Unit-2

Deep Feedforward Network

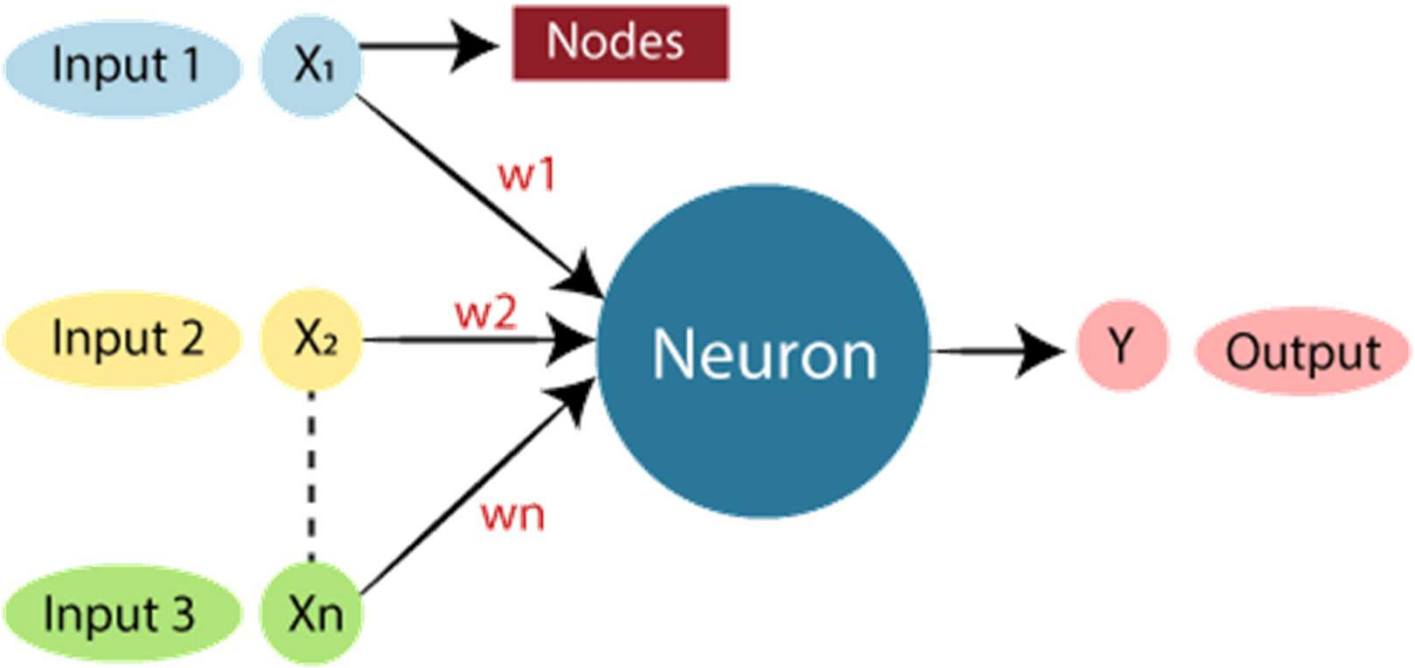
Deep Learning

BTCS 704-18

Feed-Forward Networks

- A feedforward neural network is a type of artificial neural network in which nodes' connections do not form a loop.
- Often referred to as a multi-layered network of neurons, feedforward neural networks are so named because all information flows in a forward manner only.
- The feedforward neural network is the simplest type of artificial neural network which has lots of applications in machine learning.

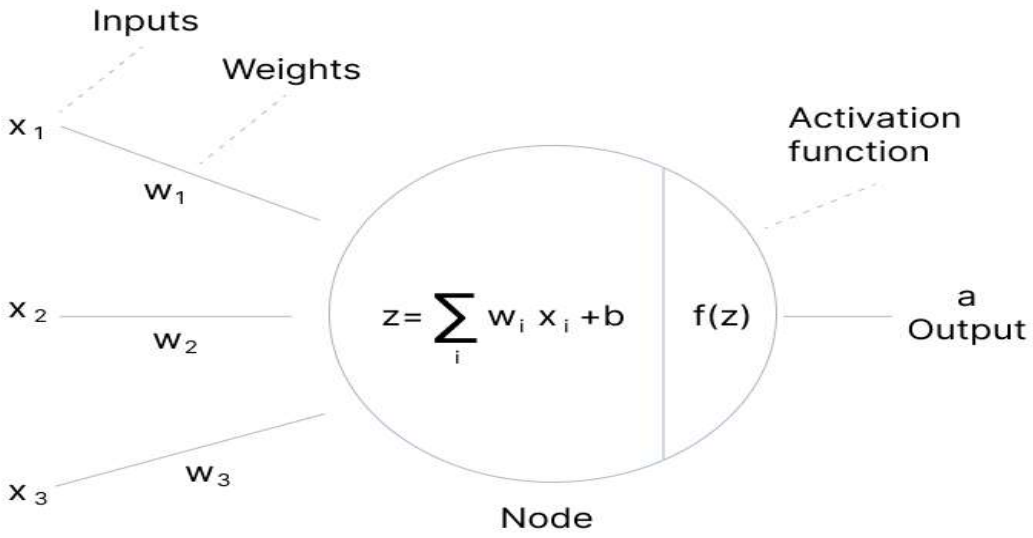
Typical Neural Network



https://www.javatpoint.com/artificial-neural-network

Activation Function

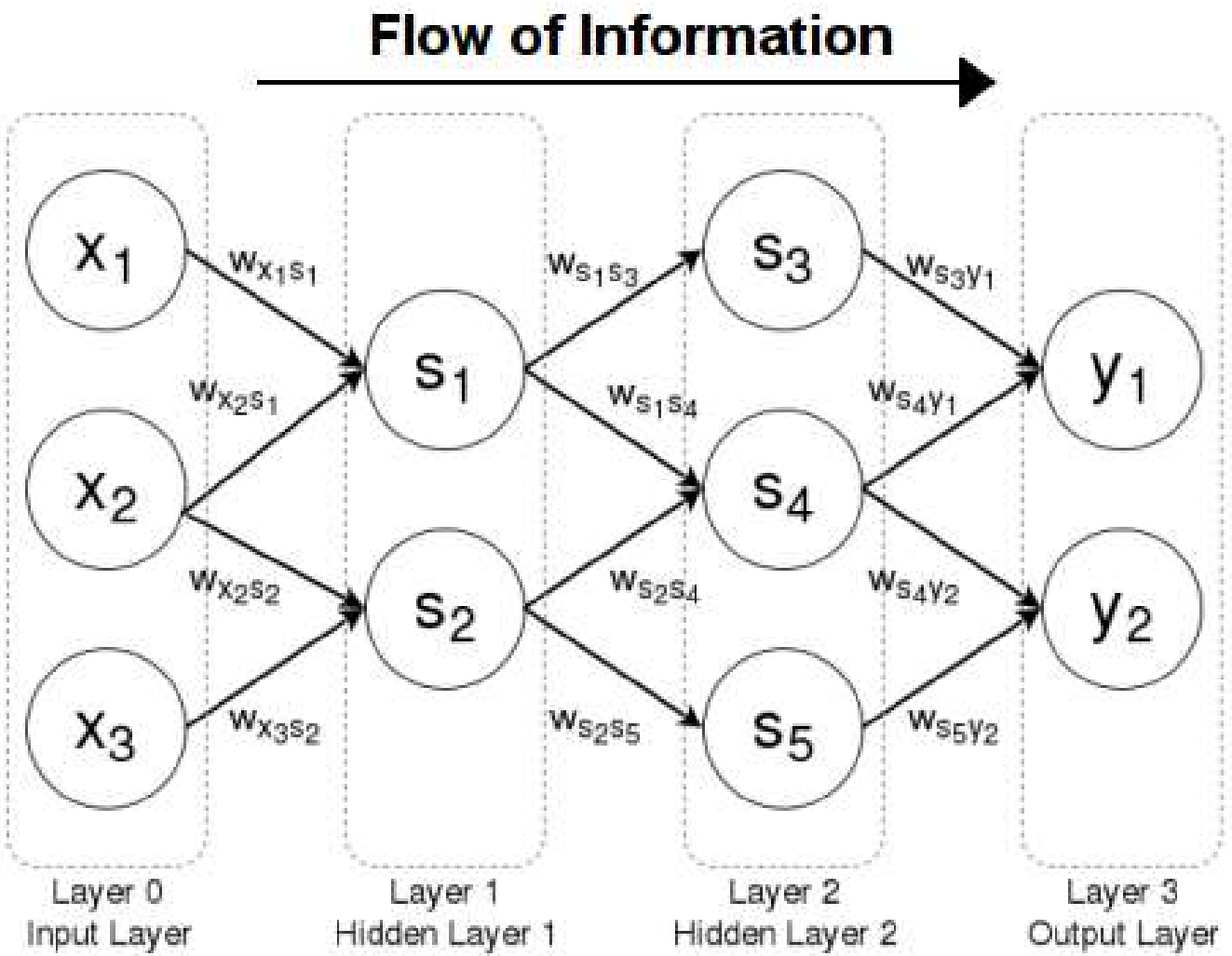
- An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.



V7 Labs

$$Y = \sum (weight * input) + bias$$

*https://www.v7labs.com/blog/neural-networks-activation-functions


Flow of Information
x1
x2
x3
s1
s2
s3
s4
s5
y1
y2
wx1s1
wx2s1
wx2s2
wx3s2
ws1s3
ws1s4
ws2s4
ws2s5
ws3y1
ws4y1
ws4y2
ws5y2
Layer 0
Input Layer
Layer 1
Hidden Layer 1
Layer 2
Hidden Layer 2
Layer 3
Output Layer

A Feedforward Neural Network's Layers

- **Input Layer**
  - It contains the neurons that receive input. The data is subsequently passed on to the next tier. The input layer's total number of neurons is equal to the number of variables in the dataset.
- **Hidden layer**
  - This is the intermediate layer, which is concealed between the input and output layers. This layer has a large number of neurons that perform alterations on the inputs. They then communicate with the output layer.
- **Output layer**
  - It is the last layer and is depending on the model's construction. Additionally, the output layer is the expected feature, as you are aware of the desired outcome.

Cost Function in Feedforward Neural Network

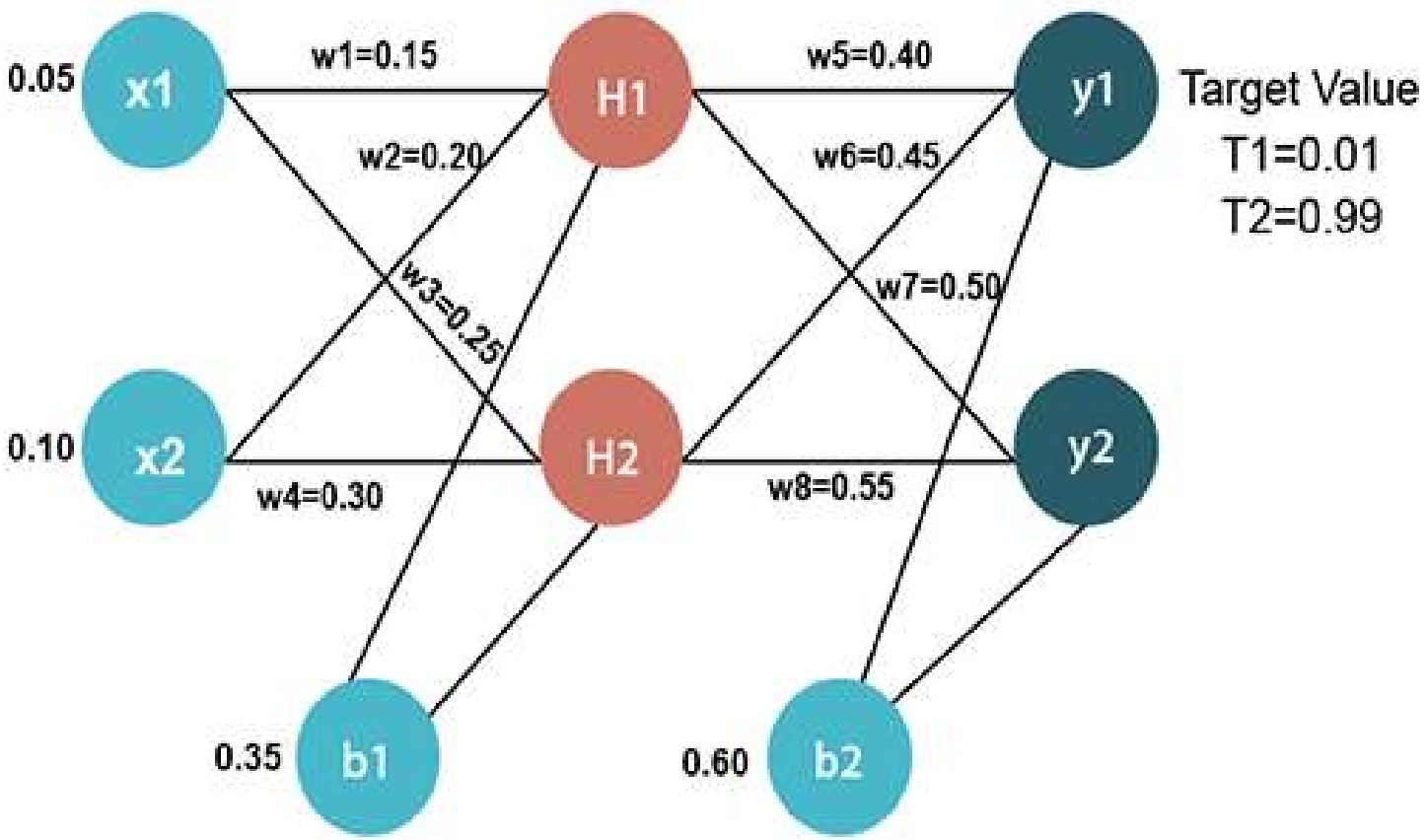
- The cost function is an important factor of a feedforward neural network. Generally, minor adjustments to weights and biases have little effect on the categorized data points. Thus, to determine a method for improving performance by making minor adjustments to weights and biases using a smooth cost function.

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2.$$

w = weights collected in the network
b = biases
n = number of training inputs
a = output vectors

Backpropagation Process in Deep Neural Network

- **Backpropagation** is one of the important concepts of a neural network. Our task is to classify our data best. For this, we have to update the weights of parameter and bias, but how can we do that in a deep neural network? In the linear regression model, we use gradient descent to optimize the parameter. Similarly here we also use gradient descent algorithm using Backpropagation.



https://www.javatpoint.com/pytorch-backpropagation-process-in-deep-neural-network

Forward Pass

To find the value of H1 we first multiply the input value from the weights as

$$H1=x1\times w_1+x2\times w_2+b1$$

$$H1=0.05\times 0.15+0.10\times 0.20+0.35$$

$$\mathbf{H1=0.3775}$$

$$H1_{final}=\frac{1}{1+\frac{1}{e^{H1}}}$$

$$H1_{final}=\frac{1}{1+\frac{1}{e^{0.3775}}}$$

$$\mathbf{H1_{final}=0.593269992}$$

We will calculate the value of H2 in the same way as H1

$$H2=x1\times w_3+x2\times w_4+b1$$

$$H2=0.05\times 0.25+0.10\times 0.30+0.35$$

H2=0.3925

$$H2_{final}=\frac{1}{1+\frac{1}{e^{H2}}}$$

$$H2_{final}=\frac{1}{1+\frac{1}{e^{0.3925}}}$$

$$\mathbf{H2_{final}=0.596884378}$$

Now, we calculate the values of y1 and y2 in the same way as we calculate the H1 and H2.

y1=H1×w_5+H2×w_6+b2

y1=0.593269992×0.40+0.596884378×0.45+0.60

y1=1.10590597

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{y1}}}$$

$$y1_{final} = \frac{1}{1 + \frac{1}{e^{1.10590597}}}$$

$$\mathbf{y1_{final} = 0.75136507}$$

y2=H1×w_7+H2×w_8+b2

y2=0.593269992×0.50+0.596884378×0.55+0.60

y2=1.2249214

$$y2_{final} = \frac{1}{1 + \frac{1}{e^{y2}}}$$

$$y2_{final} = \frac{1}{1 + \frac{1}{e^{1.2249214}}}$$

$$\mathbf{y2_{final} = 0.772928465}$$

Our target values are 0.01 and 0.99. Our y1 and y2 value is not matched with our target values T1 and T2.
Now, we will find the **total error**, which is simply the difference between the outputs from the target outputs.
The total error is calculated as

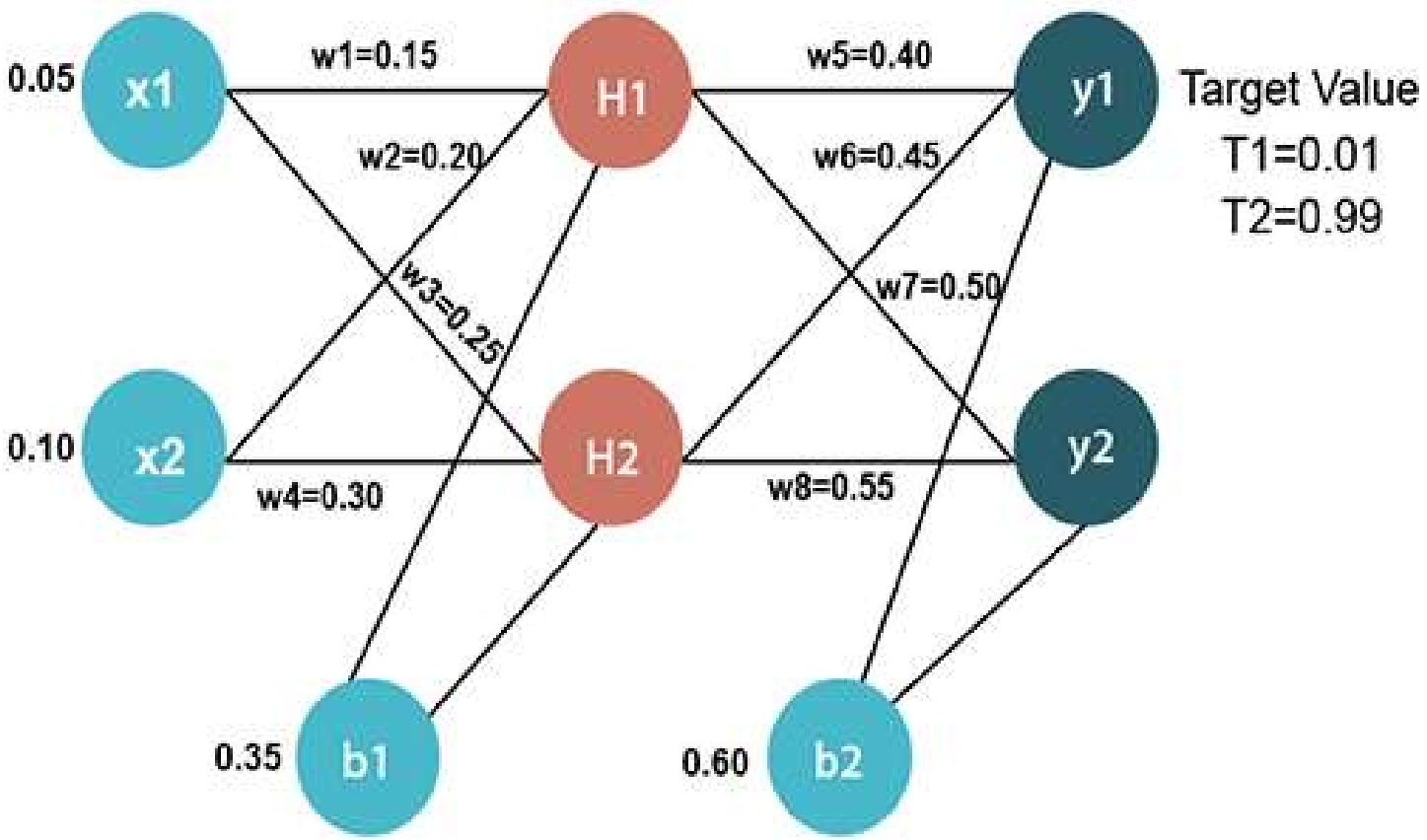
$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$= \frac{1}{2}(t1 - y1_{final})^2 + \frac{1}{2}(T2 - y2_{final})^2$$

$$= \frac{1}{2}(0.01 - 0.75136507)^2 + \frac{1}{2}(0.99 - 0.772928465)^2$$

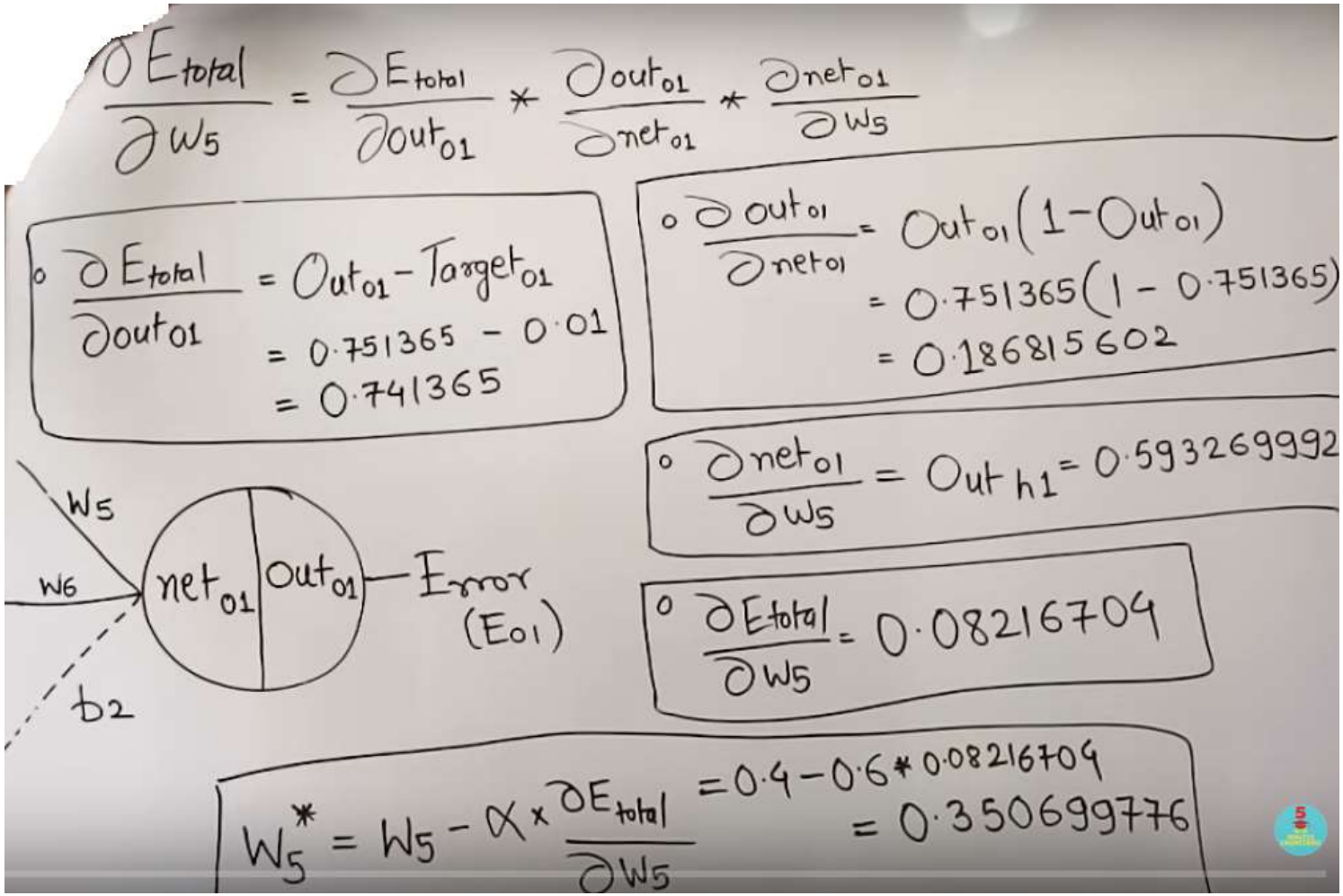
$$= 0.274811084 + 0.0235600257$$

$$\mathbf{E_{total} = 0.29837111}$$



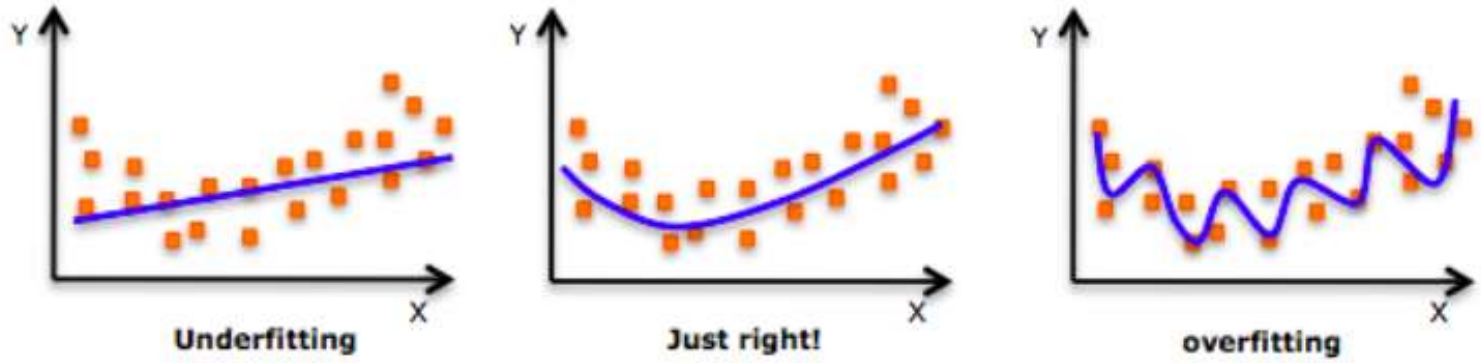
https://www.javatpoint.com/pytorch-backpropagation-process-in-deep-neural-network

Backpropagation



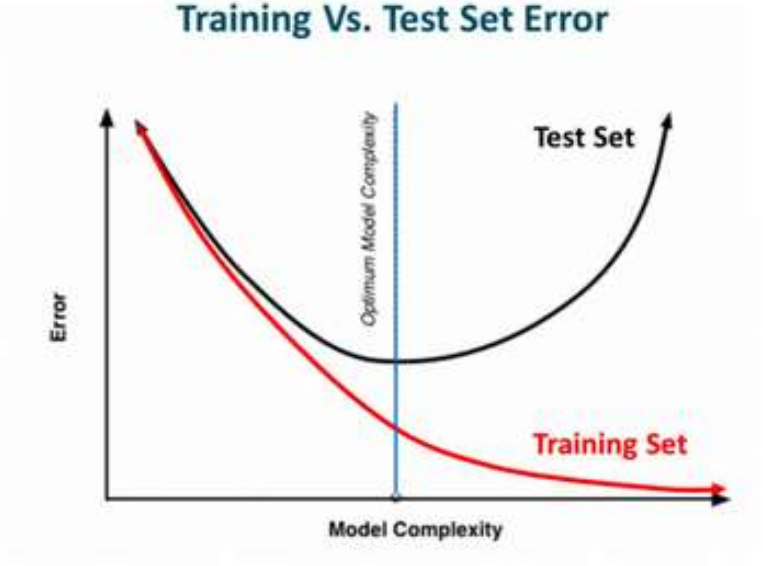
Overview of Regularization Techniques in Deep Learning

- One of the most common problems data science professionals face is to avoid **overfitting**. Have you come across a situation where your model performed exceptionally well on train data but was not able to predict test data.
- Avoiding overfitting can single-handedly improve our model's performance.



What is Regularization?

- Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.
- while going towards the right, the complexity of the model increases such that the training error reduces but the testing error doesn't. This is shown in the image below.



Regularization Techniques in Deep Learning.

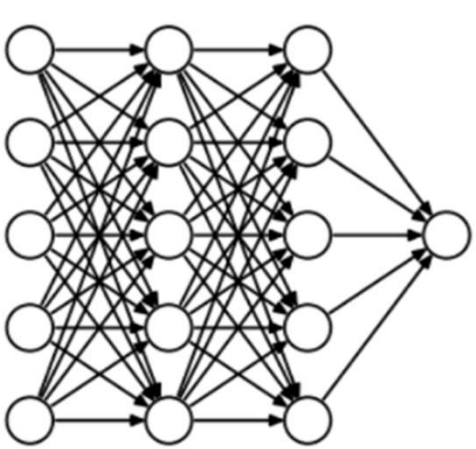
- **L2 & L1 regularization**
  - L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

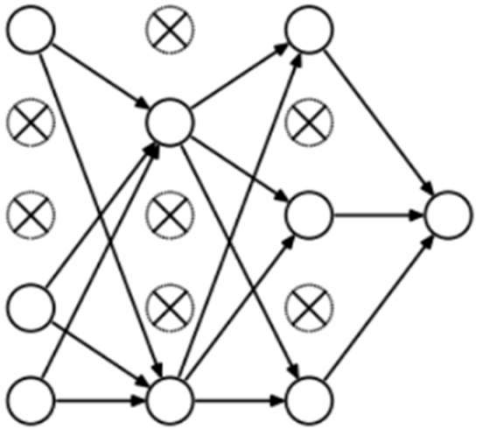
    Cost function = Loss (say, binary cross entropy) + Regularization term

- Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

Dropout Technique

- This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.
- At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.
- This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function.





Data Augmentation.

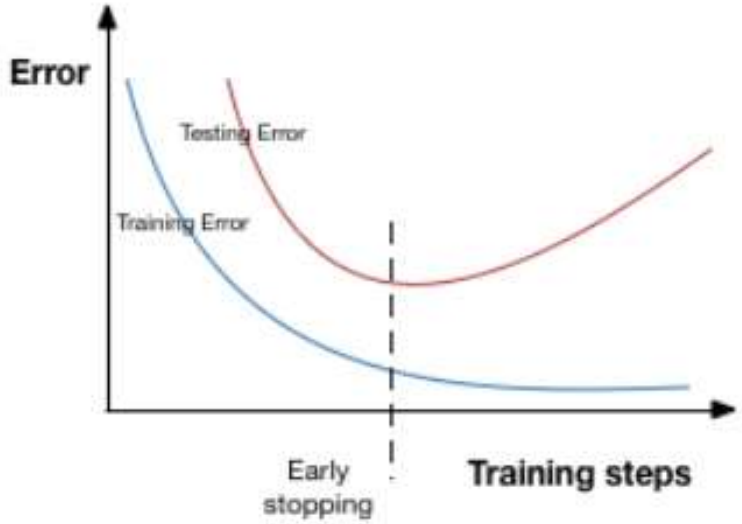
- The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, we were not able to increase the size of training data as the labeled data was too costly.
- But, now let's consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc. In the below image, some transformation has been done on the handwritten digits dataset.
- Data augmentation is the process of increasing the amount and diversity of data. We do not collect new data, rather we transform the already present data.

Data Augmentation.

- **Need for data augmentation**
  Data augmentation is an integral process in deep learning, as in deep learning we need large amounts of data and in some cases it is not feasible to collect thousands or millions of images, so data augmentation comes to the rescue.
- **Operations in data augmentation**
  1. Rotation
  2. Shearing
  3. Zooming
  4. Cropping
  5. Flipping
  6. Changing the brightness level

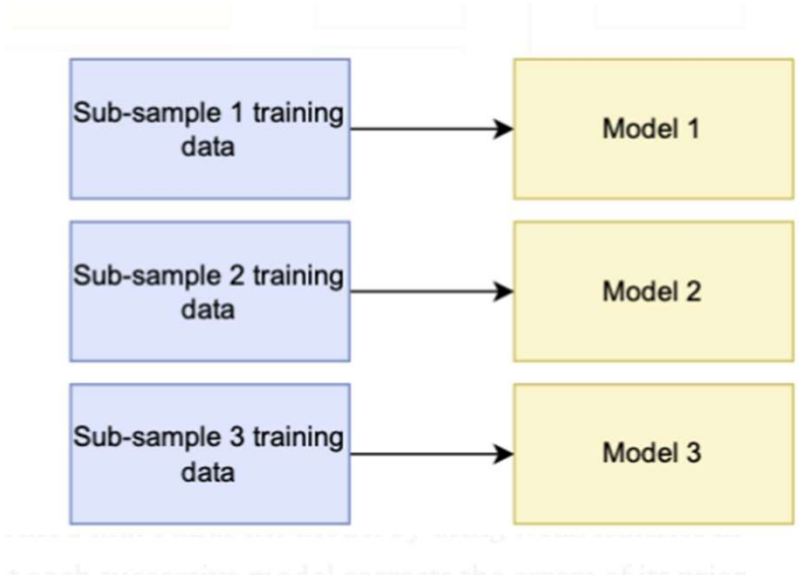
Early stopping

- Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.



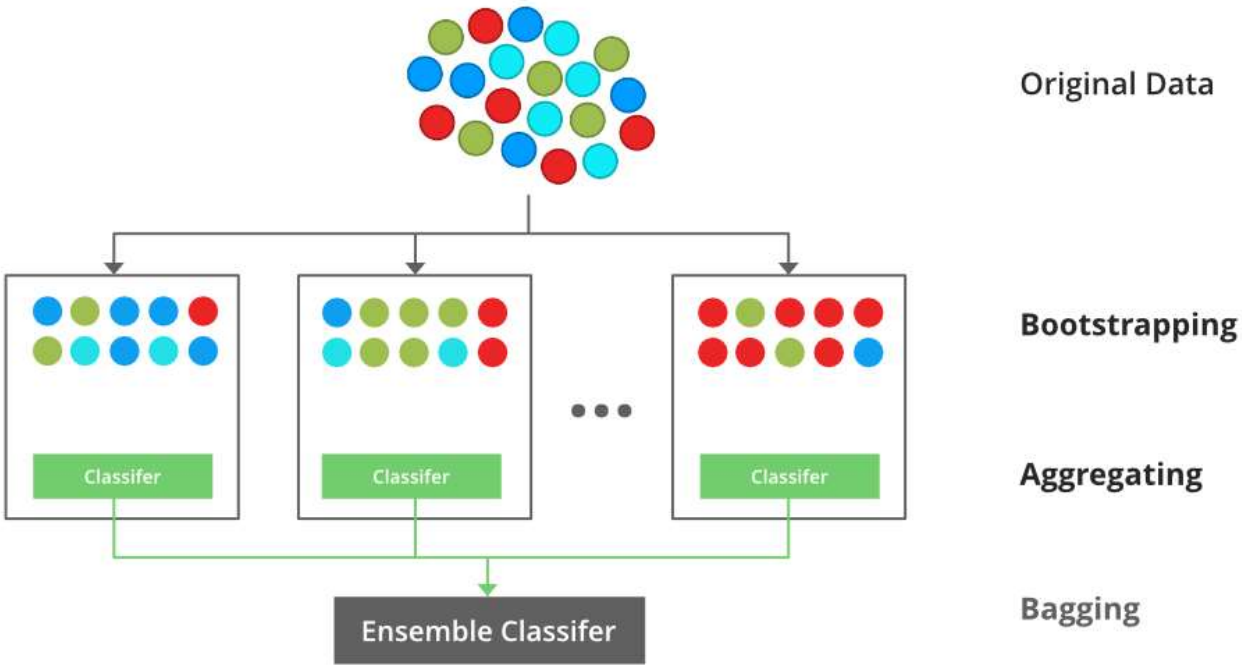
Bagging

- **B**ootstrap **A**ggregating, also known as bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It decreases the variance and helps to avoid overfitting. It is usually applied to decision tree methods. Bagging is a special case of the model averaging approach.
- Bagging is the practice of subsampling training data to improve a single type of classifier's generalisation performance. This method works well with models that tend to overfit the data.



Implementation Steps of Bagging

- **Step 1:** Multiple subsets are created from the original data set with equal tuples, selecting observations with replacement.
- **Step 2:** A base model is created on each of these subsets.
- **Step 3:** Each model is learned in parallel with each training set and independent of each other.
- **Step 4:** The final predictions are determined by combining the predictions from all the models.



Example of Bagging

The Random Forest model uses Bagging, where decision tree models with higher variance are present. It makes random feature selection to grow trees. Several random trees make a Random Forest.

https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/

Adversarial Machine Learning

What is Adversarial Machine Learning

- Adversarial machine learning is a machine learning method that aims to trick machine learning models by providing deceptive input. Hence, it includes both the generation and detection of adversarial examples, which are inputs specially created to deceive classifiers.
- Such attacks, called adversarial machine learning, have been extensively explored in some areas, such as image classification and spam detection.
- The most extensive studies of adversarial machine learning have been conducted in the area of image recognition, where modifications are performed on images that cause a classifier to produce incorrect predictions

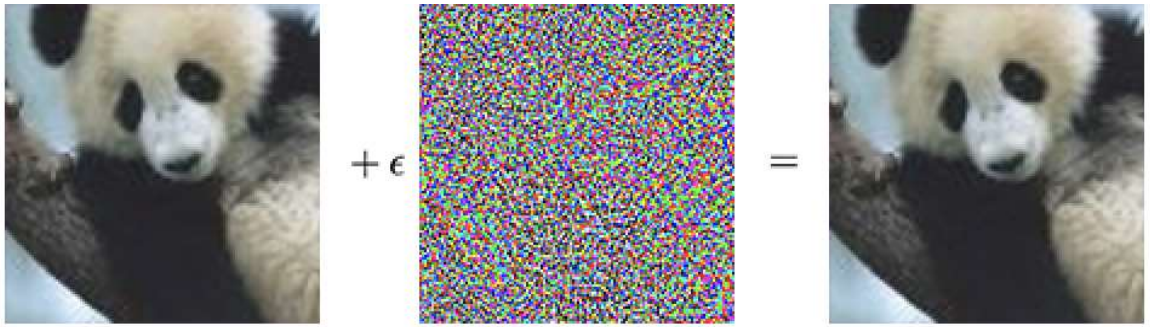
https://viso.ai/deep-learning/adversarial-machine-learning/

Adversarial Machine Learning

- Most adversarial attacks usually aim to deteriorate the performance of classifiers on specific tasks, essentially to “fool” the machine learning algorithm. Adversarial machine learning is the field that studies a class of attacks that aims to deteriorate the performance of classifiers on specific tasks.
- *Adversarial Machine Learning is a **collection of techniques to train neural networks on how to spot intentionally misleading data or behaviors**. This differs from the standard classification problem in machine learning, since the goal is not just to **spot “bad” inputs**, but preemptively locate vulnerabilities and craft more flexible learning algorithms*.

Adversarial Machine Learning Example

- As an example, let us take a **GoogLeNet trained on ImageNet** to perform image classification as our machine learning model. Below you have two images of a panda that are indistinguishable to the human's eye. The image on the left is one of the clean images in ImageNet dataset, used to train the GoogLeNet model. The one on the right is a slight modification of the first, created by adding the noise vector in the central image. The first image is predicted by the model to be a panda, as expected. The second, instead, is predicted (with very high confidence) to be a gibbon.



"panda"
57.7% confidence

"gibon"
99.3% confidence

Source: https://openai.com/blog/adversarial-example-research/

Adversarial Machine Learning Example

- In another case, researchers at Samsung and Universities of Washington, Michigan and UC Berkley showed that by making small tweaks to stop signs, they could make them invisible to the computer vision algorithms of self-driving cars. A hacker might use this adversarial attack to force a self-driving car to behave in dangerous ways and possibly cause an accident.

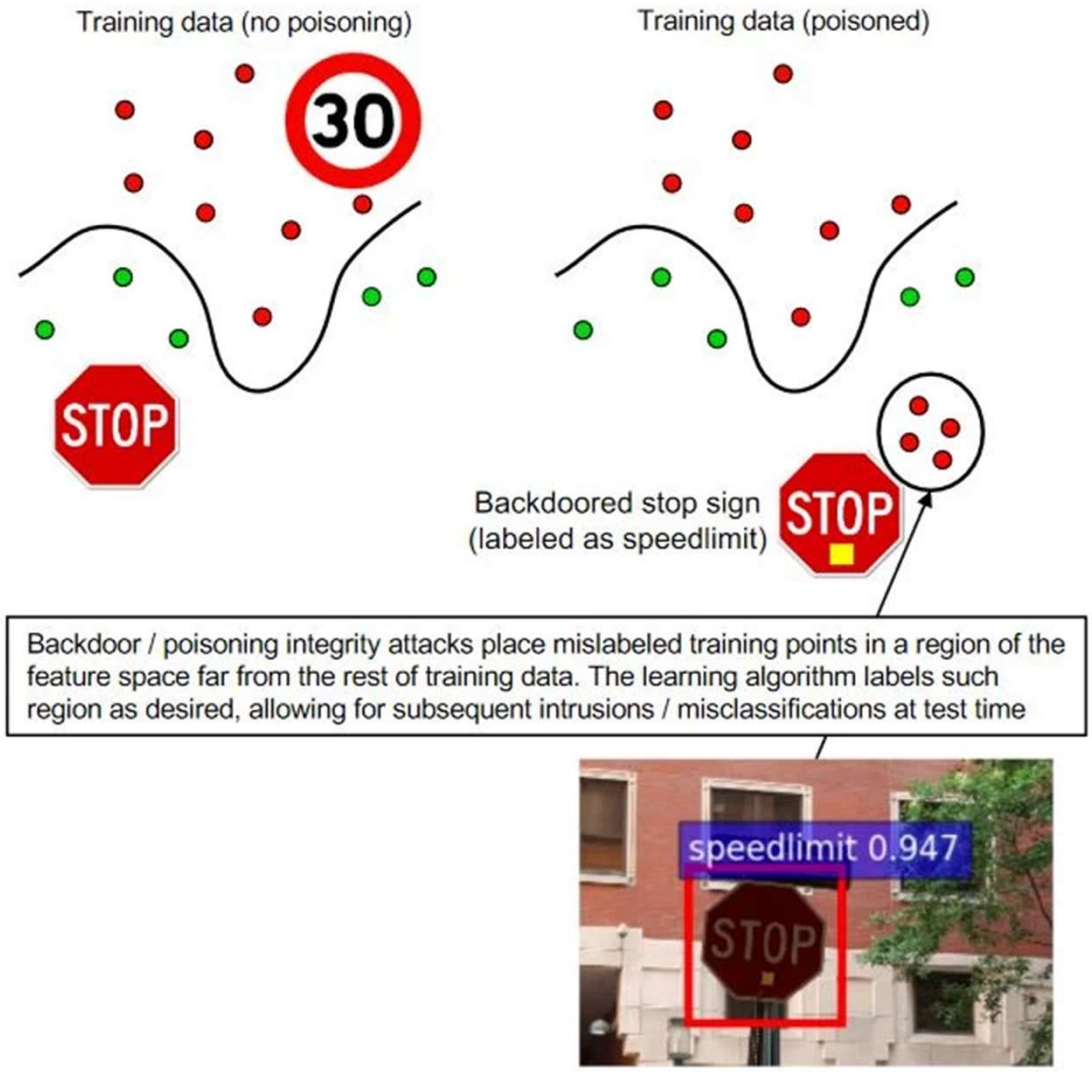
https://bdtechtalks.com/2020/07/15/machine-learning-adversarial-examples/

Adversarial Machine Learning Example

- Adversarial examples do not just apply to neural networks that process visual data. There is also research on adversarial machine learning on text and audio data.
- In 2018, researchers at UC Berkley managed to manipulate the behavior of an automated speech recognition system (ASR) with adversarial examples. Smart assistants such as Amazon Alexa, Apple Siri, and Microsoft Cortana use ASR to parse voice commands.
- For instance, a song posted on YouTube can be modified in a way that playing it would send a voice command to a smart speaker nearby. A human listener wouldn’t notice the change. But the smart assistant’s machine learning algorithm would pick up that hidden command and execute it.

Poisoning Attacks

- The attacker influences the training data or its labels to cause the model to underperform during deployment. Hence, Poisoning is essentially adversarial contamination of training data. As ML systems can be re-trained using data collected during operation, an attacker may poison the data by injecting malicious samples during operation, which subsequently disrupt or influence re-training.



Adversarial Machine Learning Defences

1. **Adversarial training**– This is a brute force supervised learning method where **as many adversarial examples as possible are fed into the model** and explicitly labeled as threatening. This is the same approach the typical antivirus software used on personal computers employs, with multiple updates every day. While quite effective, it requires continuous maintenance to stay abreast of new threats and also still suffers from the fundamental problem that it can only stop something that has already happened from occurring again.
2. **Defensive distillation**– This strategy adds flexibility to an algorithm's classification process so the model is less susceptible to exploitation. In distillation training, **one model is trained to predict the output probabilities of another model** that was trained on an earlier, baseline standard to emphasize accuracy.