Deep Neural Network Training: Diagnosing Problems and Implementing Solutions

Fahed Hassanat

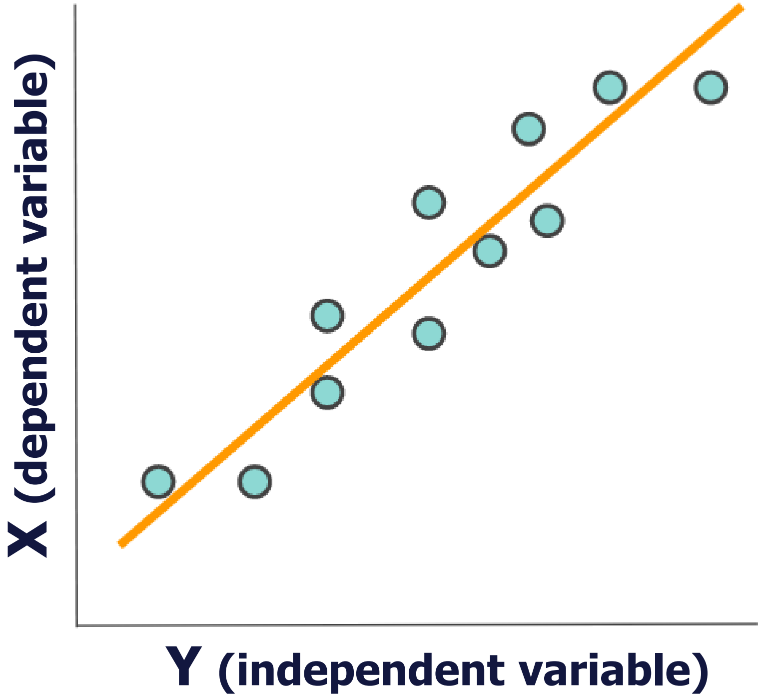
COO / Head of Engineering

Sensor Cortek

Training DNNs

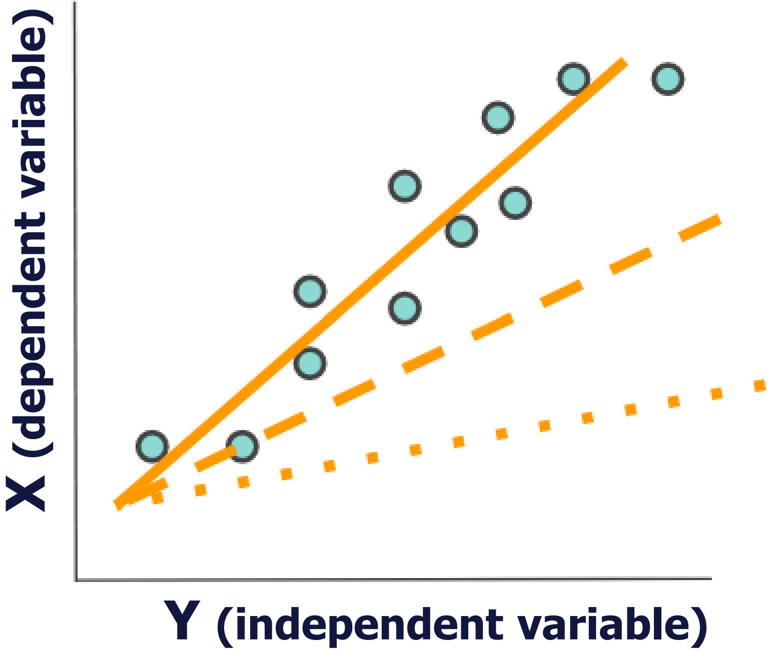
The Goal

- Find an acceptable relationship between inputs and outputs based on patterns found in historical data.



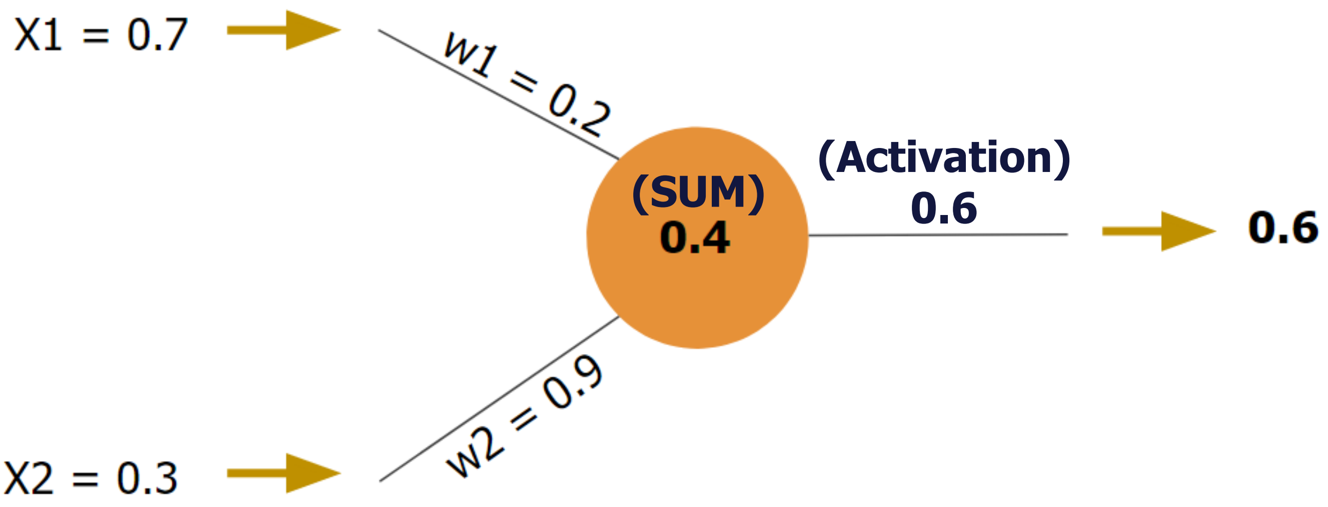
The Learning Process

- The process of minimizing the difference between the produced output and the actual output.
- Uses mathematical techniques to minimize the error.
- Stop when the results are acceptable or can no longer learn.



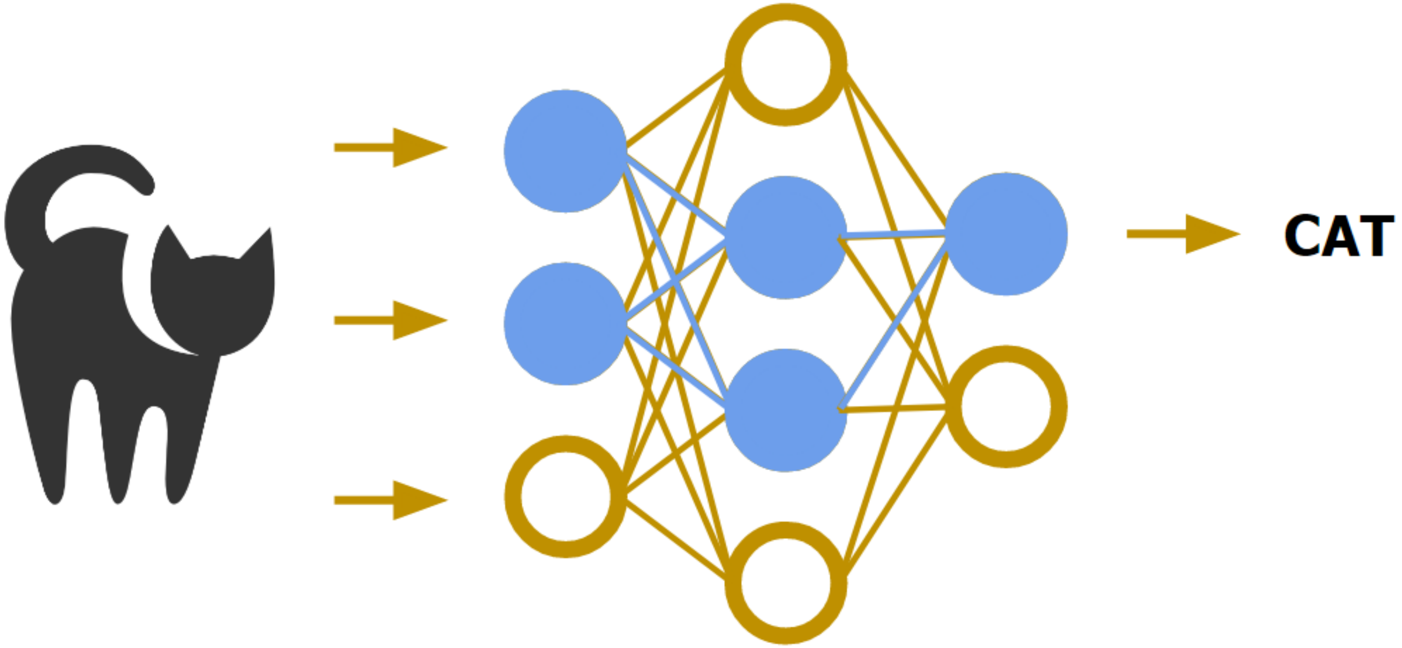
Example Forward Calculation

SUM = (0.7x0.2)+(0.3x0.9)



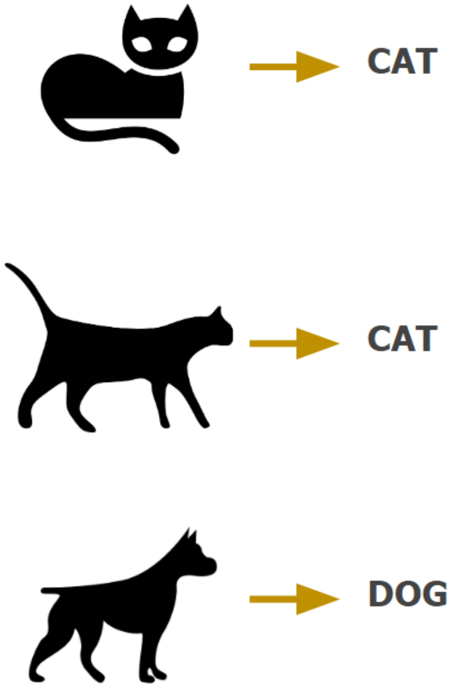
Activation (Sigmoid: $1/(1+e^{-x})$) = $1/(1+e^{-0.4})$

An Educated Network



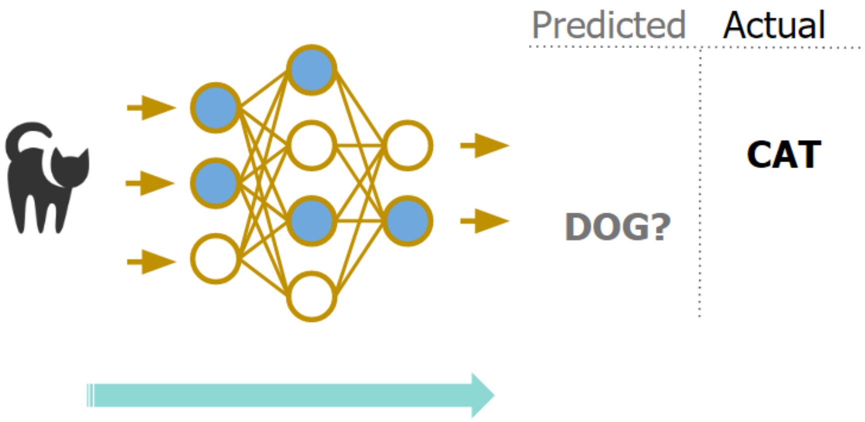
Training Data

- A labeled collection of data.
- Variations of inputs that produce same output.
- The larger and more diverse the better.



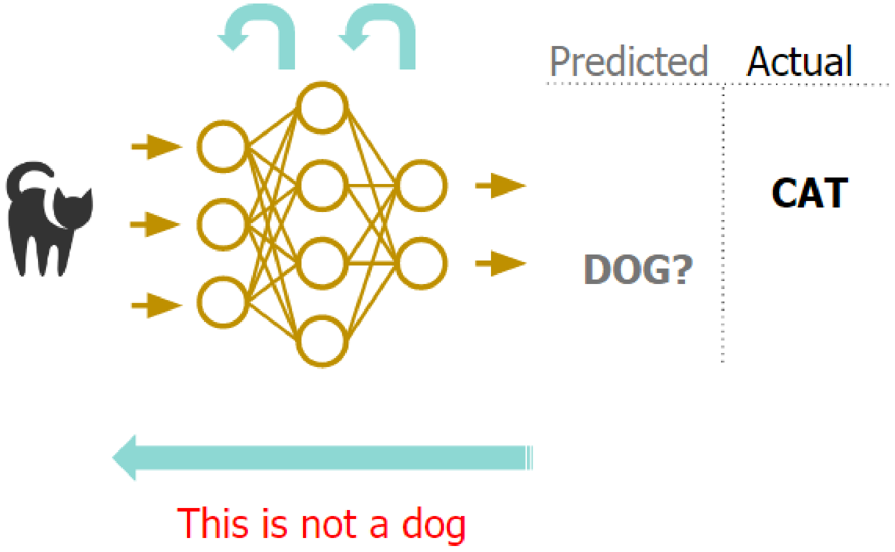
The Forward Pass

- A pass through the network in forward direction.
- Use training data.
- Produce results with the current parameters.



The Backward Pass

- Traverse all nodes starting from the last layer.
- Update the parameters (weights) to minimize the error.
- Gradient Descent is usually the technique used to work towards a minimum error.



Components of Training DNNs

Components of Training DNNs

- **Framework**: Manages the data flow and execution of the training process.
- **DNN model**: An architecture that serves a certain purpose.
- **Hyperparameters**: Controls the training process.
- **Training data**: Labeled dataset for training the model.

Framework

- **Keras**:
  - Great for prototyping. High API level and high readability.
- **TensorFlow:**
  - High performance. Suited for large datasets.
- **Pytorch**:
  - High performance. Excellent debugging.

DNN Model

Architecture	Purpose
Multi-layer perceptron	Image classification, natural language processing, and regression
Convolutional neural networks (CNN)	Image classification, object detection, and segmentation
Recurrent neural networks (RNN)	Time series prediction, and speech recognition
Transformer networks	Natural language processing, and computer vision
Generative adversarial networks (GAN)	Synthetic data generation

What is a Hyperparameter

- Hyperparameters are parameters that cannot be learned during the training process and are set prior to the training process.

- Some components of the model architecture can be considered hyperparameters.

- Model designers can create a hyperparameter that controls multiple hyperparameters in a certain fashion.

Hyperparameters

Non-architecture-based	Architecture-based
Learning rate	Number of layers
Number of epochs	Number of neurons
Batch size	Activation function
Dropout rate	
Weight initialization	
Regularization parameters	
Optimizer	

Hyperparameters: Learning Rate

- A value that controls the amount by which the network weights are updated.
- Usually between 0.0 and 1.0.
- Too large or too small affects the convergence of the model.

Hyperparameters: Number of Epochs

- The number of times the dataset is passed through the network.
- Too small and the model does not converge.
- Too large and the model overfits.

Hyperparameters: Batch Size

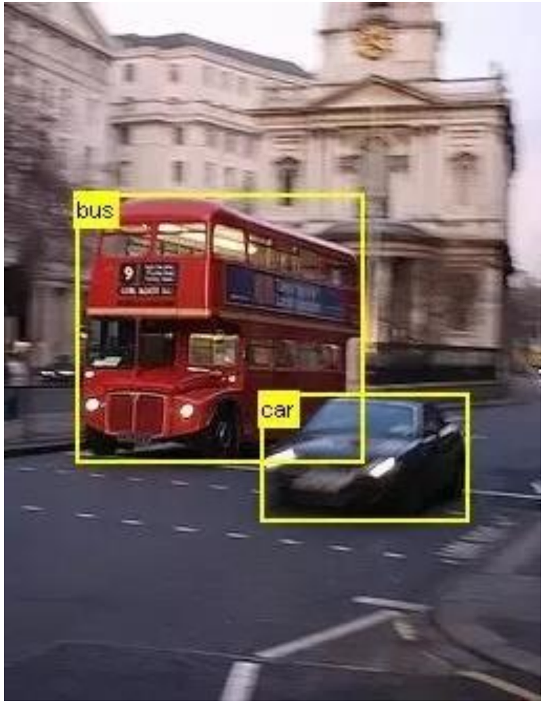
- The number of data samples used in each iteration of the optimization algorithm.

- Too large causes the model not to generalize well.

- Too small can prevent the model from converging.

Dataset

- Collection of data with corresponding labels.
- Data is used as input to the model and labels adjust and correct the output.
- Dataset is divided into training/validation/testing sets with the commonly used ratios of 60/20/20 respectively.



Attributes of a Good Dataset

Sufficient data	The dataset must contain enough data to reflect the targeted population.
Balanced data	In multiclass problems, classes must have balanced contributions to the dataset.
Relevant data	The data must represent the targeted population and population environment.
Proper labeling	Labels must be accurate, and consistent.
Data diversity	The diversity of the data should reflect the diversity of the targeted population.
Low SNR	The data should have little to no noise that cause ambiguity.

Training Metrics

Prediction Outcomes

True Positive (TP)	The model predicts **True**, and the label's **True.**
False Positive (FP)	The model predicts **True,** and the label's **False.**
False Negative (FN)	The model predicts **False,** and the label's **True.**

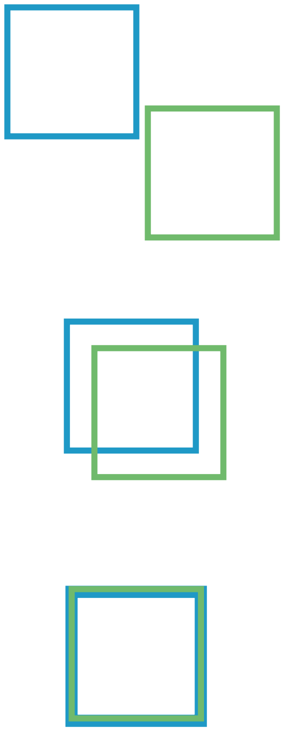
Intersection over Union

- Used in object detection.
- Calculates overlap in bounding boxes.
- Determines how close is a prediction to the ground truth.
- Ranges from 0 to 1.
- A typical value for qualifying a prediction as **TP** is 0.5.

IoU = 0

IoU = 0.5

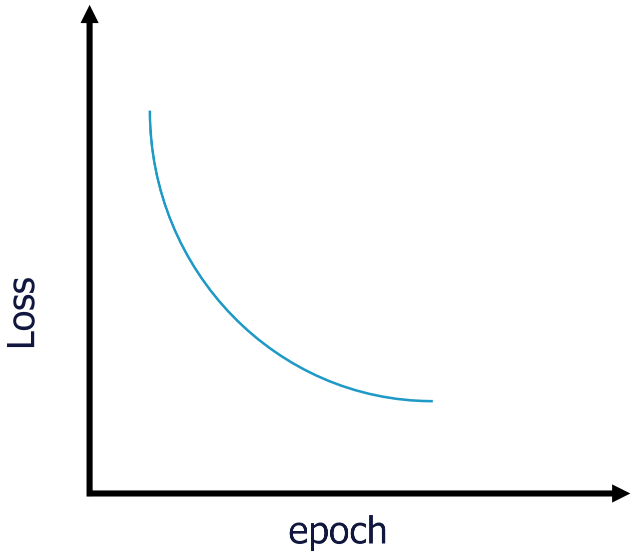
IoU = 1



Loss

- Is a measure of how far the predictions are from the actual values.

- It is the output of the loss function during training.



Accuracy

- It is the number of good predictions out of all predictions.

- Can be calculated as:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

Precision

- The number of correctly predicted labels out of all True predictions.

- Can be calculated as:

$$\frac{TP}{TP+FP}$$

Recall

- The number of correctly predicted labels out of all True labels in the data.

- Can be calculated as:

$$\frac{TP}{TP+FN}$$

F1-Score

- It is the harmonic mean of **Precision** and **Recall**.
- Gives a global picture of the performance.
- Can be calculated as:

$$2 \text{ x } \frac{Precision \text{ x } Recall}{Precision + Recall}$$

Confusion Matrix

- A window into the performance of the model on one or more classes.
- Can be used to calculate Accuracy, Precision and Recall.

		Actual Values	
		P	N
Predicted Values	N	**TP**	**FP**
	P	**FN**	**TN**

Example: Multi-class Confusion Matrix

Actual Values

Predicted Values

	A	B	C	D
A	**93**	1	5	13
B	4	**89**	5	3
C	1	7	**88**	5
D	2	1	2	**79**

Problems with Training DNNs

Common Problems

- Overfitting
- Underfitting
- Dataset imbalance
- Inefficient learning
- Parameter initialization
- Gradient masking
- Vanishing gradient
- Exploding gradient

Overfitting

Problem	Solution
When a network learns the training data too well, it can fail to generalize to new data	**Larger dataset**: either add more labeled data or use data augmentation to artificially increase the size and variation of the dataset
	Less complex model: remove layers or reduce the width of the layers
	Early stoppage / Saving checkpoints
	Apply dropout
	Apply regularization

Underfitting

Problem	Solution
When a network is unable to capture the complexity of the data, it can fail to fit the training data well enough	**Examine the dataset**: bad or missed labeling, as well as lack of class representation can cause under fitting
	Increase the number of epochs: not training the model enough causes it to fail to grasp the essential pattern in the data
	More complex model: add layers, increase the width of the layers
	New architecture

Data Imbalance

Problem	Solution
If the dataset used to train the network is not representative of the target population, the network may fail to generalize well to new data	**Add more data**: either add more labeled data to fix the imbalance or use data augmentation to artificially achieve the same effect
	Transfer learning and fine-tuning: use pretrained model weights for training on the new data. The pretrained model would have been trained on a more balanced dataset and captured that balance within its weights

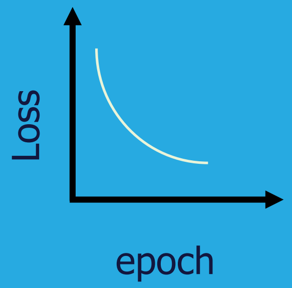
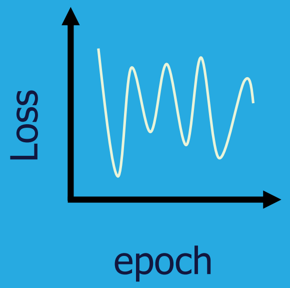
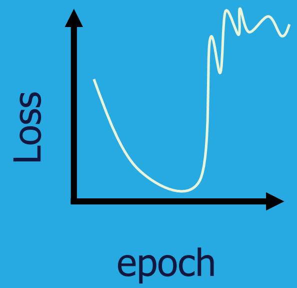
SENSOR CORTEK

Inefficient Learning

Problem	Solution
When the learning process becomes slow or stalls out during training, or when the model is unable to optimize the objective function	**Investigate learning rate:** a specific learning rate can cause the training to be stuck at a local minima
	Use momentum: accelerates convergence of the model
	New architecture

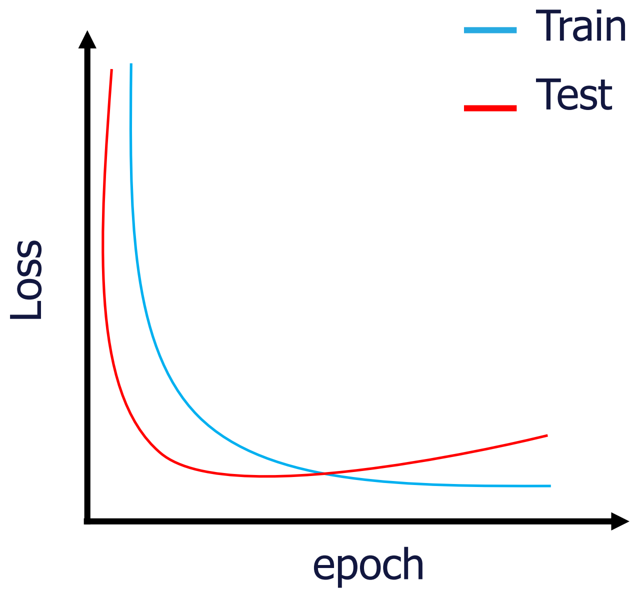
Detecting Training Problems

Loss Curve

		
Ideal	• Bad data • Model too simple • Large learning rate	• Bad data • Check activation functions

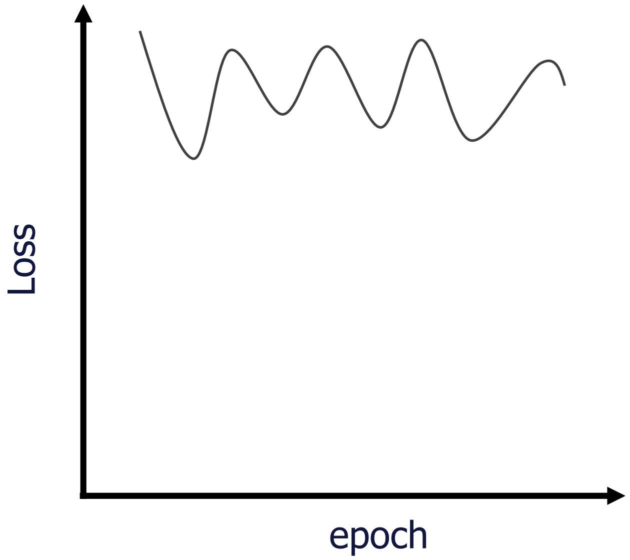
Overfitting Indicators

- The loss on the test set will start to diverge.
- Happens when the model memorizes the training data.
- Happens when the epoch number is too large.
- Happens when the dataset is too small.



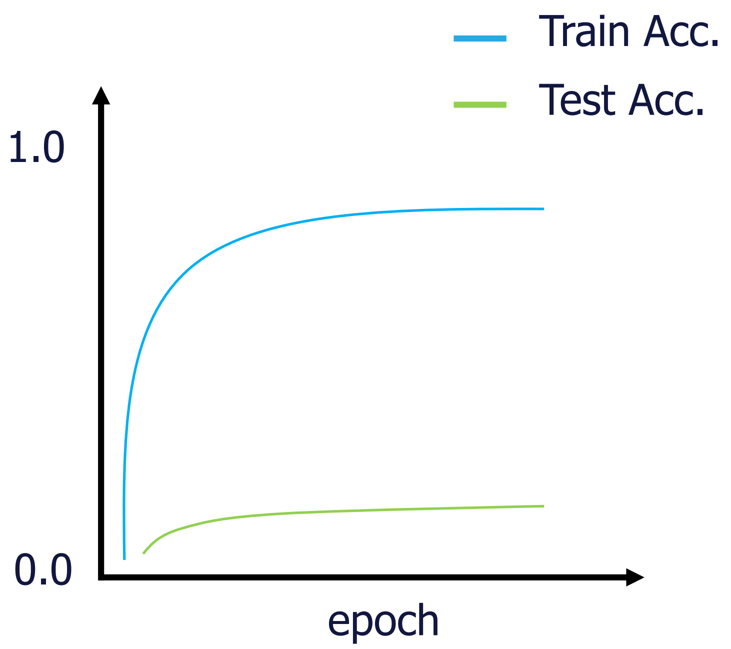
Underfitting Indicators

- Indicated by high and noisy loss values.
- The model is not able to learn from the training data.
- Happens due to bad data, small dataset or a small/low complexity model.



Accuracy Indicators

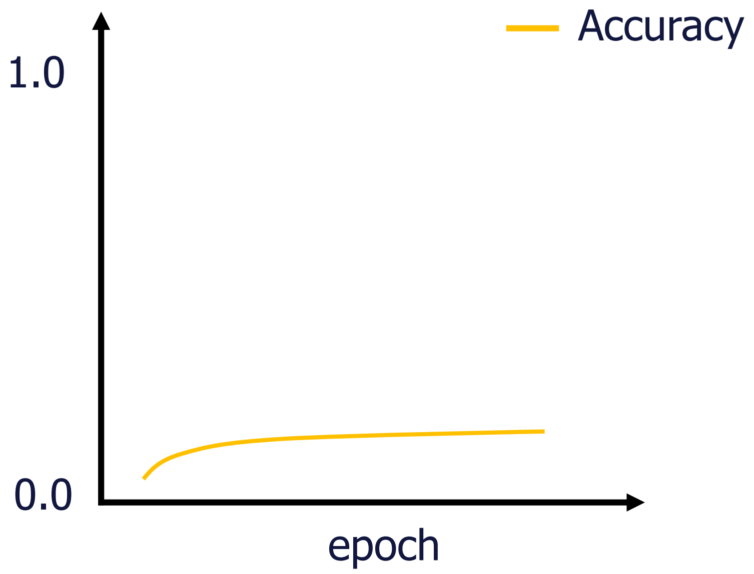
- **High train accuracy, low test accuracy**: this is a sign of overfitting the model.
- **Low train accuracy, high test accuracy**: this is a sign of imbalance between train and test data.



Low Accuracy

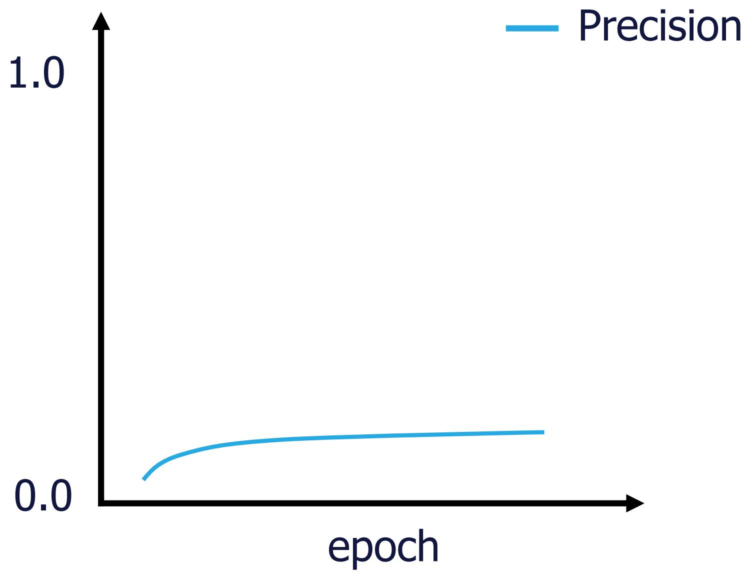
- Due to large number of incorrectly classified instances.

- Usually when the value is below 60%.



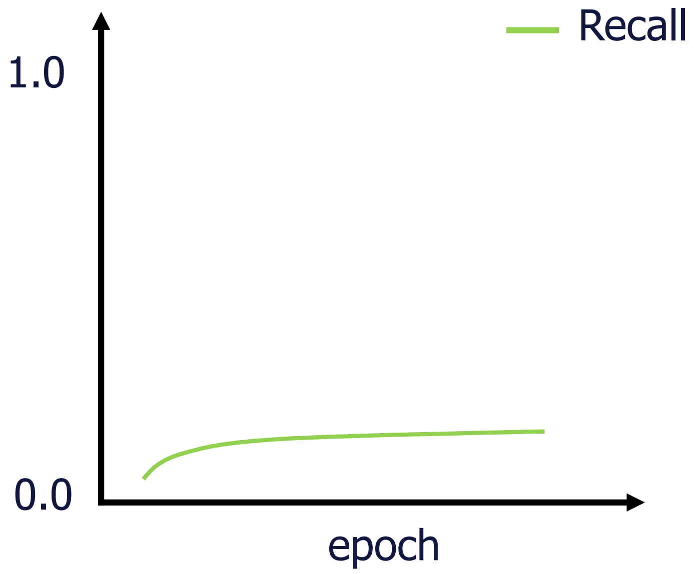
Low Precision

- Due to large number of false positives.
- The model misclassifies many negative instances as positive.
- Usually when the value is 50% or less.



Low Recall

- Due to many false negatives.
- The model is missing many positive instances.
- Recall is low when it is below 50%.



Fixing Low Accuracy/Precision/Recall

Possible cause	Remedy
Noisy or imbalanced data	Remove noise, augment data, use class weighting, use oversampling/undersampling to fix the imbalance of the data
Small dataset	Use pretrained models as they have already learned important features
Simple model	Increase the number of layers, width of the layer or change to a different architecture
Inadequate hyperparameters	Revisit the choice of hyperparameters and select appropriate parameter values
Inadequate loss function	Change to a loss function more suitable for the task

Confusion Matrix Indicators

Multi-class Confusion Matrix:

IDEAL

		Actual Values			
		A	B	C	D
Predicted Values	A	**100**	0	0	0
	B	0	**100**	0	0
	C	0	0	**100**	0
	D	0	0	0	**100**

Confusion Matrix Indicators

Multi-class Confusion Matrix:

Underfitting,

Inefficient learning,

Simple model …etc

Actual Values (columns) / Predicted Values (rows)

	A	B	C	D
A	23	29	**36**	24
B	18	17	15	**34**
C	**38**	**39**	30	26
D	21	15	19	16

Confusion Matrix Indicators

Multi-class Confusion Matrix:

Data imbalance,

Insufficient dataset

		Actual Values			
		A	B	C	D
Predicted Values	A	**93**	1	5	13
	B	4	**89**	5	3
	C	1	7	**42**	**39**
	D	2	1	**46**	**40**

Conclusion

- Training DNNs is a complex process that is subject to many problems.
- With the proper understanding of the training process, one can efficiently resolve these problems.
- Datasets, hyperparameters and model architecture are the major contributors to problems during training.
- Precision, recall, accuracy and confusion matrices are important training evaluation metrics that can help diagnose and guide towards successful training.

Resources

Interpreting Loss Curves
https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic

2023 Embedded Vision Summit

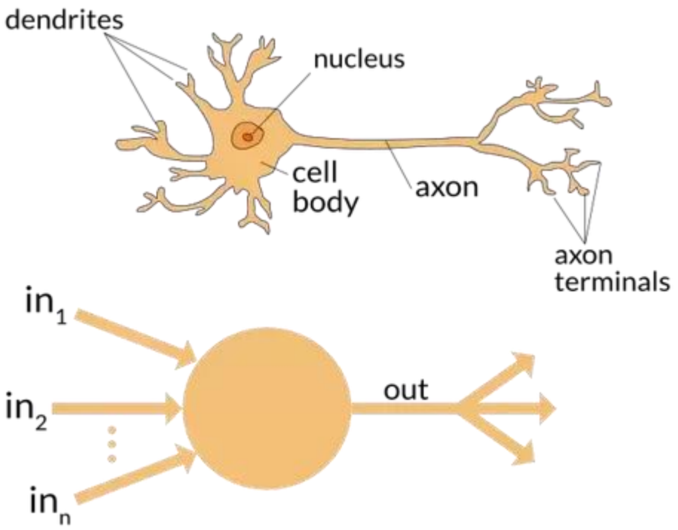
- "Fundamentals of Training AI Models for Computer Vision Applications" by Amit Mate (Tue. May 23rd, 1:30-2:35 pm)
- "Introduction to Modern LiDAR for Machine Perception" by Robert Laganiere (Wed. May 24th, 4:15-5:20 pm)

Backup Material

What is a Deep Neural Network?

Inspiration

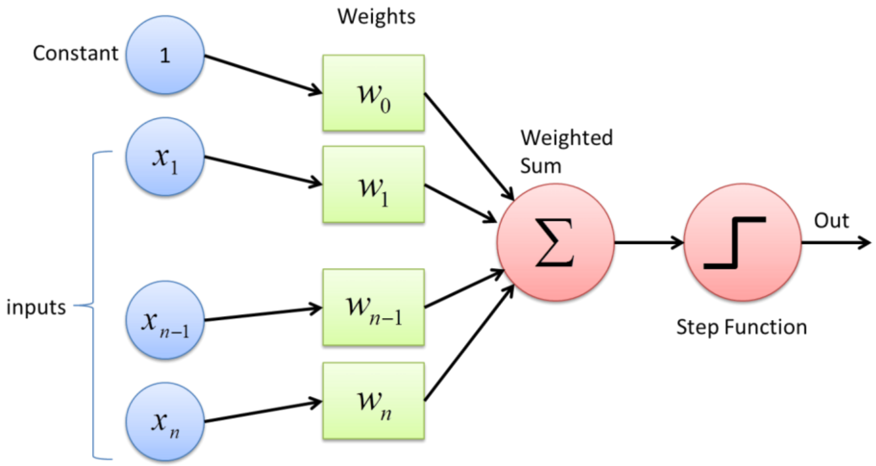
- Deep Neural networks are inspired by the function of biological neurons, the simplest unit of the nervous system in humans.
- “Deep” refers to the presence of multiple layers between the input and the output to the network.



Anatomy of the Perceptron

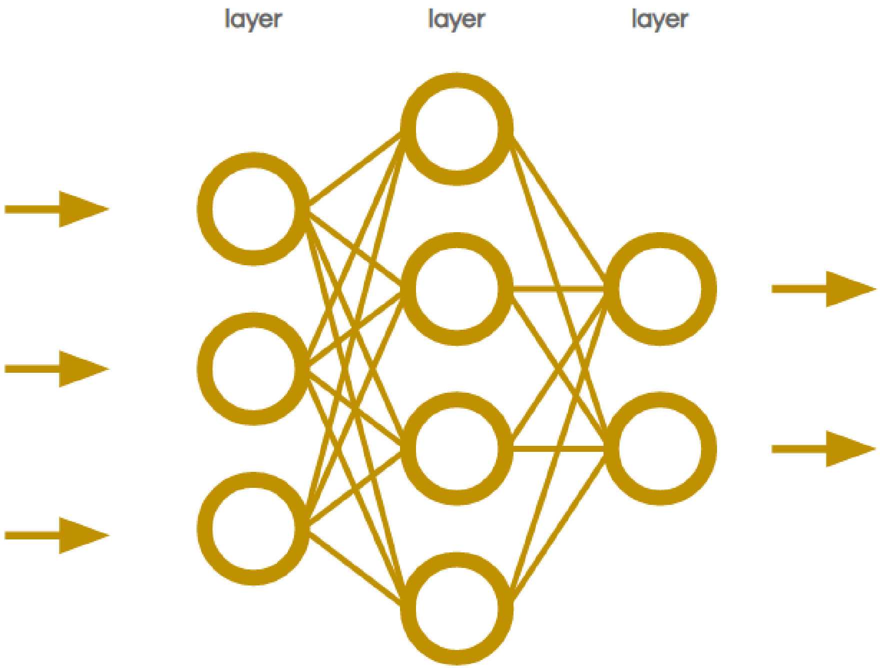
The basic unit of a NN is a perceptron:

- Inputs
- Weights
- Sum
- Activation
- Output



Building a Neural Network

- Input layer
- Hidden layer(s)
- Output layer



Additional Hyperparameters

Number of Layers (Model Architecture)

- Determines the depth of the network.
- Too large results in more parameters to train and longer training.
- Too small may not capture the complex relationship between the input and the output.

Number of Neurons (Model Architecture)

- Determines the width of the network.
- Too large results in more parameters to train and longer training.
- Too small (shallow) may not capture the complex relationship between the input and the output.

Activation Function

- Determines if the neuron is activated and should pass on the transformed input to the next layer.
- Impacts how well the model learns

Dropout Rate

- Controls the number of neurons that are randomly removed during training.
- Helps the model to generalize.
- Avoids overfitting.

Weight Initialization

- Controls how the weights are initialized prior to starting training.

Regularization Parameter

- Controls how strong the penalty term in the cost function of a model.
- Helps in controlling over/under fitting of the model.

Optimizer

- Controls the method used to update the weights.
- Helps minimize the loss and improve the accuracy.

Additional Training Problems

Gradient Masking

- **The Problem**

When gradients of some parameters vanish because of the choice of activation function, the parameters cannot be learned.

- **The Solution**

This can be addressed by using activation functions that alleviate the issue, such as leaky ReLU or Swish, or by using different initialization schemes.

Parameter Initialization

- **The Problem**

If the weights of the network are initialized randomly and are too small or too large, the network may fail to learn.

- **The Solution**

This can be addressed by using initialization techniques such as Xavier or He initialization, which ensure that the weights are initialized in a way that is appropriate for the network architecture and the activation functions used.

Vanishing Gradient

- **The Problem:**

The gradients in the deeper layers of the network become very small during backpropagation, making it difficult to update the weights of those layers.

- **The Solution**

use activation functions that don't saturate, such as ReLU or variants, and by using normalization techniques such as Batch Normalization.

Exploding Gradient

- **The Problem**

The gradients in the deeper layers of the network become very large during backpropagation, making it difficult to update the weights of those layers.

- **The Solution**

using gradient clipping techniques, which limit the magnitude of the gradients.