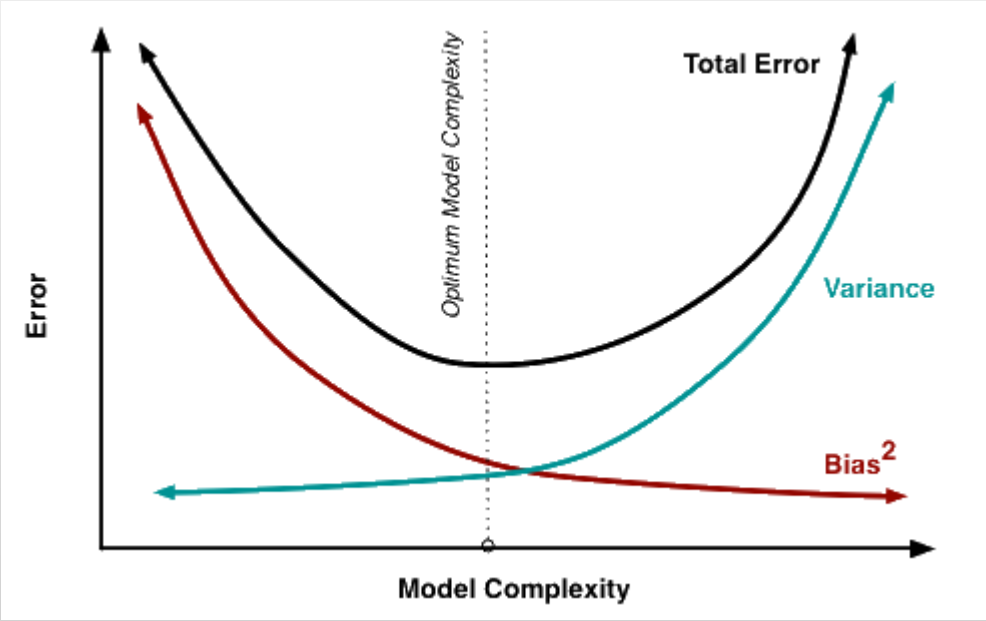
- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - **Designing a model: Bias Variance Tradeoff**
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation



$$E\left[(y-\hat{y})^2\right] = \cdots = \sigma_0^2 + Var(\hat{y}) + Bias^2(\hat{y})$$

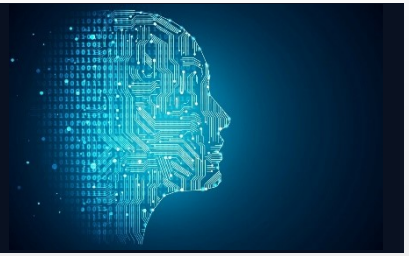
TENSORFLOW IN PYTHON

- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - **Designing a model: Bias Variance Tradeoff**
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation



$$E\left[(y-\hat{y})^2\right] = \cdots = \sigma^2 + Var(\hat{y}) + Bias^2(\hat{y})$$

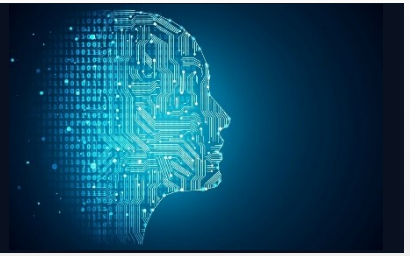
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - **Designing a model: Sparsity**
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

- Sparsity
  - Strongly related to model complexity
  - Model the phenomenon with as few parameters as possible
  - Easier for gradient descent (and in general) optimization
  - Less chance of overfitting
  - Easier to explain (interpretability)

TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model: Bias Variance Tradeoff
  - **Batch optimization**
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - Simulation Evaluation

- Initialize weights randomly
- Choose a batch of M labeled examples ($M \ll N$) , D $= \{X, y\}_{i=1}^{N}$
- Calculate the loss function

$$L(\theta) = \frac{1}{2 \cdot M} \sum_{i=0}^{M} (\hat{y}(x_i, \theta) - y_i(x_i))^2$$

- Update the model parameters by derivation

$$\frac{\partial L}{\partial \theta} = \frac{1}{M} \sum_{i=0}^{M} (\hat{y}(x_i, \theta), y_i(x_i)) \cdot \frac{\partial y}{\partial \theta}$$

TENSORFLOW IN PYTHON

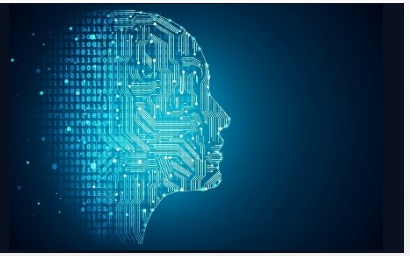
- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - **Epochs and Randomization**
  - Dimensions
  - Visualizations
  - Simulation Evaluation

$$L(\theta) = \frac{1}{2 \cdot M} \sum_{i=0}^{M} (\hat{y}(x_i, \theta) - y_i(x_i))^2$$

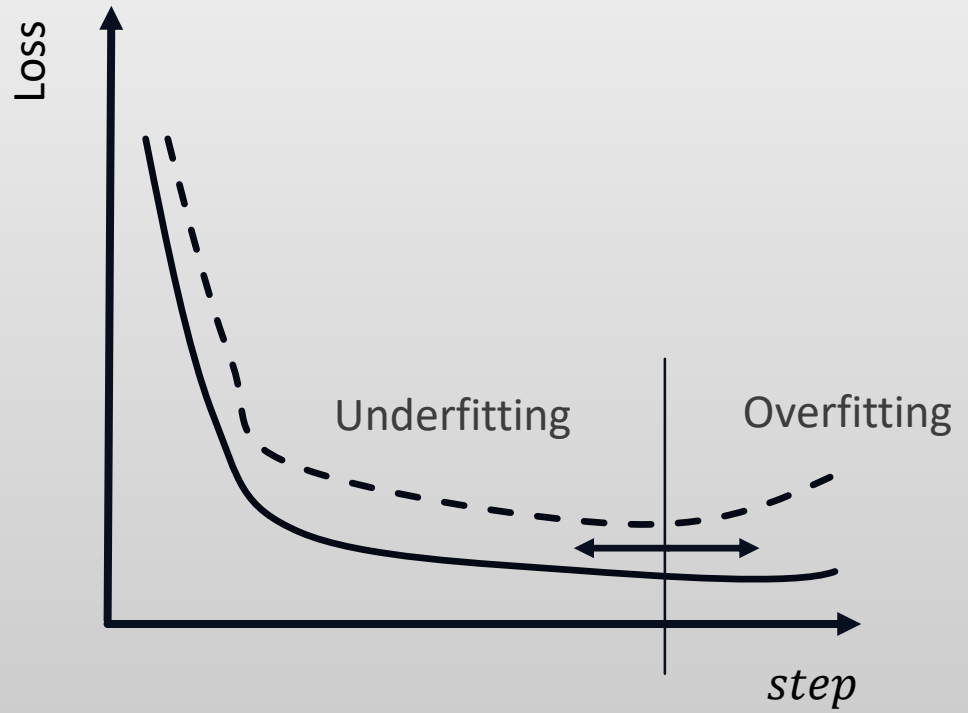
Pseudocode:

1. Randomize dataset
2. Loop num_epochs
3. Loop over num_observations
4. Choose M observations
5. Optimize $L(\theta)$
6. Observe Train and Test Loss

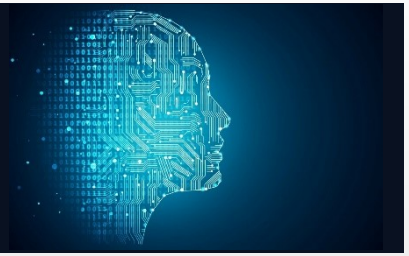
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - **Epochs and Randomization**
  - Dimensions
  - Visualizations
  - Simulation Evaluation



TENSORFLOW IN PYTHON

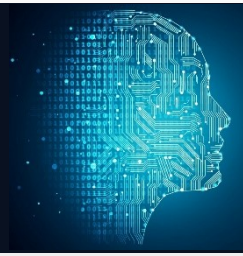


- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model: Bias Variance Tradeoff
  - Batch optimization
  - Epochs and Randomization
  - **Dimensions**
  - Visualizations
  - Simulation Evaluation

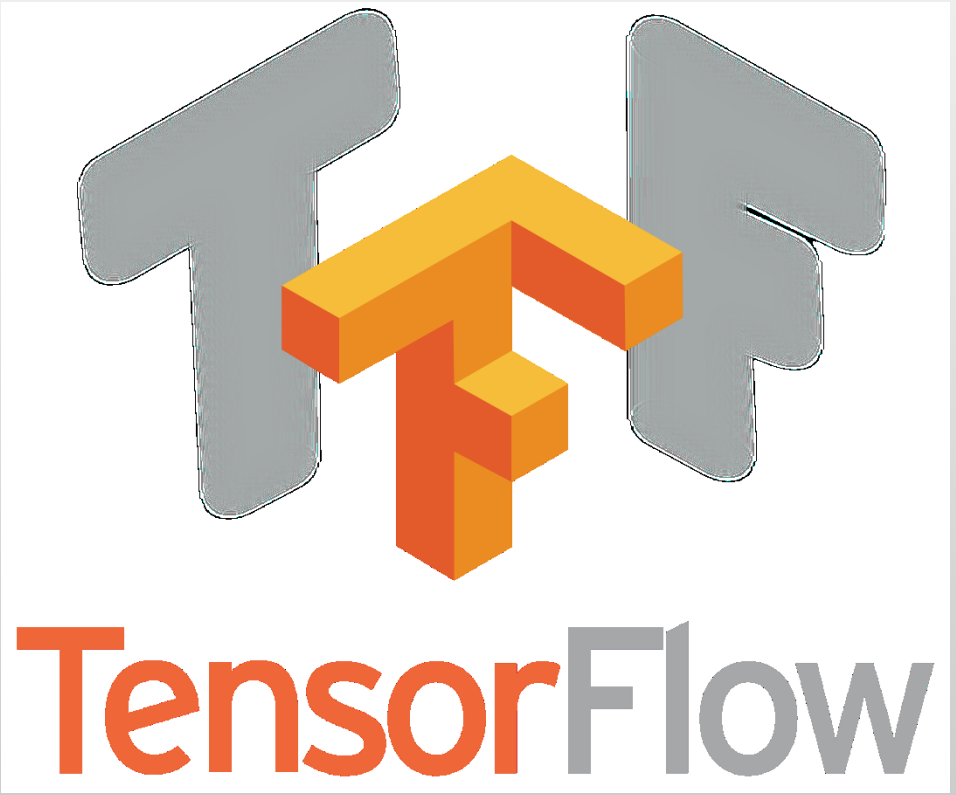
- Note the usage of redundant dimensions
- Use tf.expand_dims(tensor, axis=…) or

tf.squeeze(tensor, axis=…) when working with redundant dimensions

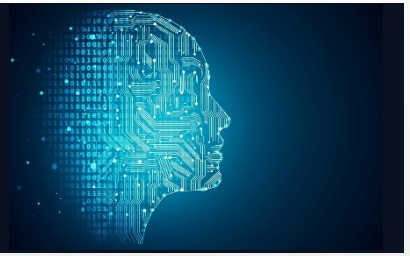
TENSORFLOW IN PYTHON



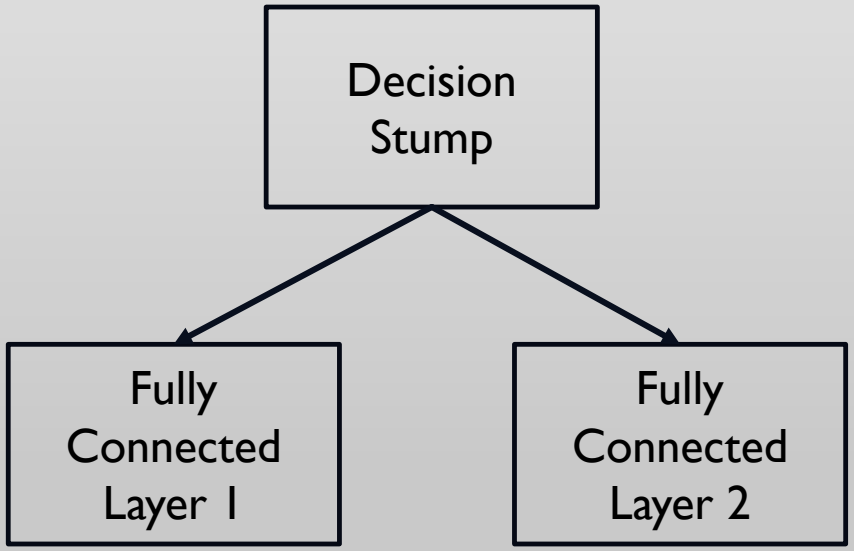
- Key concepts in deep learning simulations
  - Linear Regression competition
  - Information about customers consists of 86 variables and includes product usage data and socio-demographic data. The data was supplied by the Dutch data mining company Sentient Machine Research and is based on a real world business problem. The training set contains over 5000 descriptions of customers, including the information of whether or not they have a caravan insurance policy. A test set contains 4000 customers of whom only the organizers know if they have a caravan insurance policy.

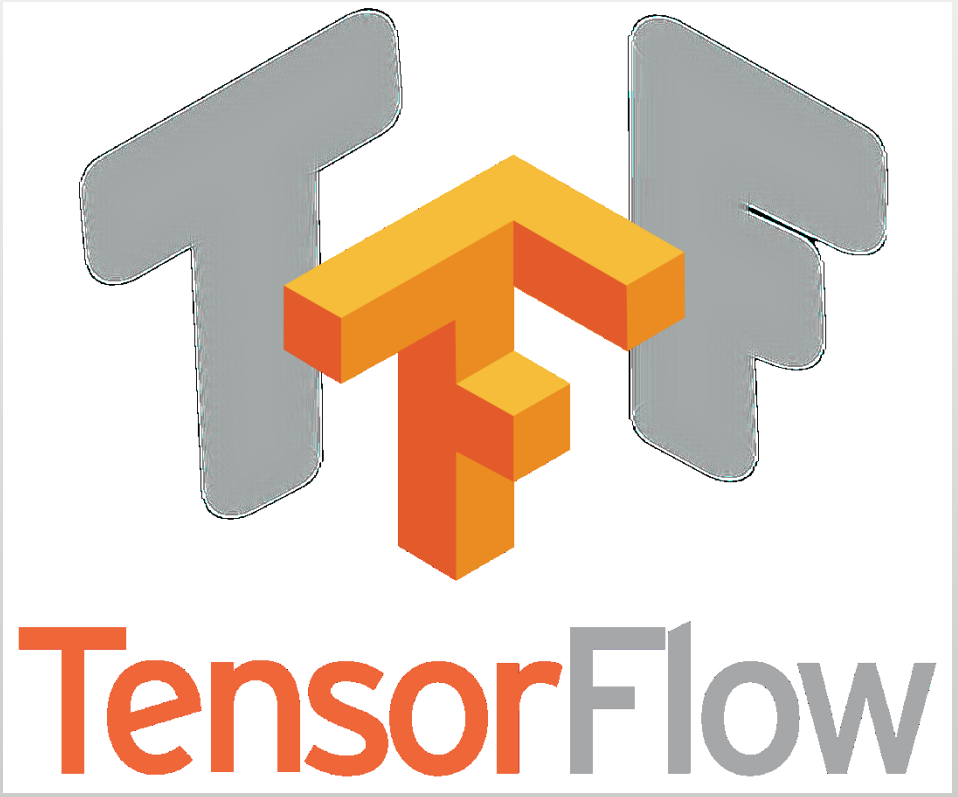


TENSORFLOW IN PYTHON

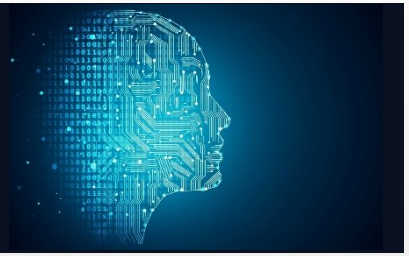


- Key concepts in deep learning simulations
  - Linear Regression competition
  - Create a custom model:
  - Hint: create two fc layers and use tf.where()





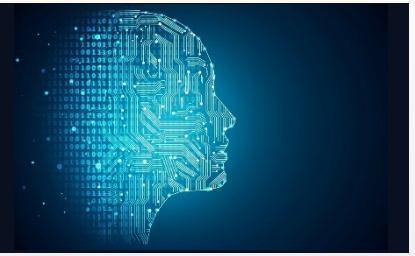
TENSORFLOW IN PYTHON



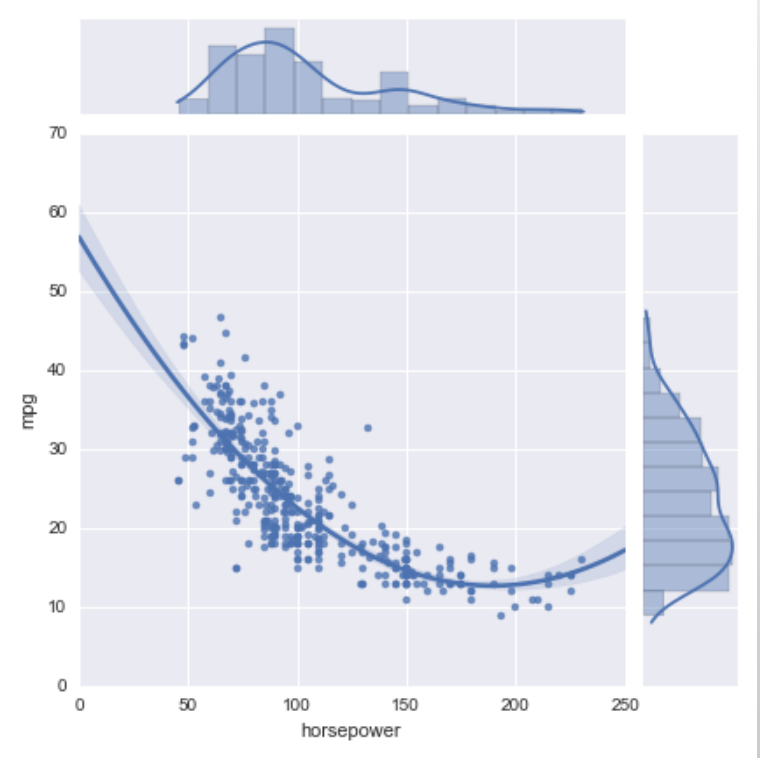
- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Custom Plots**
  - Simulation Evaluation

- Custom plots include any visualization on the output of any of the nodes in a model (or anything else)
- Recommended plots can include:
  - Plot raw data observations which the model fails on inference
  - Plot feature space clusters
  - Plot activations as a function of data
  - etc.
- Matplotlib and seaborn are great packages for many different plots.
  - Matplotlib is the standard plotting library
  - Seaborn is a beautiful extension

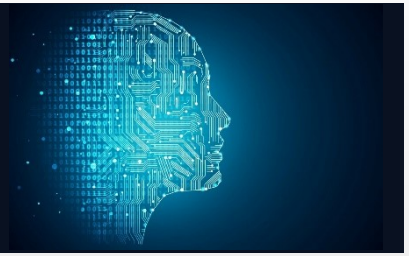
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Custom Plots**
  - Simulation Evaluation



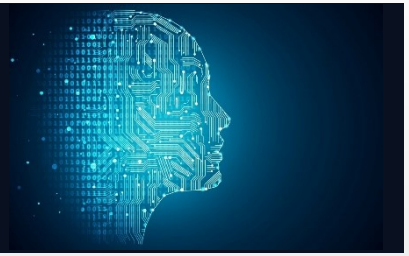
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Summaries and Tensorboard**
  - Simulation Evaluation

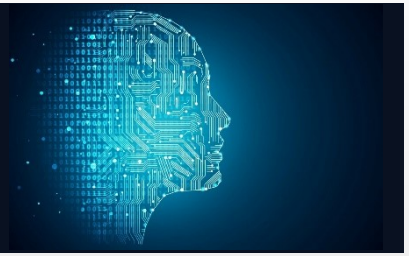
- Summaries are objects in Tensorflow which can “save” node values during model training
- Summaries also present the “saved” values into presentable format
- Currently supported:
  - tf.summary.scalar
  - tf.summary.histogram
  - tf.summary.image
  - tf.summary.audio
- View summaries (and more) with Tensorboard
- TensorBoard is a visualization tool
  - Recognizes the importance of visualization!

TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Summaries and Tensorboard**
  - Simulation Evaluation

TENSORFLOW IN PYTHON

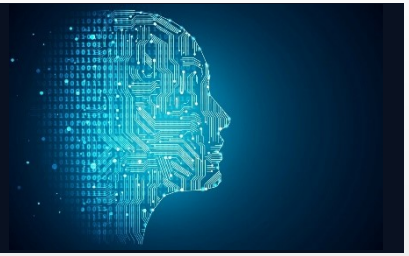


- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Summaries and Tensorboard**
  - Simulation Evaluation

- Tensorboard Python example:

```
# Create a summary to monitor cost tensor
tf.summary.scalar("loss", my_tensor_scalar0)
tf.summary.scalar(“accuracy", my_tensor_scalar1)

# Merge all summaries into a single op
merged_summary_op = tf.summary.merge_all()

# After init:
summary_writer = tf.summary.FileWriter(path, graph=tf.get_default_graph())

# During training…
summary = sess.run([merged_summary_op], feed_dict={x: batch_x, y: batch_y})

 # Write logs
summary_writer.add_summary(summary, epoch * total_batch + i)
```

TENSORFLOW IN PYTHON

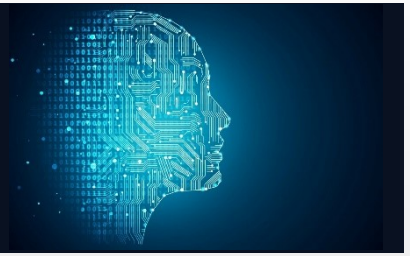


- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - **Visualizations: Summaries and Tensorboard**
  - Simulation Evaluation

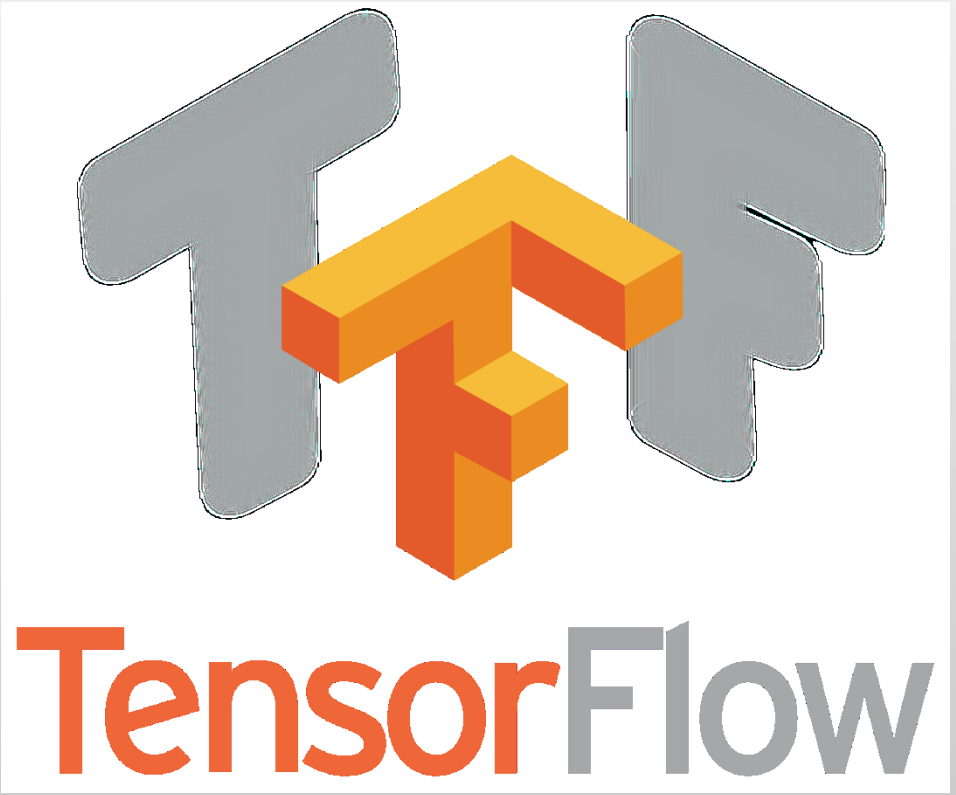
- Tensorboard activation from command line:

```
tensorboard --logdir=path
```

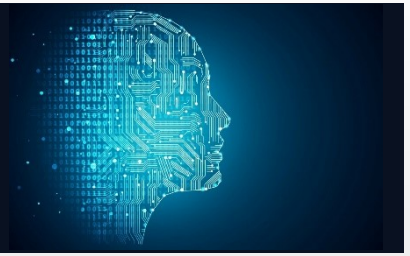
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Linear Regression competition
  - Add Tensorboard summaries
    - Loss
    - Activation histogram



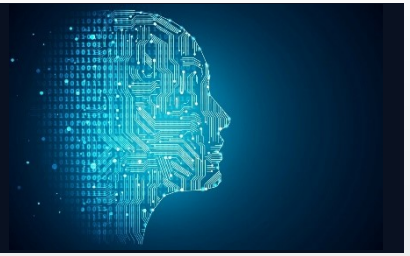
TENSORFLOW IN PYTHON



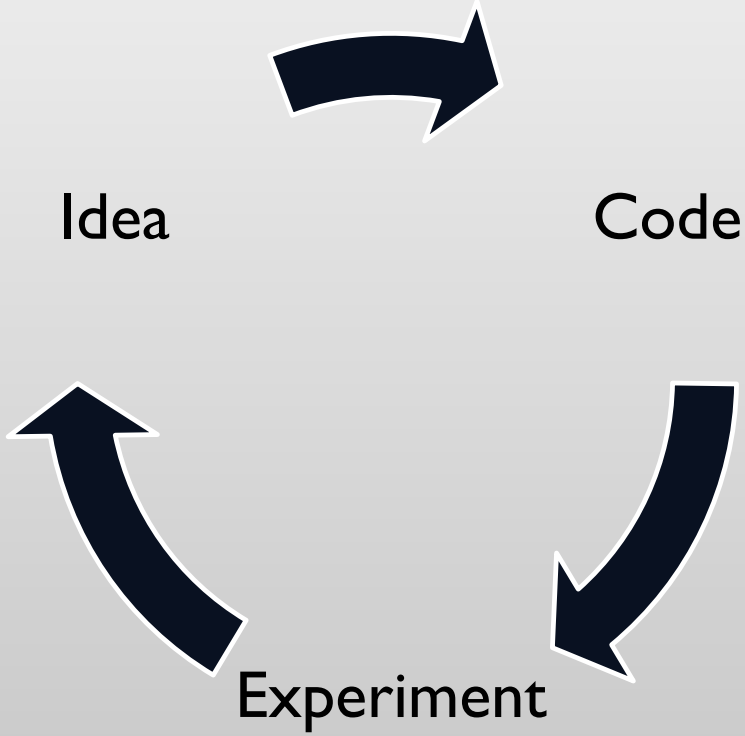
- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations:
  - **Simulation Evaluation**

- A good model is a model which can predict well in practice
- A model which is near the “optimum” with respect to overfitting and underfitting is usually a good model, is the irreducible error is low enough
- A good model needs to generalize beyond the data it is used to train on
- Testing a model for good generalization error is **extremely important** in practice

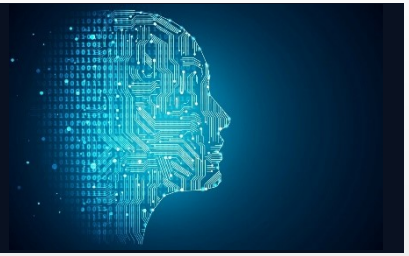
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations:
  - **Simulation Evaluation**

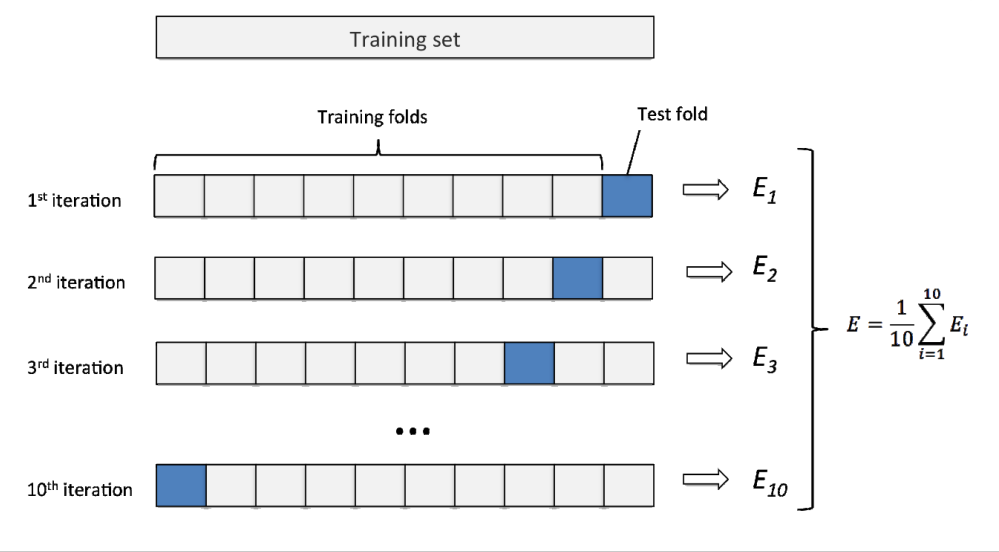


TENSORFLOW IN PYTHON

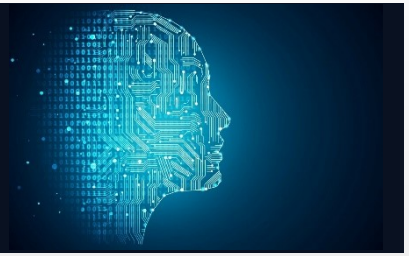


- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations:
  - **Simulation Evaluation**

- "Traditional" machine learning used to evaluate a model's performance base on k-fold cross validation
- In the example below 10-fold cross validation
- Evaluation metric is usually accuracy for classification

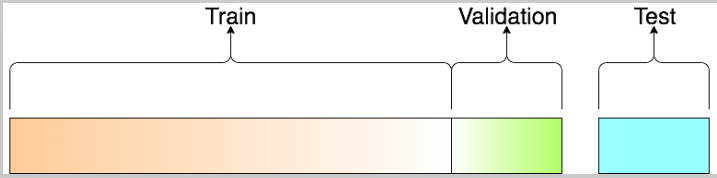


TENSORFLOW IN PYTHON



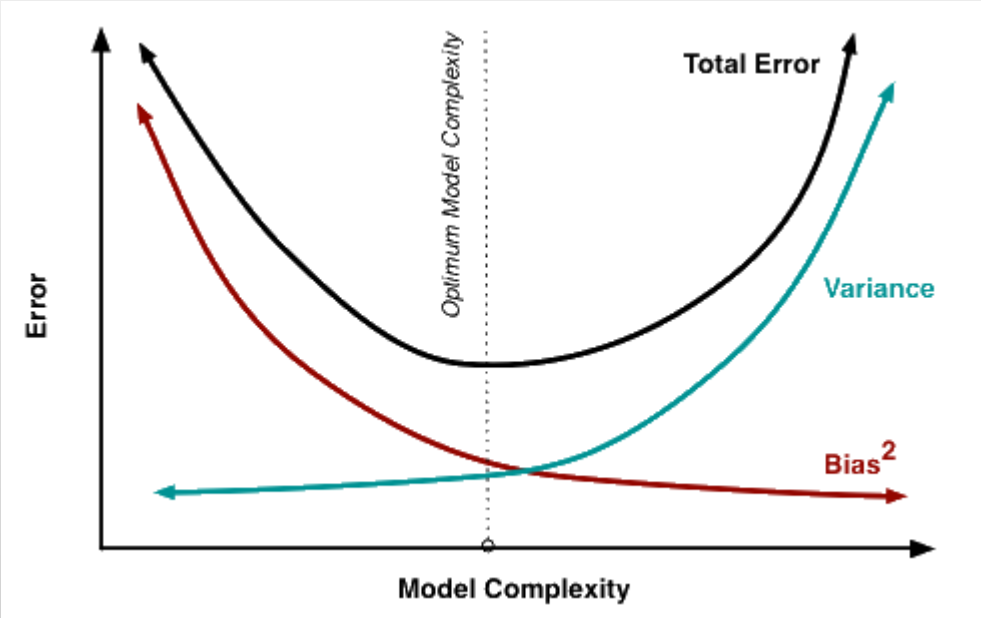
- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - Designing a model
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations:
  - **Simulation Evaluation**

- Deep Learning usually relies on evaluation by train\validation\test
- Train set is chosen as ~90% of the dataset. We choose batches of data from this and optimize
- The validation set is ~10% of the dataset. It is evaluated every N steps of gradient descent.
- We tune our simulation according to this set therefore it indirectly influences model success
- The test set is an independent set of unbiased samples, it should be chosen carefully such that it is representative of “real world” data



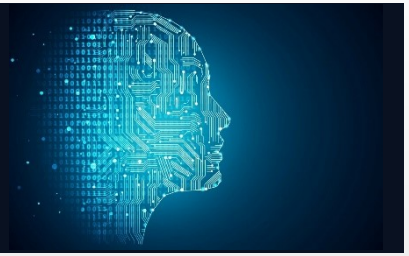
TENSORFLOW IN PYTHON

- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - **Designing a model: Bias Variance Tradeoff**
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - **Simulation Evaluation**



Train	**2%**	**14%**	**14%**	**1.5%**
Test	13%	16%	29%	2%
Human err ≈ 0	Overfitting	Underfitting	High bias and high variance	Low bias and low variance

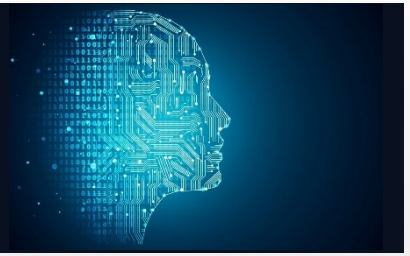
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - Reading files
  - Variables
  - **Designing a model: Bias Variance ~~Tradeoff~~**
  - Batch optimization
  - Epochs and Randomization
  - Dimensions
  - Visualizations
  - **Simulation Evaluation**

- Deep learning has shown us that the bias variance tradeoff does not necessarily need to be a strict tradeoff.
- In many real world applications, when working without big data algorithm designers tune and experiment with various paramaters\methods which usually introduce a tradeoff
- Deep learning has shown that one can attack the bias problem and the variance problem individually!
- Bias may be overcome by bigger and better neural networks
- Variance may be overcome by introducing more data and by regularization (more on this in the next lecture)
- This is **why** deep learning has taken off!

TENSORFLOW IN PYTHON

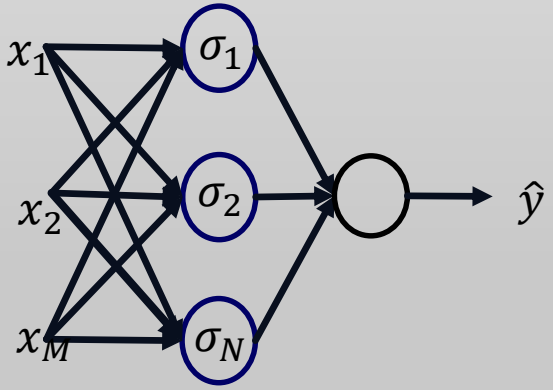


- Activation functions: Why?
  - **Universal Approximation Theorem**
  - Nonlinearity in practice

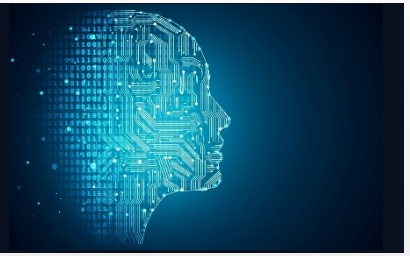
Every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron (MLP) with just one hidden layer, with non linear activation functions.

-George Cybenko 1989 (sigmoid activation)

$$\hat{y}(x) = \sum_i v_i \cdot \sigma(w_i \cdot x + b_i)$$



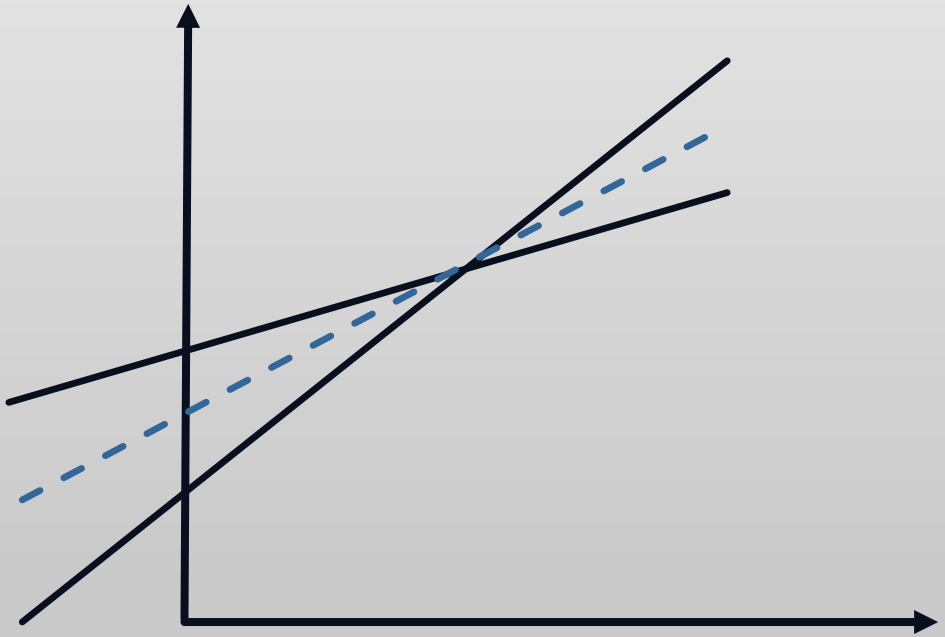
TENSORFLOW IN PYTHON



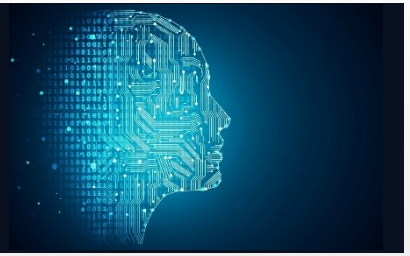
- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**

Why non-linearity?

- Sum of linear functions



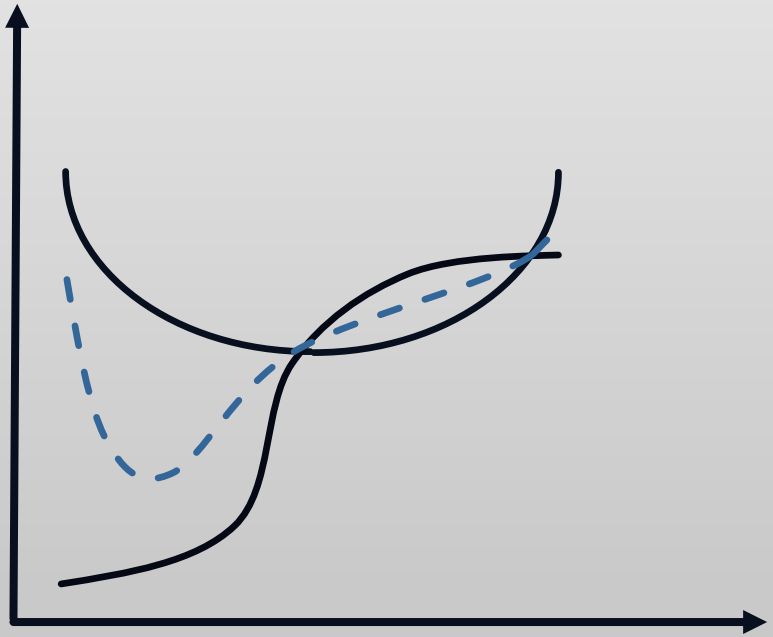
TENSORFLOW IN PYTHON



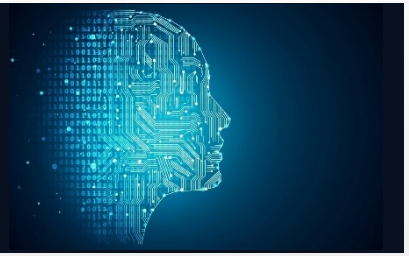
- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**

Why non-linearity?

- Sum of non-linear functions



TENSORFLOW IN PYTHON



- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**

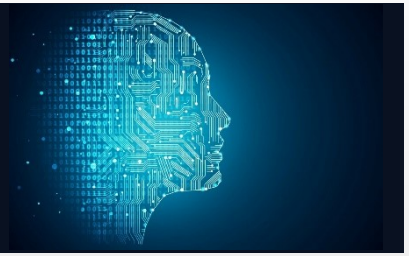
- The general approximation theorem does not specify how to calculate the approximation function

"A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly"

-Ian Goodfellow

- Theoretically approximating any continuous function is **possible**, but may require arbitrarily large amounts of data and nodes, making such a neural net unfeasible in practice

TENSORFLOW IN PYTHON



- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**

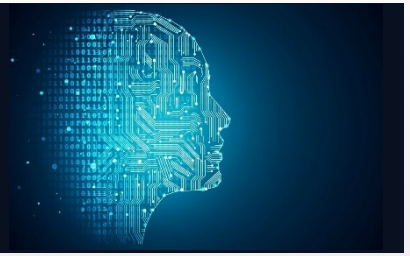
- The general approximation theorem does not specify how to calculate the approximation function

"A feedforward network with a single layer is sufficient to represent any function, but the layer may be infeasibly large and may fail to learn and generalize correctly"

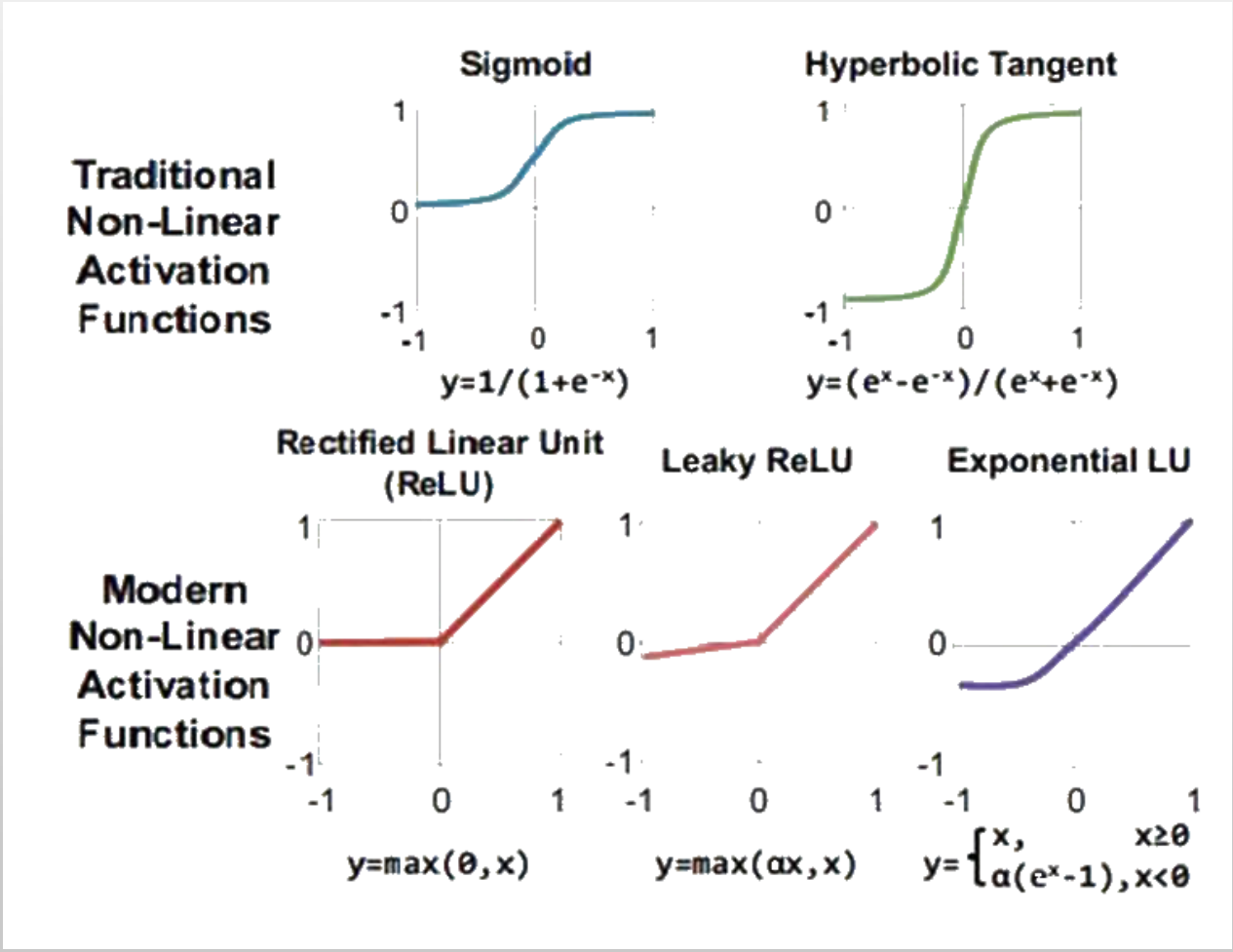
-Ian Goodfellow

- Theoretically approximating any continuous function is **possible**, but may require arbitrarily large amounts of data and nodes, making such a neural net unfeasible in practice

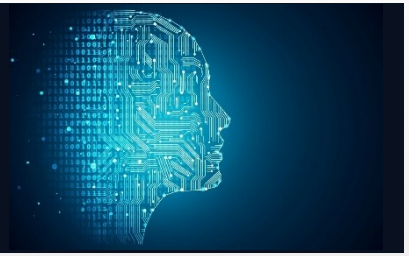
TENSORFLOW IN PYTHON



- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**



TENSORFLOW IN PYTHON



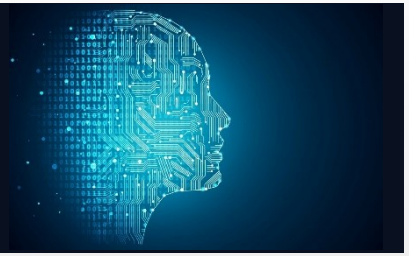
- Activation functions: Why?
  - Universal Approximation Theorem
  - **Nonlinearity in practice**

- Softmax function is another important function in practice, especially for classification
- Softmax is a “differentiable max() function”

$$Softmax(x) = \frac{e^{x_0}}{e^{x_0} + e^{x_1} + \cdots + e^{x_N}} + \cdots + \frac{e^{x_N}}{e^{x_0} + e^{x_1} + \cdots + e^{x_N}}$$

- Softmax is bounded between 0 and 1

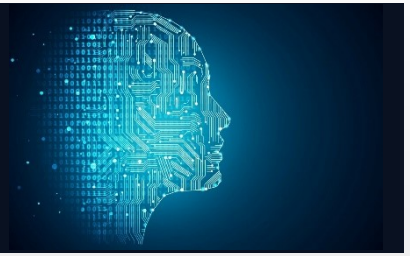
TENSORFLOW IN PYTHON



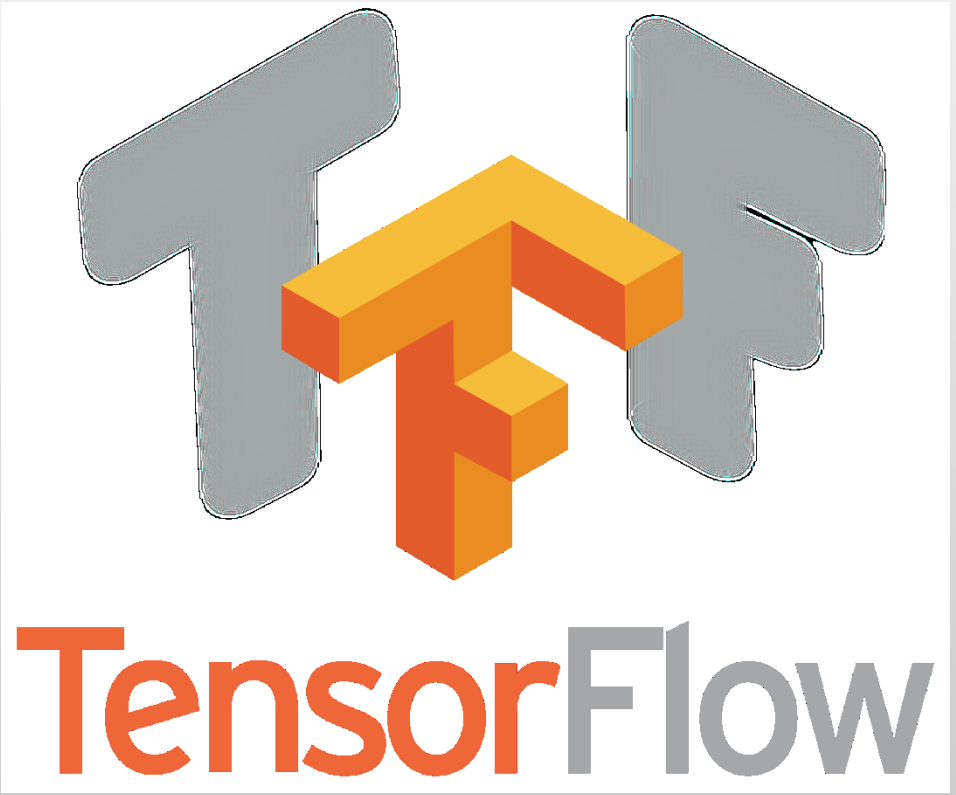
- One hot encoding
- This is another “practical” aspect of deep learning
- Instead of specifying that $y \in \{0,1,2 \ldots\}$, a specific integer class, we transform this representation into a vector with zeros in every entry except the integer value of the class.
- For example, lets say we want to classify 2 classes, cats and dogs. Lets assume cats are assigned class 0 and dogs are assigned class 1.
- We will represent cats via one-hot encoding as

$y = [1,0]$ and dogs as $y = [0,1]$

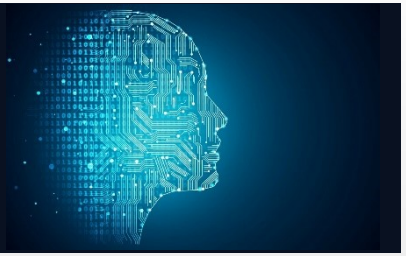
TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - MNIST Classification competition



TENSORFLOW IN PYTHON



- Key concepts in deep learning simulations
  - MNIST Classification competition
  - Add ***Tensorboard*** summaries
    - Loss
    - Accuracy
    - Activation histogram
  - Add ***feature space visualization***
    - Show 2d\3d projection of feature space using PCA as projection