



Corey Stewart – Graduate Student ( M.S. )  
Auburn University – Electrical and Computer Engineering  
Email: ces0171@auburn.edu  
LinkedIn: <https://www.linkedin.com/in/corey-s/>

“A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .” - Tom M. Mitchell et al. “Machine Learning”

## Overview

- The A.I. / M.L. Landscape and Day-to-Day Applications
- M.L. Brick-by-Brick
- Radar Specific Applications



**M.L.** → Computer Assisted Modeling, Prediction, and Statistics (C.A.M.P.S.)

vs.

**A.I.** → Computer-Agents making decisions to best navigate their environments

Video URL: [https://www.youtube.com/watch?v=owI7DOeO\\_yg](https://www.youtube.com/watch?v=owI7DOeO_yg)

My personal view of Machine Learning (ML) is that it is namely computer assisted modeling, prediction, and statistics that allows humans to explore potential decisions and potential relationships between concepts in novel ways. This differs from Artificial Intelligence (A.I.) because A.I. seeks to have a computer-agent make decisions that allow the computer-agent to best navigate their target environments.

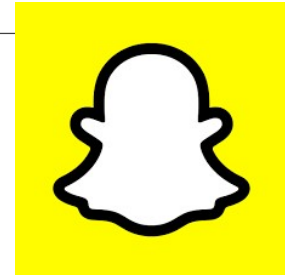
# Day-to-Day Machine Learning



Google Lens



Kasparov vs. Deep Blue (1996 ~ 1997)



Snapchat



## Convolutional Neural Networks (CNNs)

[ Processing Images ]

- Google Lens
- Snapchat

## Unsupervised Learning (K – Nearest Neighbors Clustering)

[ Computer Discovering and Creating Groupings ]

- A potential approach to recreating YouTube (/ Targeted Advertising)

## Recurrent Neural Networks (RNNs)

[ Analyzing Sequential Data ]

- Voice Recognition (Alexa, Siri, Google Assistant, etc.)
- Stock Analysis and Trading

## Reinforcement Learning:

- Deep Blue (used more of a brute force approach)
- AlphaZero (Able to play the games of Chess, Shogi, and Go)
- DeepMind (Played Starcraft2 at a professional gaming level)
- OpenAI (Played Dota2 at a professional gaming level)

Mini-Project (Machine Learning in Action):

- 1.) Using Google Images perform a search using the term “fruit”
- 2.) Right click on an isolated fruit of your choosing
- 3.) Select “Search Image with Google Lens”

Your results may vary, but Google Lens should be able to classify the fruit (or the photo’s original website).

I was able to successfully do this using the Google Chrome browser on my laptop and on my phone, a Google Pixel 5.

## Rosenblatt's Perceptron

- 1958 [ F. Rosenblatt ]:  
Cornell Aeronautical Laboratory  
\*Simulations on an IBM 704
- 1969 [ Minsky and Papert ]:  
“*Perceptrons*” the book  
(a.k.a. “This can’t work...”)
- 1980s ~ 1990s:  
“*Perceptrons – Expanded Edition*”  
(a.k.a. “This might work!”)

Slope - Intercept Form of a Line:

$$y = mx + b$$

### The Perceptron Idea:

The “Perceptron” was an electronic device which was constructed in accordance with biological principles and showed an ability to learn. Rosenblatt’s device showed some ability to register objects such as triangles.

### Limitations:

Minsky and Papert produced a mathematical proof detailing the limitations of two-layer networks. They were also under the impression that training several layers of a network would be difficult.

### A Familiar Equation:

The slope-intercept form of a line is somewhat foundational in the field of machine learning.

# Reframe of $y = mx + b$

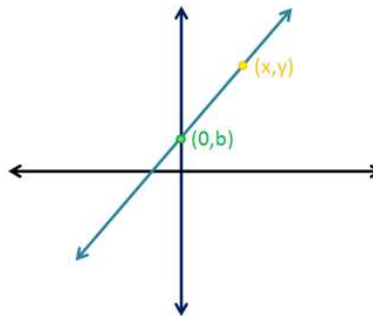
$y$  → Outcomes / Predictions / Targets

$m$  → Weights

$x$  → Inputs / Features / Parameters

$b$  → bias (/ threshold on occasion)

\* *Line of Best Fit* \*



$m$ : slope  
 $(x,y)$ : any point on the line  
 $(0,b)$ : y-intercept of the line

---

$$m = \frac{y - b}{x - 0}$$
$$mx = y - b$$
$$y = mx + b$$

Source Link: <http://mathandmultimedia.com/2011/03/18/equation-of-a-line/>

$y$ ,  $m$ ,  $x$ , and  $b$  essentially become vectors from this point forward.

These variables also take on different names:

$y$  → Predictions

$x$  → Features

$m$  → Weights

$b$  → Biases

A major aim of machine learning is to create a “*something* of best fit”. That “*something*” may be a curve through multi-dimensional space (housing price prediction or stock prediction) or grouping data intelligently in those same high-dimension environments (YouTube or Netflix user demographics).

## Rosenblatt's Perceptron

- 1958 [ F. Rosenblatt ]:  
Cornell Aeronautical Laboratory
- 1969 [ Minsky and Papert ]:  
"Perceptrons" the book  
(a.k.a. "This can't work...")
- 1980s ~ 1990s:  
"Perceptrons - Expanded Edition"  
(a.k.a. "This might work!")

Slope - Intercept Form of a Line:

$$y = mx + b$$

Slope - Intercept Form of a Line  
with extra steps (a.k.a. The Activation Function):

$$f(x) = \begin{cases} 1, & \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0, & \text{otherwise} \end{cases}$$

Fixing the Equation:

Home in on the [  $mx + b$  ] portion and transform it into [  $wx + b$  ], which is a better representation of what one might find in the ML field.

Using Thresholds:

An early version of a perception may have used "thresholding" to elicit signals when certain conditions have been met. The function that governs when the output will when provided with a particular [  $wx + b$  ] input in the context of ML is often call the "Activation Function".

# My Mother's Car

( I spy a Logical AND! )

$$f(x) = \begin{cases} 1, & \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{x} = \{S, P\}$$

$$\mathbf{w} = \{w_s, w_p\} = \{4, 1\}$$

$$b = -4$$

$$f(x) = w_s S + w_p P + b$$

$$f(S, P) = 4S + P - 4$$

| "Start" Button Pushed (S) | Brake Pedal Pushed (P) | Can Move Gear Selector (y) |
|---------------------------|------------------------|----------------------------|
| 0                         | 0                      | 0                          |
| 0                         | 1                      | 0                          |
| 1                         | 0                      | 0                          |
| 1                         | 1                      | 1                          |

A Real-World Scenario:

To move the gear selector from Park to Drive in my mother's car, the start button must be pushed, and the brake pedal must be pushed.

Modeling:

Using the perceptron framework, that situation can be modeled as a function.

A Touch of History:

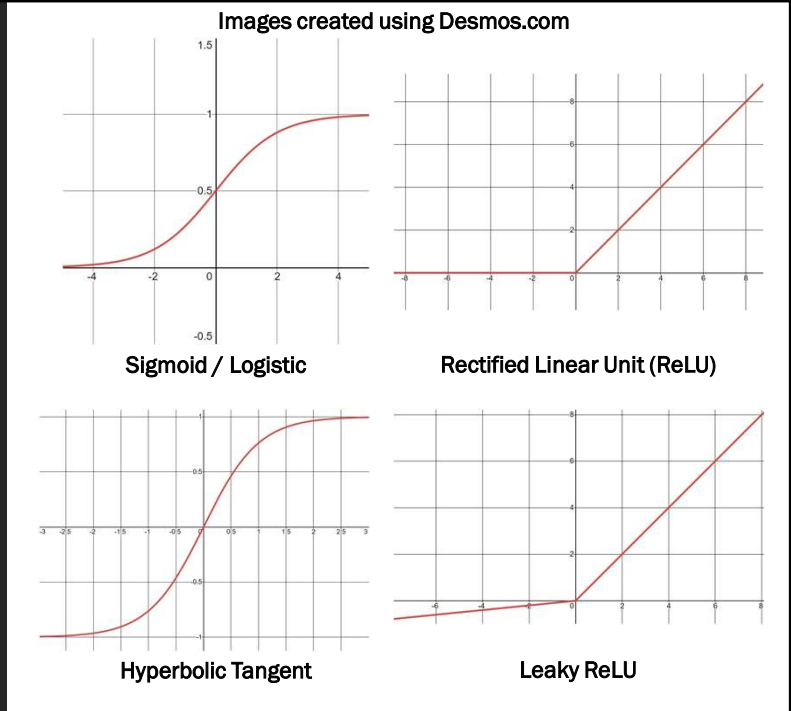
In their 1969 book, Minsky and Papert were under the impression that perceptrons would not be able to create XOR logic. However, their information sources must not have been the greatest because it was proven in ~1964 that perceptrons could produce XOR logic!



# Activation Functions

- There are a wide variety of activation functions tailored to suit specific types of ML problems.
- Sigmoid → Classification Systems
- ReLU → Increased “Learning” Speeds

List of Common Activation Functions  
[https://en.wikipedia.org/wiki/Activation\\_function](https://en.wikipedia.org/wiki/Activation_function)



The framework used in the previous slide relied on a binary output for the  $[ wx + b ]$  input, but the latest activation functions are incredibly varied and allow for many real-valued outputs.

# Rosenblatt's Perceptron

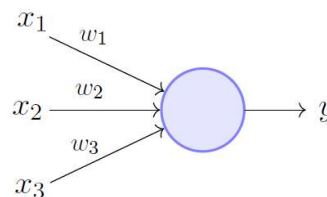
- 1958 [ F. Rosenblatt ]:  
Cornell Aeronautical Laboratory
- 1969 [ Minsky and Papert ]:  
"Perceptrons" the book  
(a.k.a. "This can't work...")
- 1980s ~ 1990s:  
"Perceptrons - Expanded Edition"  
(a.k.a. "This might work!")

Slope - Intercept form of a Line:

$$y = mx + b$$

Slope - Intercept form of a Line  
with extra steps (a.k.a. The Activation Function):

$$f(x) = \begin{cases} 1, & \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0, & \text{otherwise} \end{cases}$$



Perceptron Model (Minsky-Papert in 1969)

Source Link: <https://towardsdatascience.com/what-is-a-perceptron-210a50190c3b>

A Single Perceptron (or Node):

The final version of this slide introduces a visual representation of a perceptron with all its elements.

The General Process:

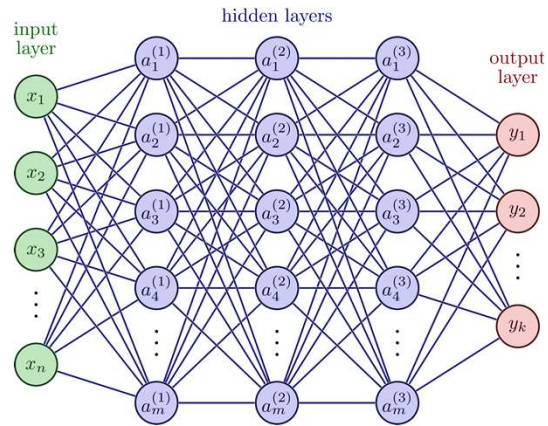
- The "x" layer provides inputs.
- The "w" layer scales those inputs.
- The perceptron or "node" adds these scaled inputs together, adds the necessary bias value, and sends that result through the activation function.
- The node output, "y", is the generally the output of the activation function.

The Dynamic Duo:

The key element that Minsky and Papert missed was the effect of layering. They did not believe that layering perceptrons beyond 2 layers could ever server a useful purpose. Time showed them to be incorrect, but there was a logic to their thought process. Rarely does adding two or more bad things together produce a good thing.

# ( Deep ) Neural Networks

1. Input Layer:  
Features / Parameters of the system
2. Hidden Layers:  
Layer(s) of nodes connected to some (or all) nodes in a previous layer
3. Output Layer:  
The final layer from which information will be extracted to yield a desired result



A Lot of Bad Things (a.k.a. A Lot of Layers):

Sometimes, you need several bad things systemically layered in a matrix format with multiple layers to produce an amazing thing!

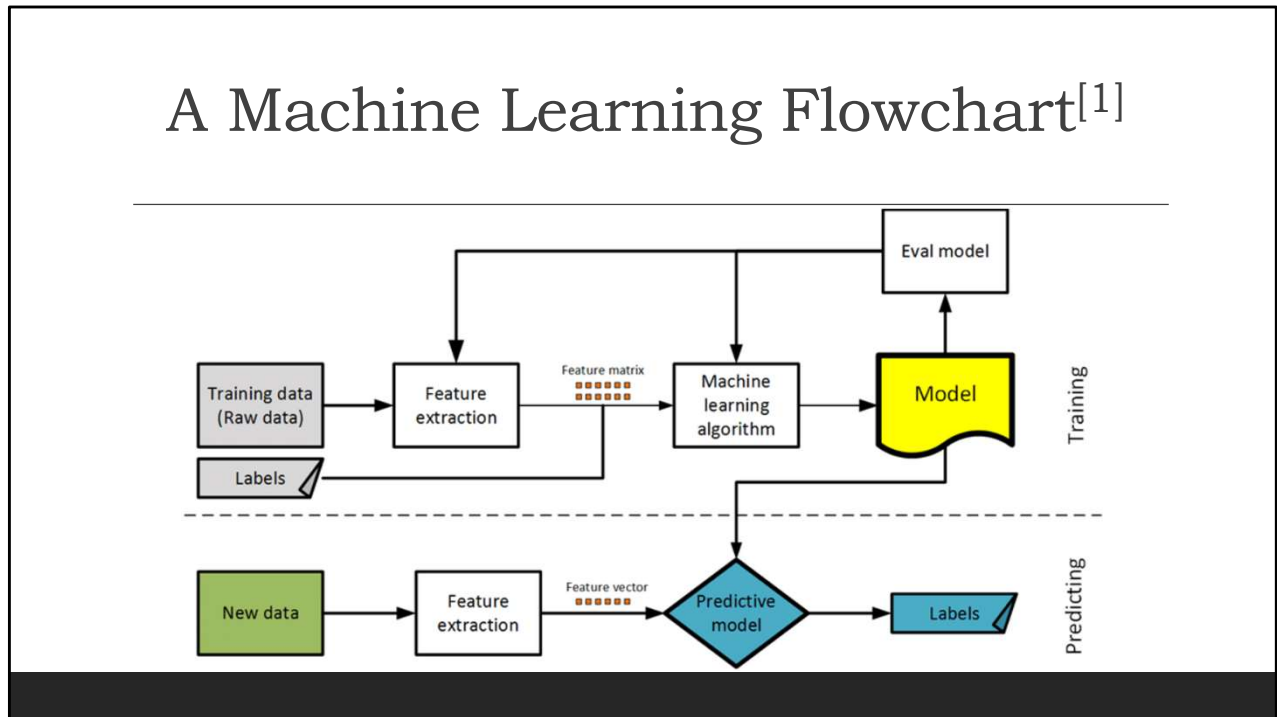
A Network Like Me:

This style of network is very generic, but it's most likely a classification network of some kind. It could be many things, but it's possible to envision  $Y_1$  as being a "Person",  $Y_2$  a "Car", and  $Y_k$  as "The Statue of Liberty"! If those things were in this network's training set, such classifications wouldn't be impossible for a network configuration like the one shown.

# Summary of Machine Learning Foundations

| Input Layer  | Hidden Layer(s)   |   | Output Layer   |
|--|---|---|--|
| <p><b>Layer 1</b></p> <p>Preprocessed Features, Parameters, or Inputs</p> <p>(usually represented as a vector)</p> $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_3 \end{bmatrix}$ | <p><b>Layer 2</b><br/>(<math>H_1</math>)</p> <p>Pre-Activation Function<br/><math>Z_{n_l, l} = \mathbf{w}_{n_{l-1}, l-1} \cdot \mathbf{x} + b</math></p> <p>Activation Function<br/><math>A_{n_l, l} = G_l(Z_{n_l, l})</math></p> | <p><b>Layers 3 through L - 1</b><br/>(<math>H_2</math> through <math>H_M</math>)</p> <p>Pre-Activation Function<br/><math>Z_{n_l, l} = \mathbf{w}_{n_{l-1}, l-1} \cdot \mathbf{A}_{l-1} + b</math></p> <p>Activation Function<br/><math>A_{n_l, l} = G_l(Z_{n_l, l})</math></p> | <p><b>Layer L</b><br/>(Output Layer)</p> <p>Pre-Activation Function<br/><math>Z_{n_L, L} = \mathbf{w}_{n_{L-1}, L-1} \cdot \mathbf{A}_{L-1} + b</math></p> <p>Network Output<br/><math>\mathbf{O} = G_L(Z_{n_L, L})</math></p> |

# A Machine Learning Flowchart<sup>[1]</sup>



## A Possible ML Flow:

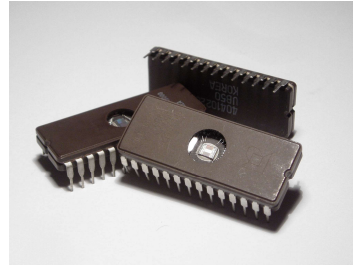
- Training data / Raw data is collected from some source
- Important features are manually or automatically extracted (e.g. parameters for a smart microwave vs. Facebook user demographics)
- Labels may or may not be applied (Supervised Learning vs. Unsupervised Learning)
- The appropriate ML algorithm for the problem is used to produce a model
- The last model of training become the predictive model that will operate on completely new data

## Eval Model:

Model evaluation can be somewhat complex, and it is a topic that is saved for the Model Evaluation section later on.

“Rome wasn’t built in a day.”

- By current ML standards, we’ve analyzed a single brick, but a brick in isolation does not make a castle!
- Next Up:
  - Machine Learning Family Tree
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)

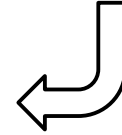


Integrated Circuit

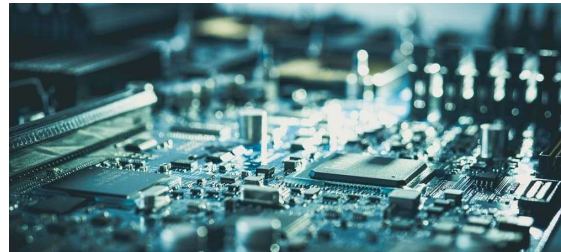
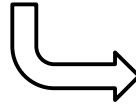
Source Link:

[https://en.wikipedia.org/wiki/Integrated\\_circuit](https://en.wikipedia.org/wiki/Integrated_circuit)

This...



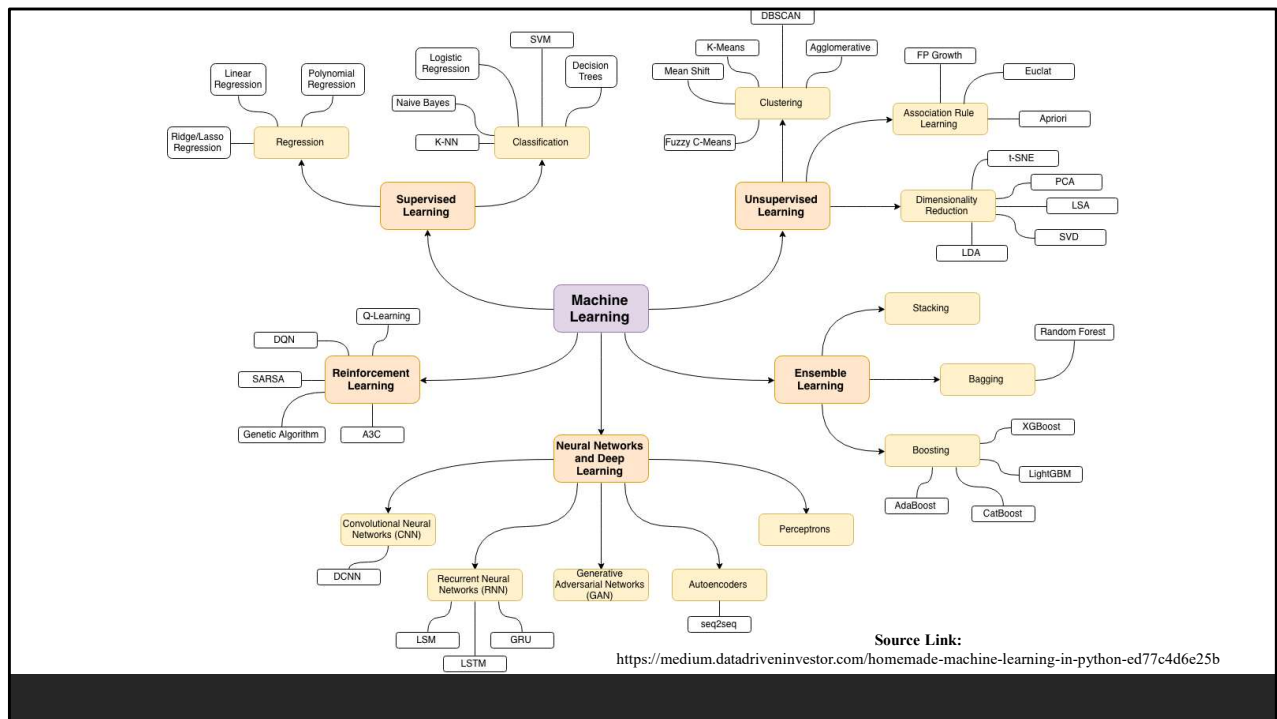
...is not exactly that.



Motherboard

Source Link:

<https://www.hp.com/us-en/shop/tech-takes/what-does-a-motherboard-do>



### Supervised Learning:

The model is trained on labeled data with the hope that when the model makes a prediction for a “new” input, the appropriate label for the data will be chosen.

- Linear Regression → predicting house prices based on zip code, square footage, bedroom and bathroom #, etc.
- Classification → detecting fraudulent credit card activity based on purchase history

### Unsupervised Learning:

The model is allowed to determine relationships for itself between inputs, features, and combinations of features.

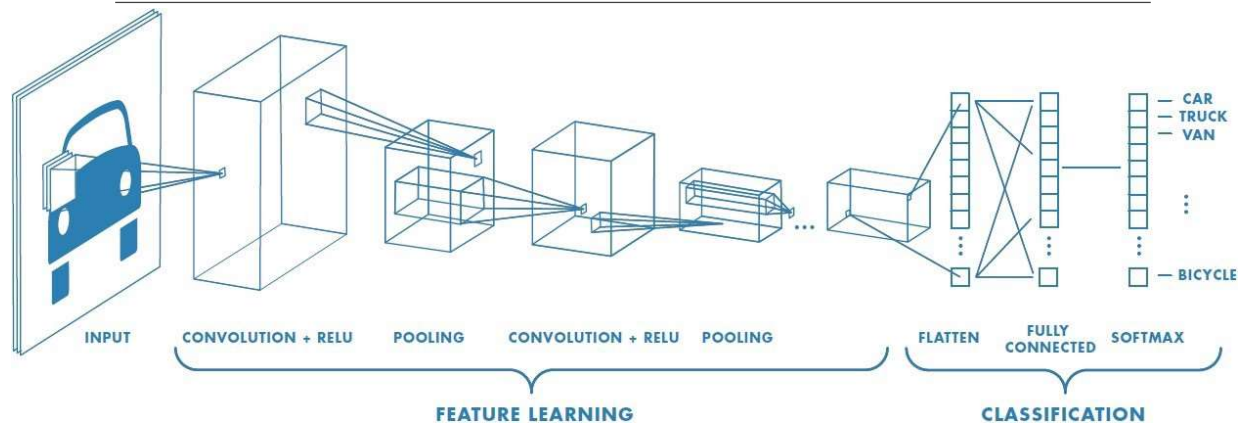
- YouTube and Netflix → suggesting videos based on certain user demographics or being ahead of the curve and finding new demographics that are gaining traction.

### Reinforcement Learning:

The model is provided rewards or penalties as it performs a task. Rewards or penalties are given to incentivize certain behavior patterns.

- Chess, StarCraft2, Pong, Mario Kart → Video games naturally provide environments with rewards and penalties. As a result, many RL algorithms are evaluated in game-like environments.

# Convolutional Neural Networks ( Feature Learning )



Source Link: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

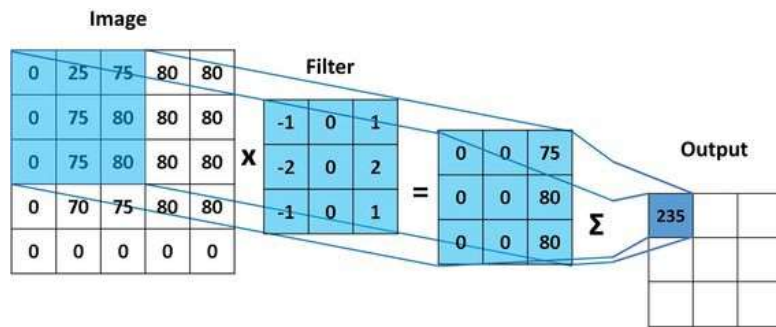
Convolutional Neural Networks (CNN) are powerful tools that have found a home in the image analysis space.

A Possible Architecture:

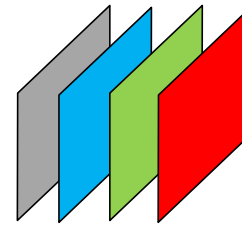
Though incredibly varied, a basic architecture for a CNN may include an image convolution step (with a ReLU activation function) followed by pooling. That general process would be repeated for as many times as desired before being connected to a classification network (e.g. a network like the one shown in slide 10).



# CNNs: Convolution



Source Link: <https://medium.com/@kinisanketh/getting-started-with-cnn-18c03efc7d06>



Possible Image Filters

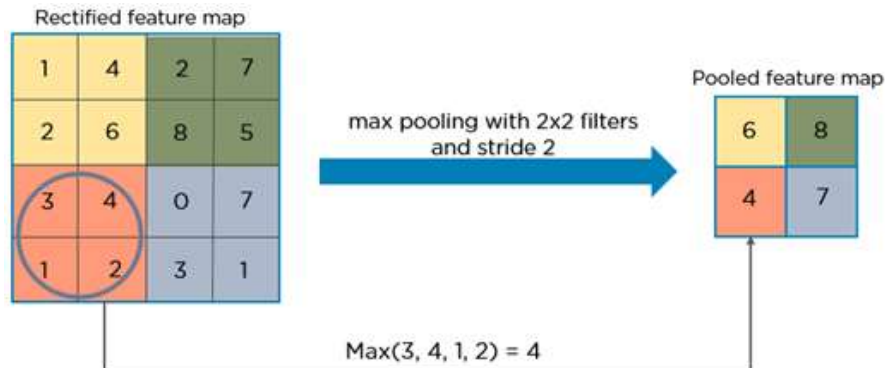
- Color Detection
- Edge Detection
- Shape Detection

Filters are used during the image convolution process to spot artifacts of note such as edges, certain shapes, and certain colors.

Explaining the Dimensions of the Output:

- Take a standard RGB image that is 200 x 200 pixels.
- Assume 15 filters that are each 10 x 10 pixels are required for this CNN problem.
- Using  $(S_i - S_f + 1)$ , the output image should be 191 x 191 pixels for a single filter.
- Thus, the output's result becomes a block of the size 191 x 191 x 15 pixels.

# CNNs: Pooling



Source Link: <https://medium.com/@kinisanketh/getting-started-with-cnn-18c03efc7d06>

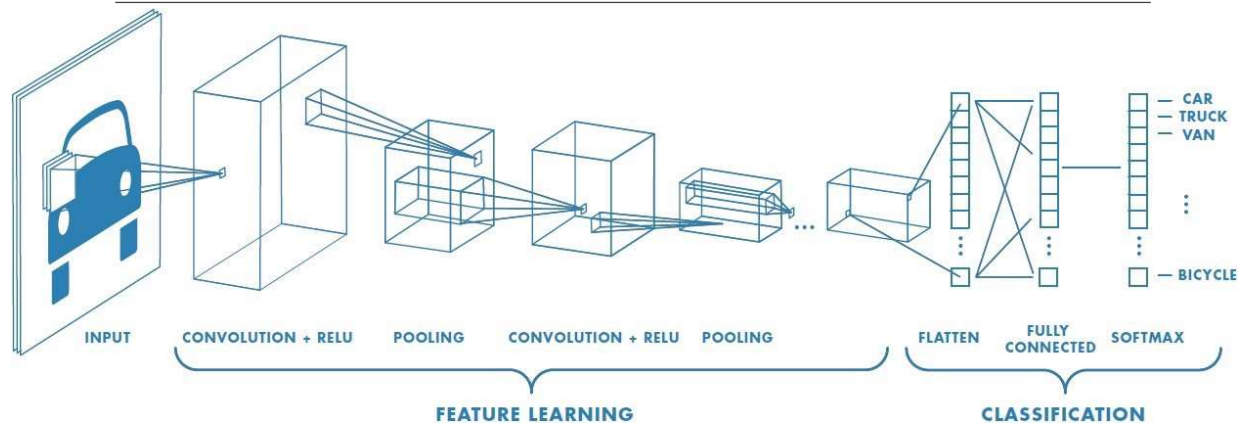
## Max Pooling:

This technique is used to highlight features that are most prominent despite the possible orientation of the image. If the network being built is a tiger classifier for example, the filters for the colors “black” and “orange” may yield exceptionally high values in certain areas of the image. It would be desirable to keep those “high”, “prominent”, and “important” values throughout the network as it begins to learn how to associate those two colors in later network layers. Under these circumstances, that relationship may still be learned whether the tiger is climbing a tree, hidden by brush, closer to the background of the image, or closer to the foreground of the image.

## Average Pooling:

This technique is like max pooling, but the average value in a region is maintained as opposed to the max value. Depending on the data and use case, average pooling may maintain important features that max pooling ignores.

# Convolutional Neural Networks ( Classification )



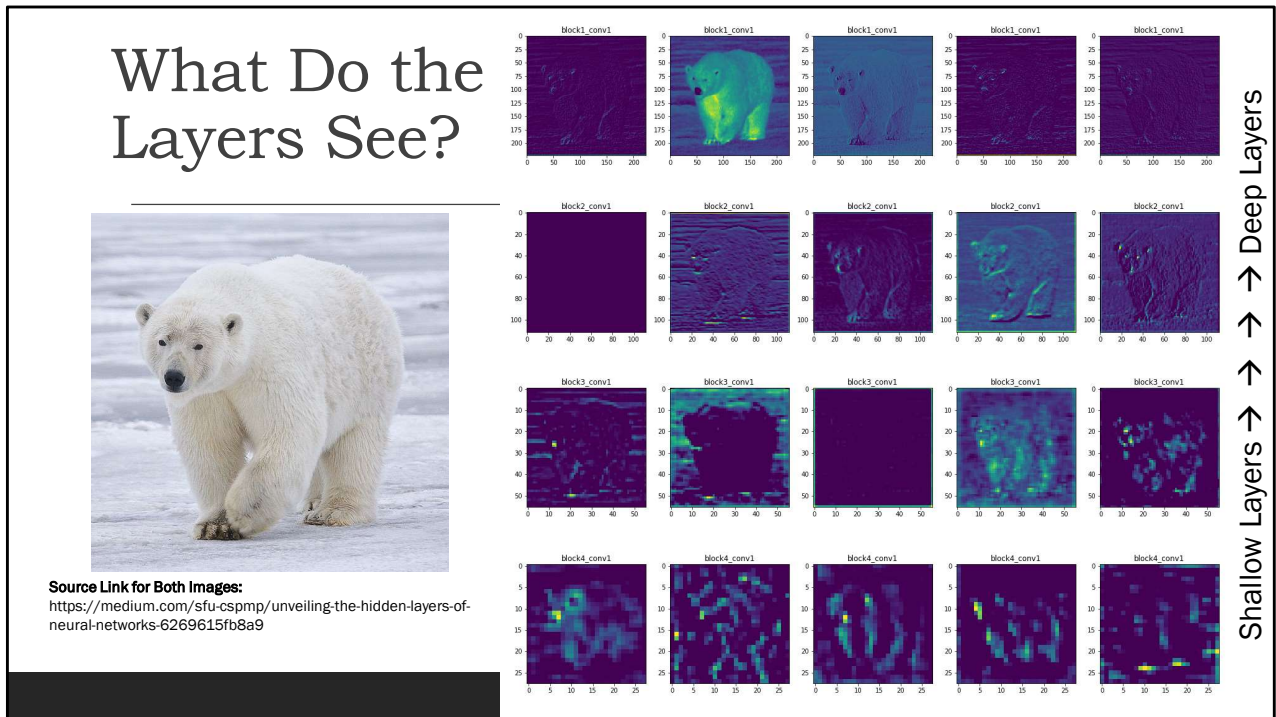
Source Link: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

The Classification Block:

Flattening is (normally) treating every pixel of the final block as an input into a network. For example, an output block that is 100 x 100 x 15 pixels would become a vector of size 1 x 150,000 - which isn't unreasonable for a system!

The Fully Connected part just means that every node in a previous layer is connected to every node in the next layer.

The softmax layer is like the output layer from slide 10 and would most likely use a sigmoid / logistic activation function. In a simple CNN implementation, every output node may correspond to precisely one "object" if the goal is object detection (e.g., Car, Truck, Van, Bicycle, Tiger, etc.).



**Layers Vision:**

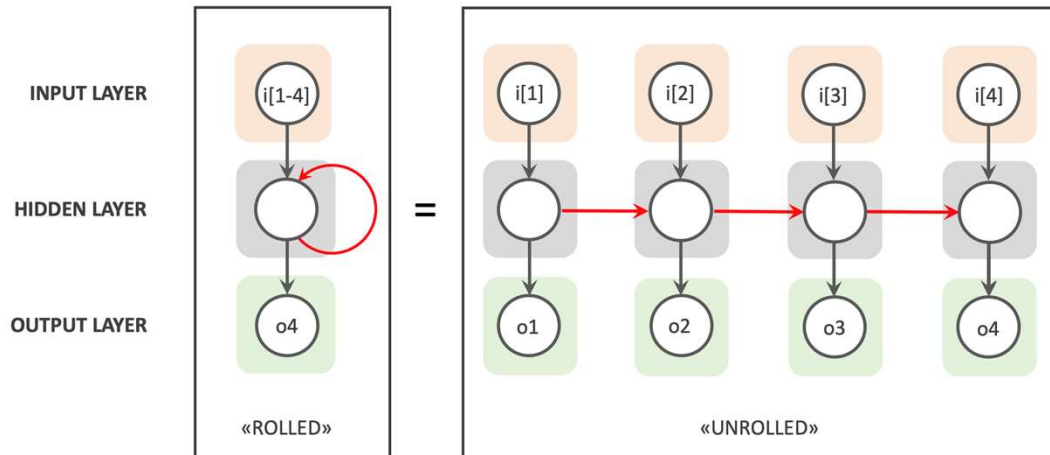
If one visualizes the outputs from the hidden layers, appropriate visualization implementations may show something like the flow below.

- Early / Shallow Layers → Low level features such as edges and areas of stark color contrast.
- Middle Layers → These layers may begin to pick up on key features (e.g. paws and limbs of a polar bear)
- Deep Layers → These layers may notice higher level features or features that provide immense detail (e.g. nose, ears, and eyes of a polar bear)

In the graphic from the slide, the deepest layer shown may be doing a few things.

- It may be providing “context” for the polar bear by analyzing the background in greater detail
- It may be evaluating very fine image features such as texture (e.g. fuzzy polar bear fur vs. smooth or coarse ice)

# Recurrent Neural Networks



Source Link: <https://www.bouvet.no/bouvet-deler/explaining-recurrent-neural-networks>

## Recurrent Neural Networks (RNNs):

These networks analyze sequential data. RNNs allow for a great deal of flexibility regarding the sizes of their inputs. Regarding the graphic from the slide, a potential network approach could be to have every input at every time step produce an output that will then be a part of the total output (many – to – many). Another network approach could rely on generating an output only after the entire sequence has been sufficiently analyzed by the network ( many – to – one ). The latter case may act as a sentiment classifier, where a program must determine whether a statement is “relatively positive” or “relatively negative”.

## “Who Traveled Where?” A Traveler’s Dilemma

### Potential Approach

- Vocabulary of decent size so words can be one-hot encoded.
- Feed input  $I_1$  into a Neural Network at timestep  $T_1$ .
- Maintain some aspects of network output  $Y_1$  to use alongside input  $I_2$  during timestep  $T_2$ .

Input Sentence: “George went to Spain.”

| x[1]   | x[2] | x[3] | x[4]  |
|--------|------|------|-------|
| George | went | to   | Spain |

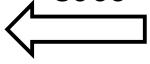
### Vocabulary and Representation

$$V = \begin{bmatrix} \text{Aardvark} \\ \vdots \\ \text{George} \\ \vdots \\ \text{Spain} \\ \vdots \\ \text{to} \\ \vdots \\ \text{went} \\ \vdots \\ \text{Zyzyyva} \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 8965 \\ \vdots \\ 52467 \\ \vdots \\ 58113 \\ \vdots \\ 66429 \\ \vdots \\ 80000 \end{bmatrix}$$

### One-hot Encoded Form of “George”

$$x[1] = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

A “1” at position 8965



Assume a task encompasses analyzing pieces of literature to determine the travelers in an article and the places they visited.

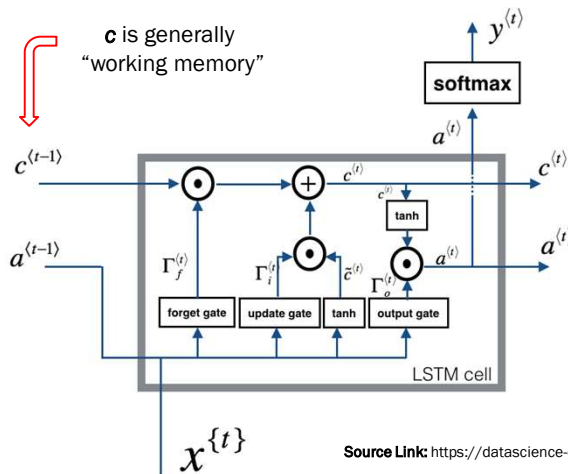
Perhaps travel agents or cruise line executives would find such information useful, and it may prove that providing free trips and vacations to celebrities - on the condition they “tweet” or blog about the trips - is a practical business strategy. (The greater goal being to search through social media posts and see if people book trips based on where celebrities say they are going or have been!)

Zyzyyva → A Zyzyyva is essentially a tropically beetle! More specifically, it’s a genus of South American weevils.

[ The statements below act as support for slide 22 ]

Using the “George went to Spain” example, the softmax layer could be composed of a “location” node and a “proper noun / name” node. The “proper noun / name” node may choose to remember “George” and largely ignore the other words in the original sentence. The “location” node would most likely have a higher-than-average activation on “went to” and be very high when it encounters “Spain”.

# RNN: Long Short-Term Memory (LSTM)



$$\begin{aligned} \Gamma_f^{(t)} &= \sigma(W_f[a^{(t-1)}, x^{(t)}] + b_f) \\ \Gamma_u^{(t)} &= \sigma(W_u[a^{(t-1)}, x^{(t)}] + b_u) \\ \tilde{c}^{(t)} &= \tanh(W_c[a^{(t-1)}, x^{(t)}] + b_c) \\ \dots \\ c^{(t)} &= \Gamma_f^{(t)} \circ c^{(t-1)} + \Gamma_u^{(t)} \circ \tilde{c}^{(t)} \\ \Gamma_o^{(t)} &= \sigma(W_o[a^{(t-1)}, x^{(t)}] + b_o) \\ a^{(t)} &= \Gamma_o^{(t)} \circ \tanh(c^{(t)}) \end{aligned}$$

This slide provides something of an insight into the Hidden Layer node(s) from slide 20.

Intuition behind Forget Gate:

This gate determines if information from the current input or previous activation is worth "forgetting" from the OLD working memory.

Intuition behind Update Gate:

This gate determines if information from the current input or previous activation should be part of the NEW working memory.

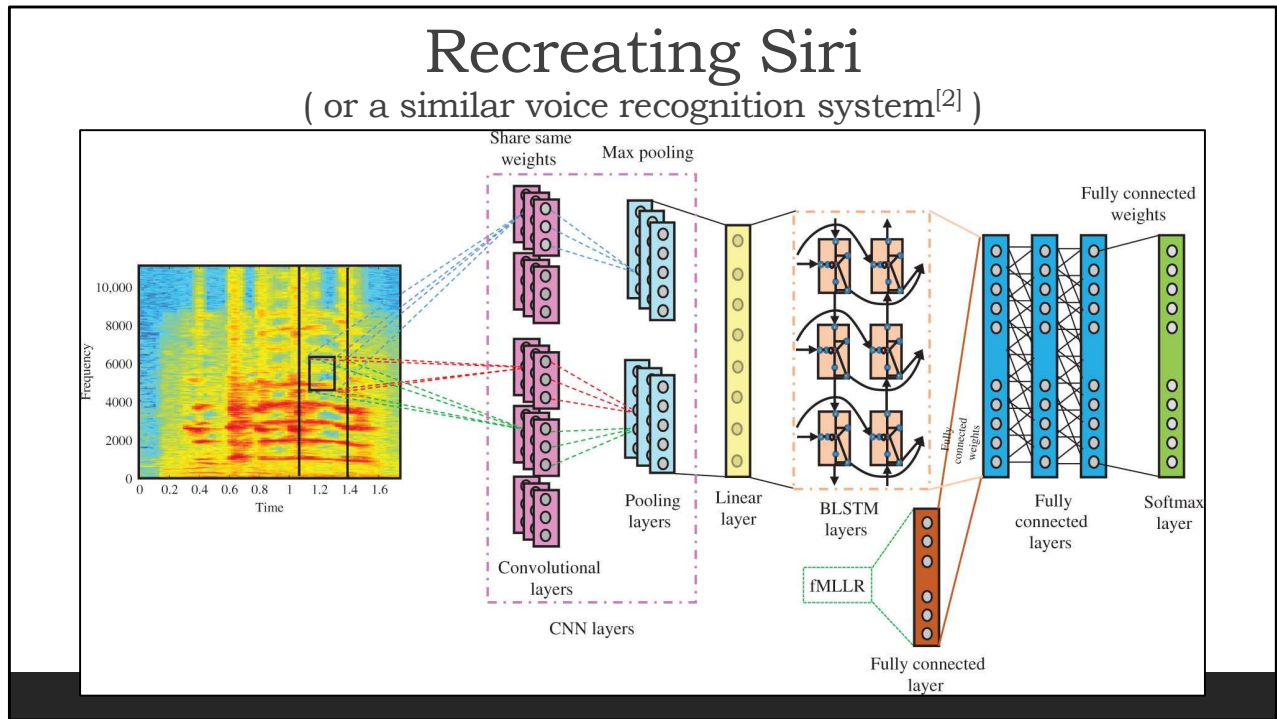
Intuition behind Output Gate:

This gate governs the importance of the current working memory based on the current input and previous activation.

IMPORTANT NOTE:

Upon close inspection, one may note that the LSTM cell activation is a function of the current working memory and not the input.

# Recreating Siri ( or a similar voice recognition system<sup>[2]</sup> )

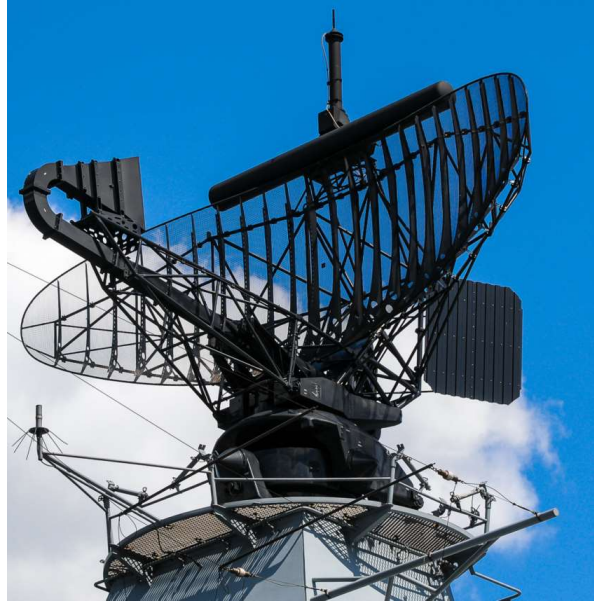


A system like the one depicted may analyze spectrogram information from a voice recording and accurately determine what was said. Such a system may also be able to suggest an appropriate reply to spoken questions, identify various speaker in the recording, or schedule a woman's haircut at a salon [ Google Duplex ]:

Google Duplex Demonstration Link:  
<https://www.youtube.com/watch?v=D5VN56jQMWM>



Radar  
Applications:  
Adaptive Radar



**Source Link:**

<https://www.kriegsschiffe.net/landingpage/main/elektronik/radar/lw-08/>

# Theory of Adaptive Radar<sup>[3]</sup>

Paper by L.E. Brennan and L.S. Reed ( 1973 )

## Section 1

Developing a theory for an adaptive processor that maximizes the probability of detection for a fixed false-alarm rate.

## Section 2 ( Mostly Mathematics )

Assume one has a vector (or matrix) of Signal + Noise data at certain timestamps. Have a filter vector of weights,  $W$ , that has been optimized to maximize detection probability with a fixed false-alarm rate.

## Section 3 - *Remove Ground Clutter*

## Section 4 [ Part 1 ]

The computation is expensive to perform in real-time, but if one can maximize the signal to noise ratio by maximizing  $W$  - which in turn requires knowledge of the noise covariance matrix  $M$  - then  $P_D$  is maxed.

# Theory of Adaptive Radar<sup>[3]</sup>

Paper by L.E. Brennan and L.S. Reed ( 1973 )

## Section 4 [ Part 2 ]

Estimate the noise covariance matrix  $M$  (and remember that  $W$  is a function of  $M$ ). Take the gradient of  $W$  and perform the process of "method of steepest ascent". After several radar sweeps,  $W$  approaches its max.

## Section 5

These methods converge (steady state) for an adaptive array antenna.

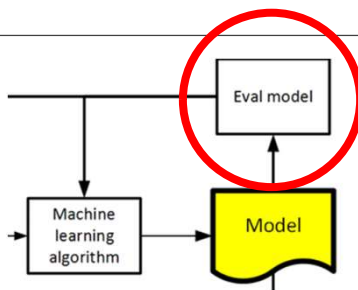
## Section 6

A simulation of the situation was generated and evaluated.  
[ Namely, after enough iterations 10 ~ 20 dB sidelobe reduction poor case ~35 dB better case ]

# The Machine is Broken

Model evaluation has many moving parts...

- Forward Propagation  
Cost / Loss Function
- Backward Propagation  
Gradient Descent
- Regularization  
Overfitting / Underfitting
- **The Right Network for the Job**



You make a model, you test it, and for some reason it thinks  $2 + 2 = 17$ !

On the fortunate side, you may have set up that model correctly!

On the unfortunate side, the model you created may not have the capacity to map your inputs to the desired output!

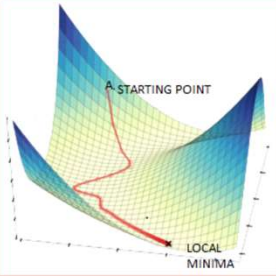
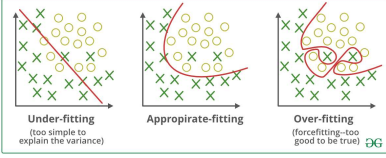
To Get Technical:

Forward Propagation → A forward pass through the network. The Left – to – Right behavior of slide 11.

Backward Propagation → A backward pass through the network. The Right – to – Left behavior of slide 11 used to update the weights and biases to more appropriate values.

Regularization → The act of scaling down certain feature values such that differences in orders of magnitude between features does not unduly skew the model toward a feature with a large value.

# Model Evaluation: “What’s the Worst that Could Happen?”

| Cost Function   | Gradient Descent   | Regularization  |
|---|--|---|
| $J = \frac{1}{2m} \sum_{i=1}^m (\hat{y} - y)^2$ <p>The function that acts as a measure of how far your model's predictions are from ground truth values during training.</p> <p>The equation above would be the Mean-Square-Error (MSE) for a Regression Model.</p> |  <p>Take a small step in the opposite direction of the activation function's maximum increase</p> |  <p>Underfitting may be a result of high model bias, which causes a model to miss key relationships</p> <p>Overfitting may be caused by high variance in a model, which makes a model attempt to fit input noise</p> |

Gradient Descent Source Link:

<https://www.hackerearth.com/blog/developers/3-types-gradient-descent-algorithms-small-large-data-sets/>

Regularization Source Link:

<https://www.geeksforgeeks.org/regularization-in-machine-learning/>

# You're Ready!



# References

---

1. Nguyen, Dong & Nguyen, Canh & Duong-Ba, Thuan & Nguyen, Hung & Nguyen, Anh & Trần, Tuấn. (2017). Joint network coding and machine learning for error-prone wireless broadcast. 1-7. 10.1109/CCWC.2017.7868415.
2. Passricha, Vishal and Aggarwal, Rajesh Kumar. "A Hybrid of Deep CNN and Bidirectional LSTM for Automatic Speech Recognition" *Journal of Intelligent Systems*, vol. 29, no. 1, 2020, pp. 1261-1274. <https://doi.org/10.1515/jisys-2018-0372>
3. L. E. Brennan and L. S. Reed, "Theory of Adaptive Radar," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-9, no. 2, pp. 237-252, March 1973, doi: 10.1109/TAES.1973.309792.

# Machine Learning Resources

---

- [ Python / R ]  
[kaggle.com/learn](https://kaggle.com/learn)
- [ MATLAB ]  
[mathworks.com/services/training.html](https://mathworks.com/services/training.html)
- [ ML Concepts by Stanford's Andrew Ng ]  
<https://youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShTMGgzqpfVfbU>
- [ Free to Learn – Only Pay for the Certificate ]  
[coursera.org/learn/machine-learning](https://coursera.org/learn/machine-learning)