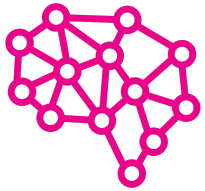
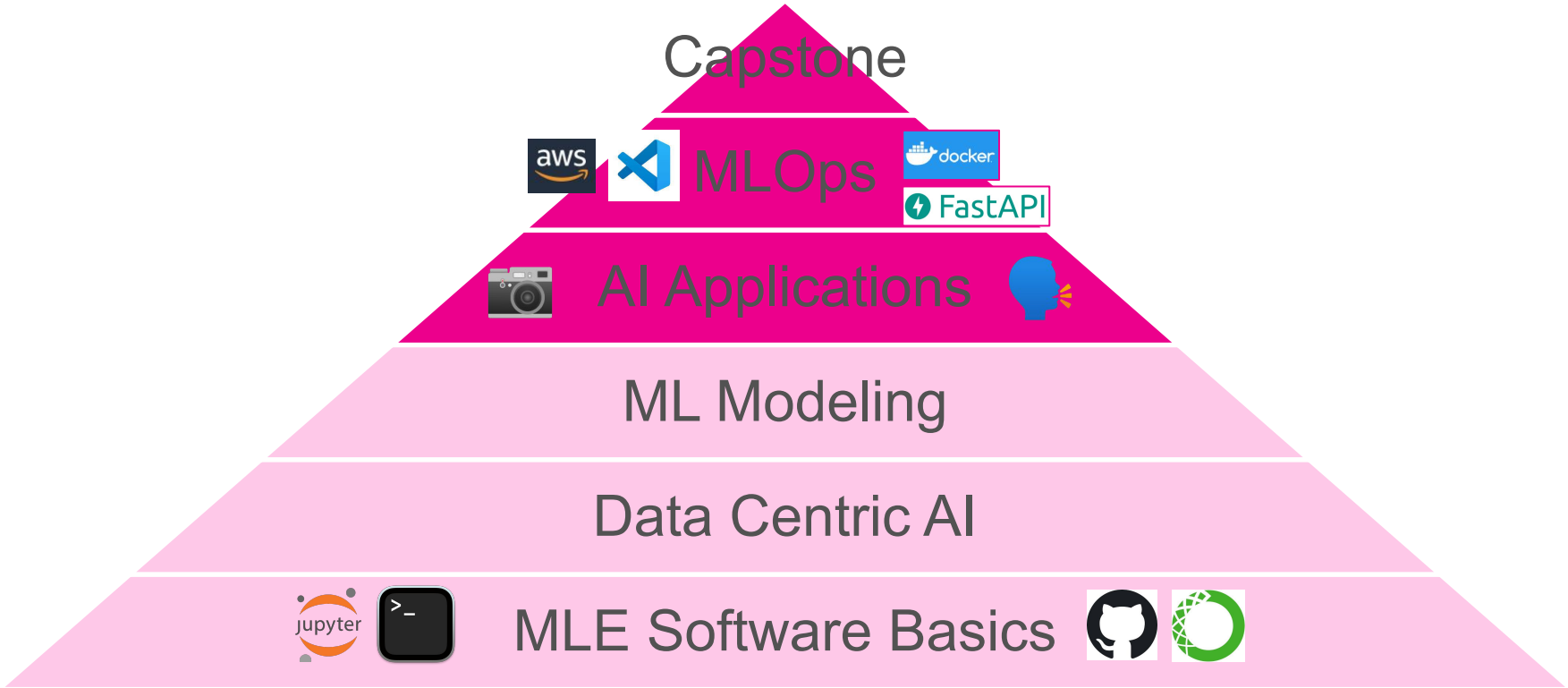


FourthBrain

MLE Program, Cohort 11 (MLE11)

Week 7: Neural Network Basics, CNN, RNN, GNN, GAN

Becoming a Machine Learning Engineer



Our Updated Curriculum!

1. ML Project Scoping
2. Real, Live Data Streams
3. Data Wrangling & Exploratory Analysis
4. Big Data

DATA CENTRIC AI

5. Supervised ML
6. Deep Learning
7. Unsupervised, Semi- & Self-supervised Learning

ML MODELING

8. Computer Vision
9. Natural Language Processing
10. Transformers & Fine Tuning Pre-Trained Networks

AI APPLICATIONS

11. Building ML Web Apps
12. Containerization
13. Model Serving
14. Machine Learning in Production

MLOps

Last Week!

Concepts

- Regression
- Classification
- Data Imbalance
- Accuracy Metrics
- AI Explainability

Hands on

- Predicting customer conversion using an explainable ML pipeline

🤖 This Week!

Concepts

- Neural Networks Basics
- Convolutional Neural Network
- Recurrent Neural Networks
- Graph Neural Networks
- Generative Adversarial Networks

Hands on

- Leveraging deep learning for tabular data

Agenda

- Theory (2.5 hrs)
  - Dive deep into a neural network
  - NN flavors
- Break 30 min
- Coding 2.5 hours

What questions do you have?

Intro to Artificial Neural Networks

Neural Networks

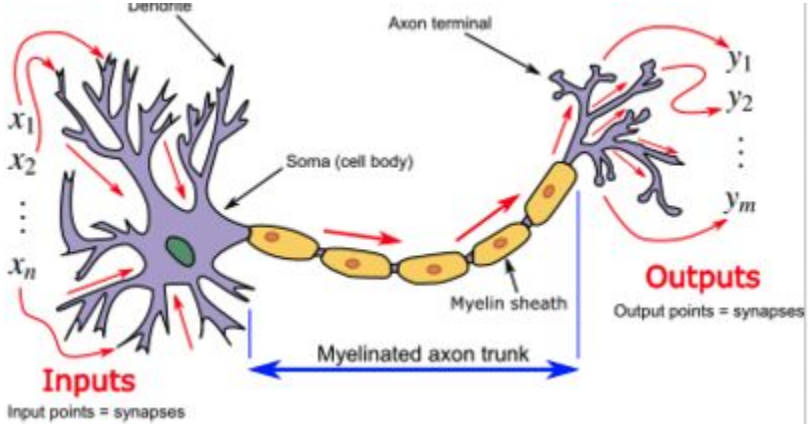
- Neural Networks rely on training data to learn and improve their accuracy over time.

- They are used to predict:
  - Continuous Data: i.e., the price of a house
  - Categories: i.e., a cat or a dog

- These networks usually take time to train and reach the levels of accuracy needed, but once there, they are powerful tools in Computer Science and Artificial Intelligent, that allow to classify and cluster data at a high velocity (and sometimes real-time)

History and why now?

- McCulloch-Pitts (MCP) neuron in 1943 - the earliest ANN model



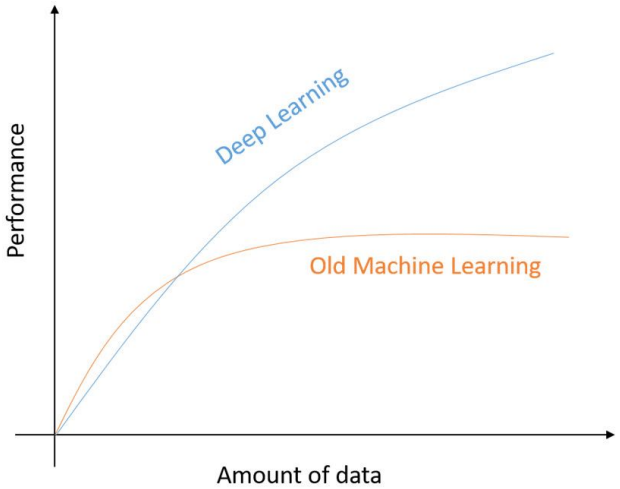
- Perceptron learning rule in 1957 - algorithm to automatically learn the optimal weights

Image source: https://en.wikipedia.org/wiki/Biological_neuron_model

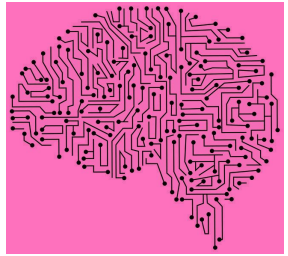
History and why now?

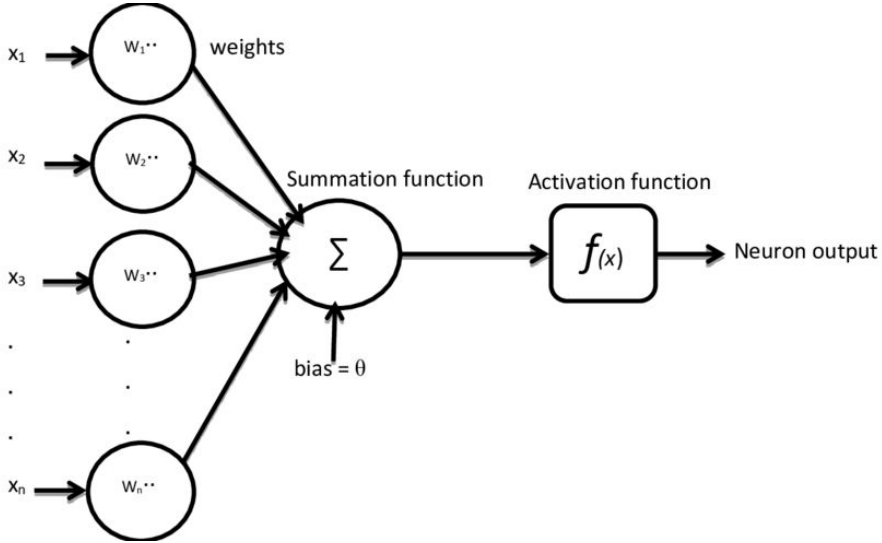
- Deep Learning (DL) is a branch of Machine Learning (ML) that is completely based on Artificial Neural Networks (ANN)

- The concept of DL is not new, however, the subject is on hype nowadays because of the exponential advancements made in computing power which allowed these networks to become realistically feasible



What is an artificial neuron?

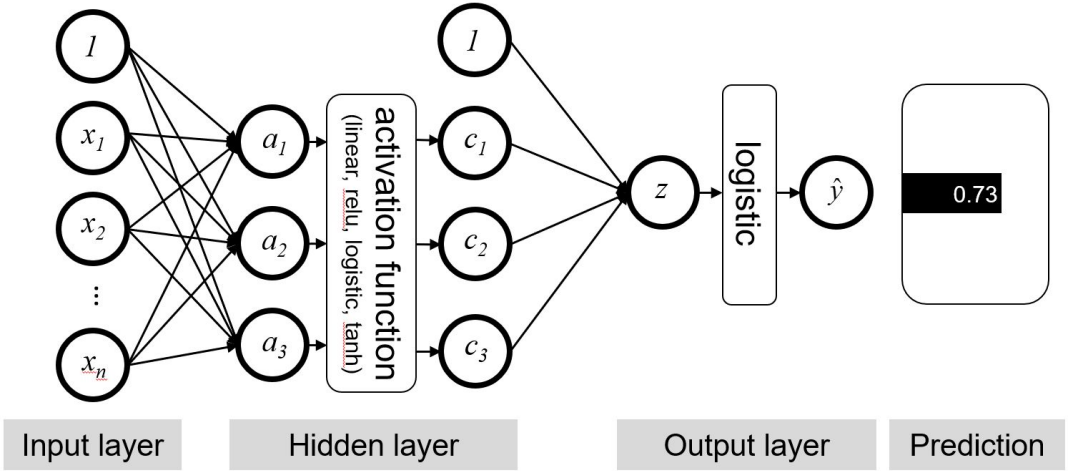




- Feedforward network with a single layer
- Linear Combination of inputs and weights
- Net Input value and the threshold (or activation function)
- Bias Unit (more resources for bias: Link1, Link2, Link3)
- Outputs

Image source https://www.researchgate.net/figure/The-structure-of-the-artificial-neuron_fig2_328733599

Artificial Neural Network



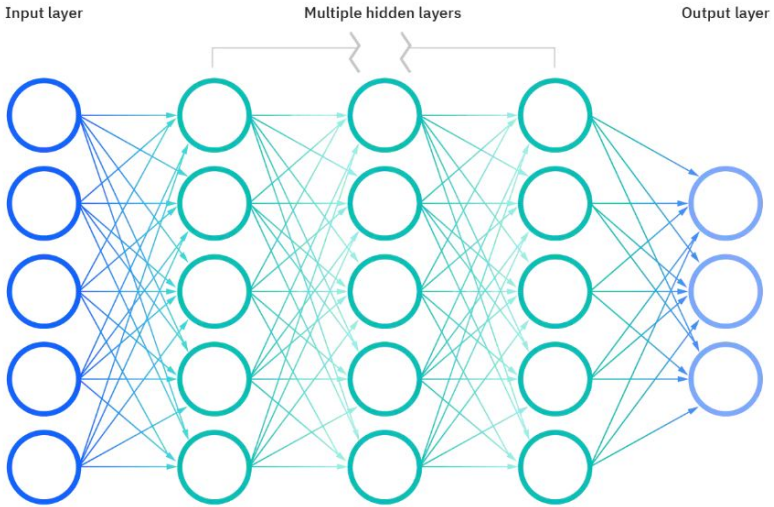
- fully-connected multilayer feedforward nn = multilayer perceptron
- Each Neural Network has 3 types of layers:

1. Input Layer: the first layer of the NN
2. Hidden Layers: any layer between the first and last layers
3. Output Layer: the last layer of the NN

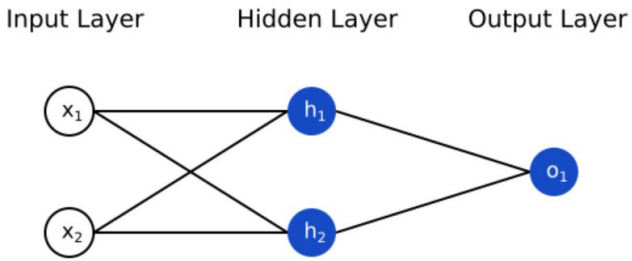
Image source https://carpentries-incubator.github.io/ml4bio-workshop/05-logit-ann/index.html

Neural Networks vs Deep Learning

- “Deep” in Deep Learning is just referring to the depth of layers in a Neural Network
- A Neural Network that consists of only 2 or 3 layers is a basic Neural Network
- A Neural Network with more than 3 layers is considered a Deep network



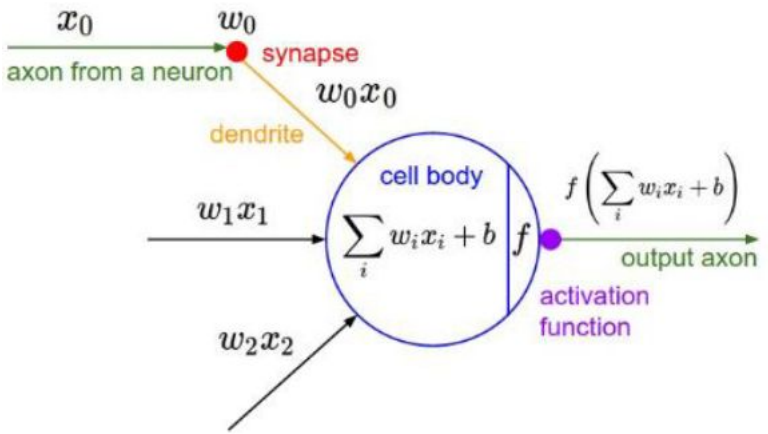
Hidden Layer



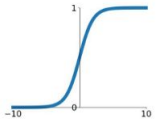
- The hidden layers might consist of more than 1 layer
- Notice in the picture that the inputs of the Output Layer (o1) are the outputs from h1 and h2 – That's what makes this a network!

Hidden Layer and activation functions

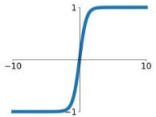
$$Y = Activation(w_0 + w_1x_1 + w_2x_2)$$



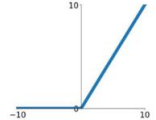
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



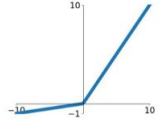
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

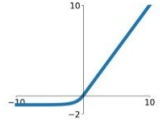


Leaky ReLU
$\max(0.1x, x)$



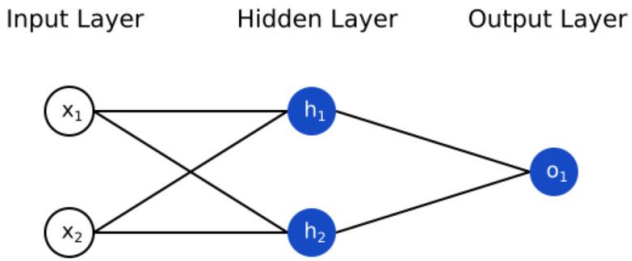
Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



[More on common Activation Functions]

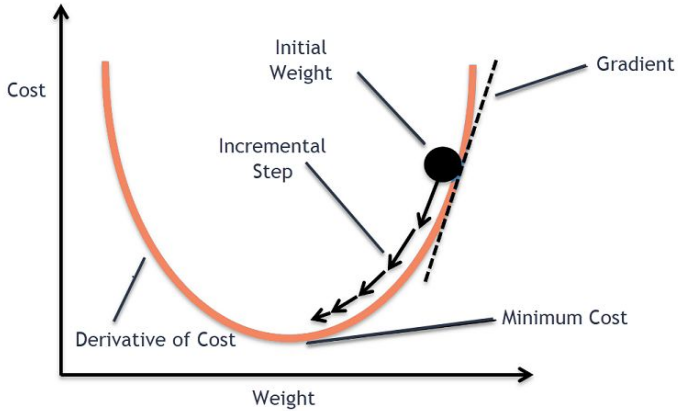
Output Layer



- Depending on the task: regression vs. classification
- linear function for regression outputs
- sigmoid units for binary classification
- softmax units for multiclass classification

Neural Network Training Vocabulary

- Once the model learned some weights, it is time to make it better!
- **Epochs** are iterations through the training dataset
- Goal is to minimize the **cost function (error)**
- **Batch** is a number of training samples
- **Gradient Descent** used to optimize by minimizing the error
  - batch vs. stochastic vs. mini-batch

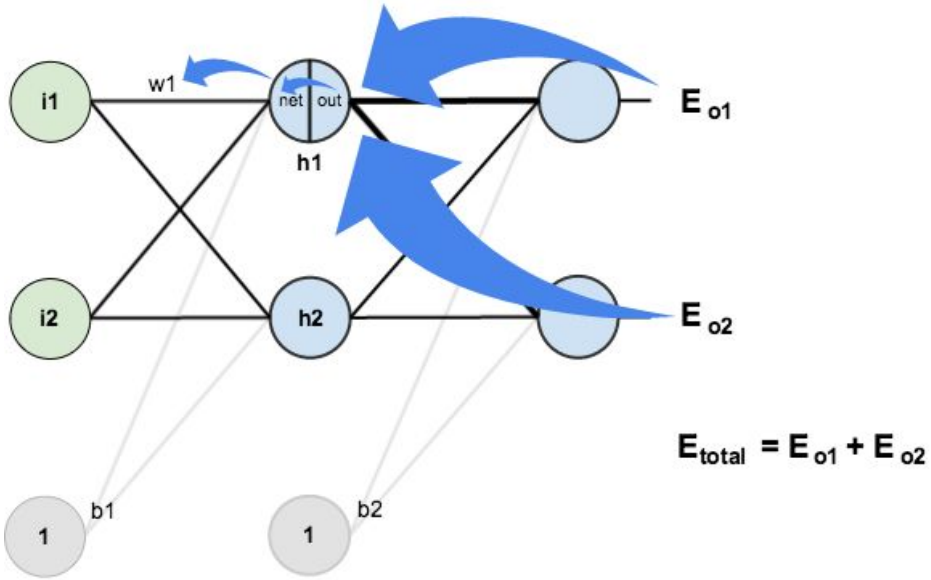


Backpropagation

1. To **calculate the loss** means to subtract the square difference between the expected output and the predicted output
2. Use this to **calculate the gradient** using algorithm called **backpropagation**
3. This gradient is used to **recalculate the weights(parameters)**
4. We stop when we are satisfied with the predicted output

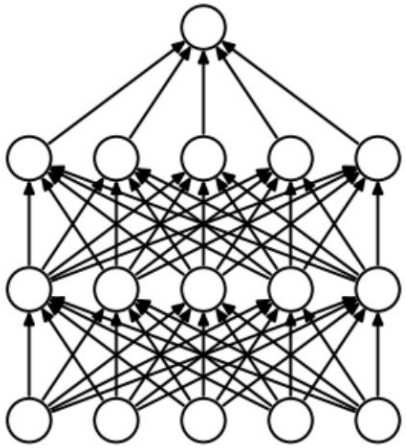
$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

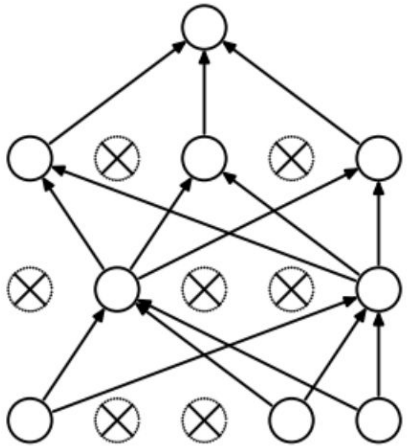


Dropout

At each training stage, individual nodes are either dropped out of the net with probability *1-p* or kept with probability *p*, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.



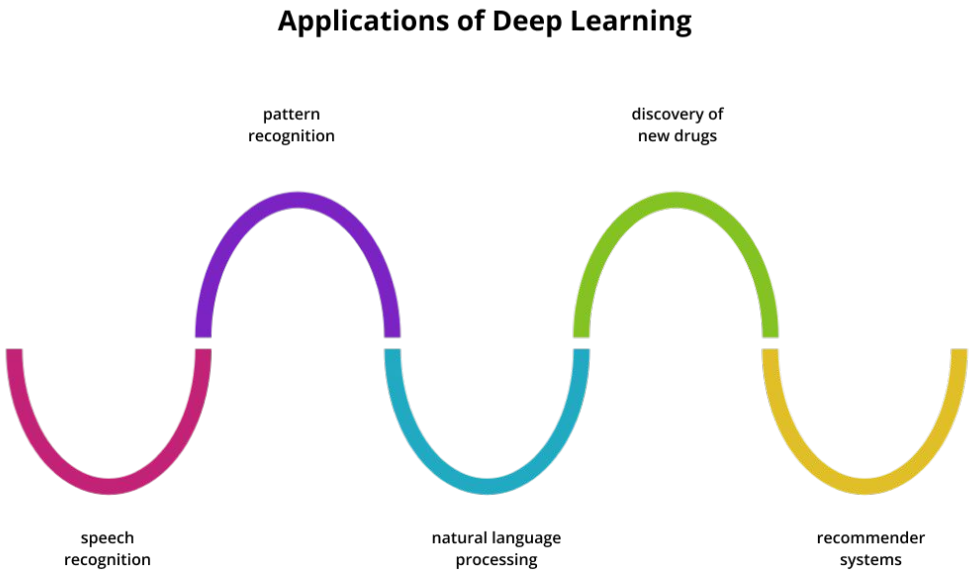
(a) Standard Neural Net



(b) After applying dropout.

Deep Learning Applications

- Healthcare
- Transportation
- Agriculture
- Self Driving Cars
- Fraud Detection
- Pixel Restauration
- Virtual Assistants
- Handwriting Generation



Deep Learning Frameworks

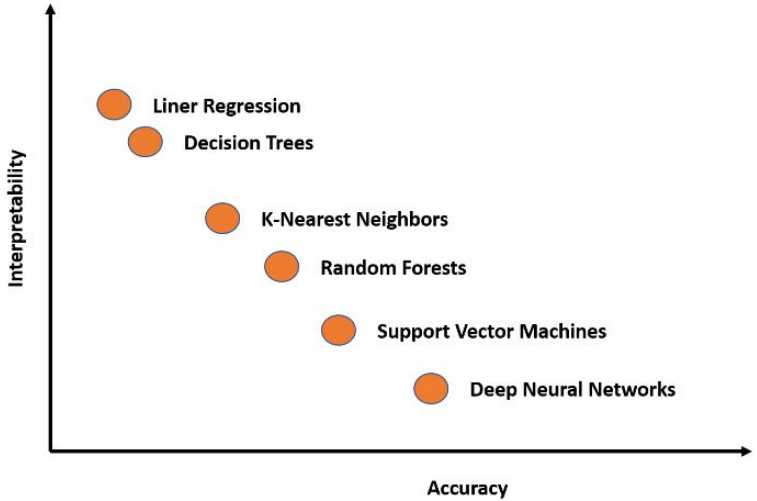
- Tensorflow
- Pytorch

TensorFlow

PyTorch

Deep Learning Limitations

- Data Availability
- Model Complexity
- Lack of Global Generalization
- Incapable Multitasking
- Hardware Dependence

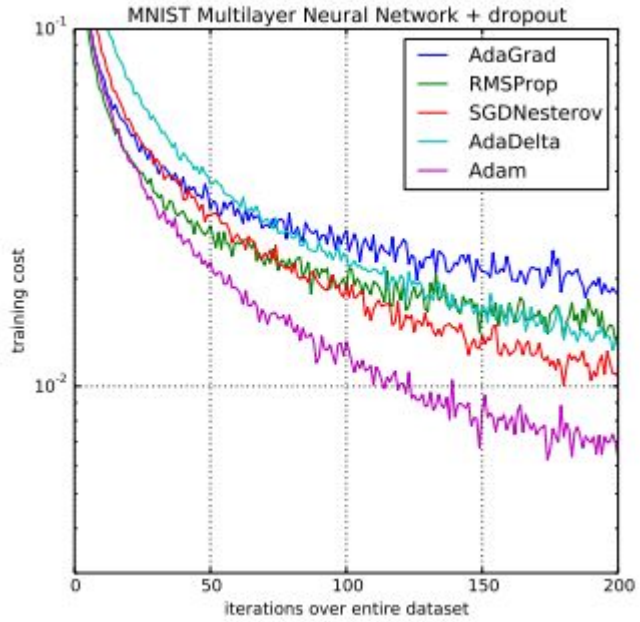


Hyperparameters

- Dropout
- Network Weight initialization
- Activation function
- Learning Rate
- Number of epochs

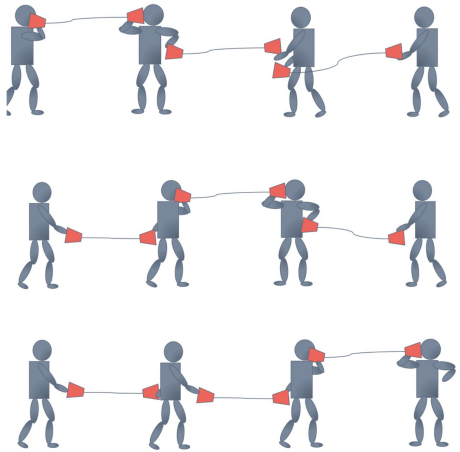
Optimization

- Adam



Batch normalization

- a layer that allows every layer of the network to do learning more independently
- Normalize the output of the previous layers
- Learning becomes efficient also it can be used as regularization to avoid overfitting of the model



$$y_i = BN_{\gamma,\beta}(x_i)$$

$$\mu_b = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_b^2 = \frac{1}{m}\sum_{i=1}^{m}(x - \mu_b)^2$$

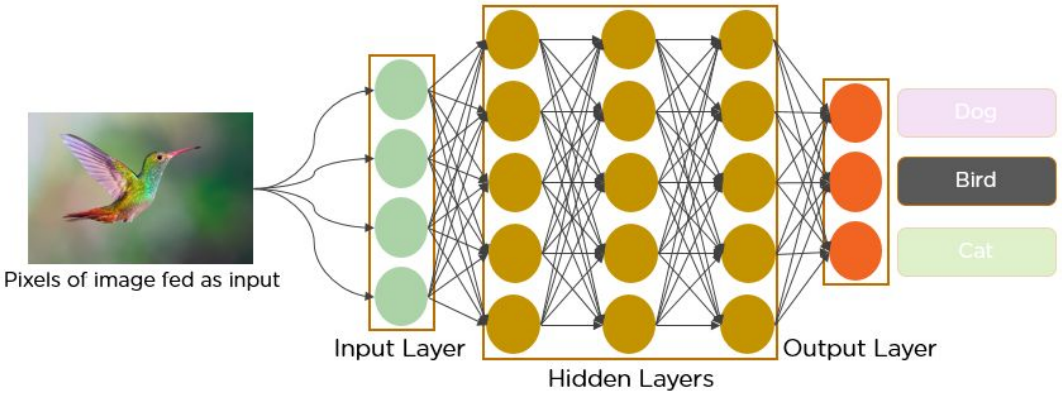
$$\hat{x}_i = \frac{x_i - \mu_b}{\sqrt{\sigma_\beta^2 + \epsilon}}$$

$$y_i = \gamma * \hat{x}_i + \beta$$

Convolutional Neural Network

What are Convolutional Neural Networks?

- The niche of Deep Learning algorithms
- Mostly used in the field of image recognition
- A mathematical model that uses a special method called convolution
- Consist of multiple steps hidden from the end-user

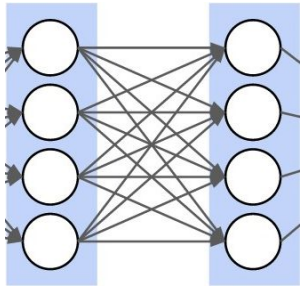


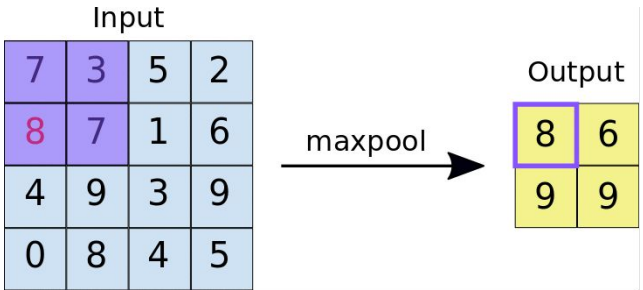
Convolutional Neural Networks

- Consist of various layers of artificial neurons
- The behavior of each CNN neuron is defined by the value of its weight
- This artificial neuron is capable of recognizing various visual features and specification
- The deeper you go into the CNN, the more high-level the information becomes:
  - Object Detection
  - Face Detection
  - Cat/Dog Classification

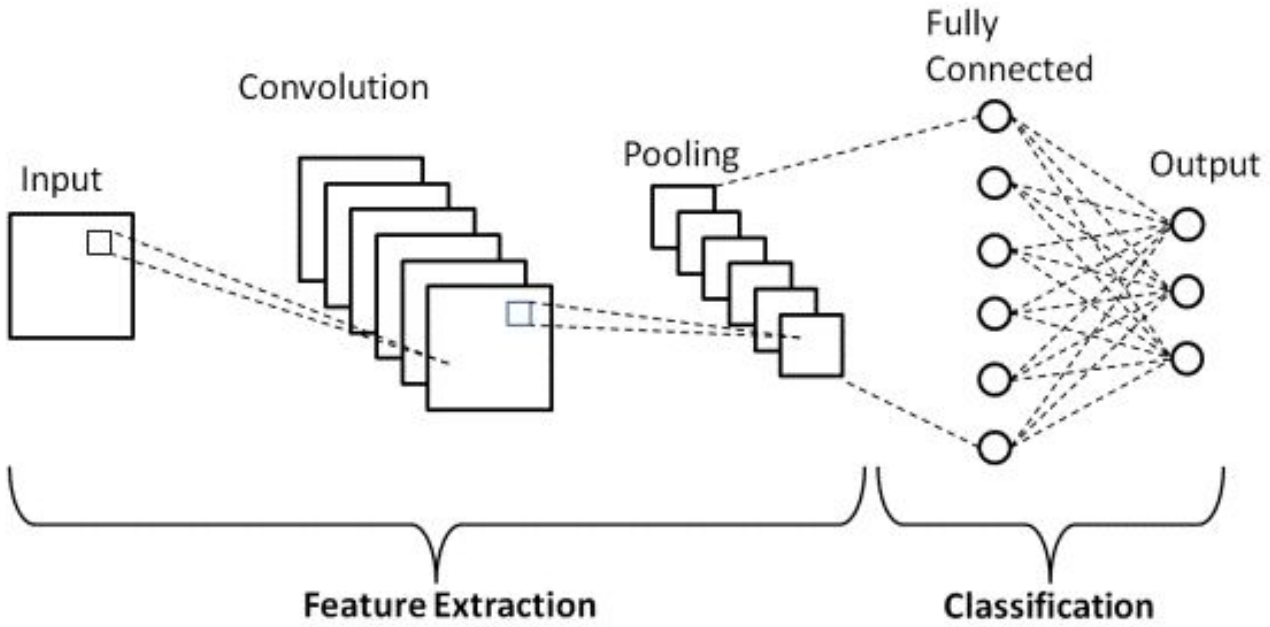
Layers in Convolutional Neural Networks

- Type of layers in CNN
  - Convolution
  - Pooling
  - Fully Connected
- As you go deeper in the NN:
  - The height and width decrease
  - The number of channels increase





Layers in Convolutional Neural Networks



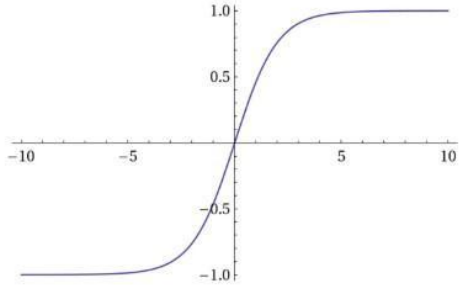
Pooling Layer in CNN

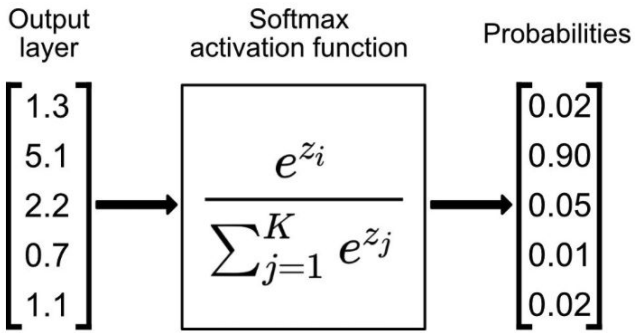
- Responsible for the reduction of the size (spatial) of the convolved feature.
- This decrease the computing power required to process the data by a significant reduction in dimension
- There are 2 types of pooling
  - Average Pooling
  - Max Pooling
- It is most common to have a 2x2 filter with stride = 2 for pooling. But sometimes you can find different filter sizes

CNN example:

- CONV – POOL – CONV – POOL – FC – FC – FC - SOFTMAX

	Activation Shape	Activation Size	# parameters
Input	(32,32,3)	3072	0
CONV1 (f=5, s=1)	(28,28,8)	6272	608
POOL1	(14,14,8)	1568	0
CONV1 (f=5, s=1)	(10,10,16)	1600	3216
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,120 (400*120 + 120)
FC4	(84,1)	84	10,164
Softmax	(10,1)	10	850 (84*10 + 10)

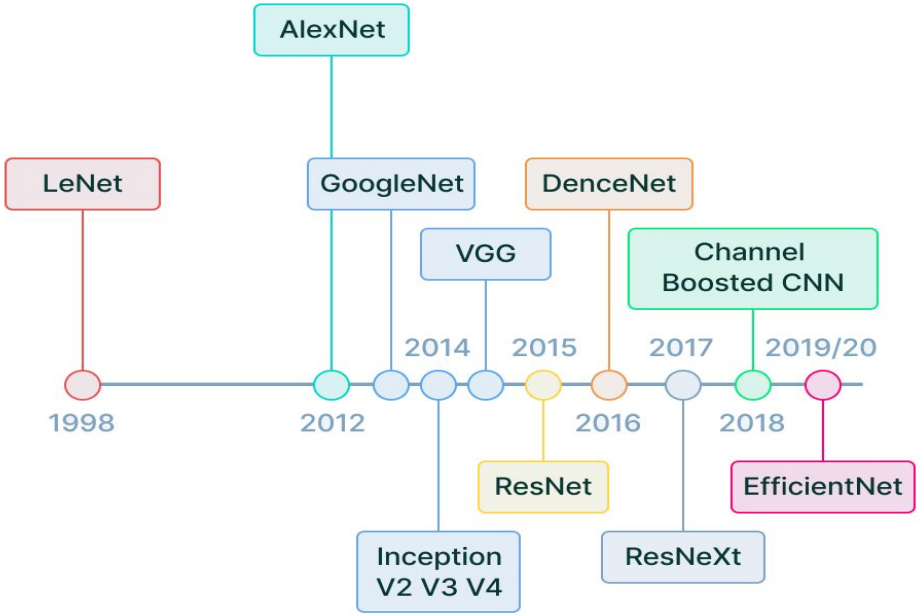




CNN example:

CNN History ➔ YoloV7 (CV)

- AlexNet
- VGGNet
- GoogLeNet
- ResNet



Recurrent Neural Networks

Recurrent Neural Networks (RNN)

- A RNN is a type of artificial neural network which uses sequential data or time-series data.

- These DL algorithms are commonly used for problems such as:
  - Language translation
  - Natural language processing (NLP)
  - Speech recognition
  - Image captioning

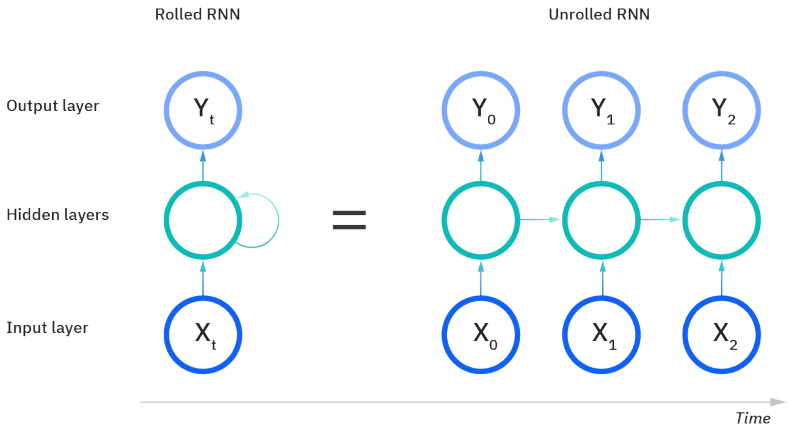
- They are incorporated into popular applications such as:
  - Siri
  - Voice Search
  - Google Translate

Recurrent Neural Networks (RNN)

- Like CNN, recurrent neural networks utilize training data to learn
- RNN are distinguished by their “memory” as they take information from prior inputs to influence the current input and output
- Opposing to traditional NN, that assume that inputs and outputs are independent of each other, the output of RNN depend on the prior elements within the sequence
- While future events would also be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions

RNN in depth

- The “rolled” visual of the RNN represents the whole neural network
- It also represents the entire predicted phase
- The “unrolled” visual represents the individual layers – time steps – of a neural network
- Each layer maps to a single word in that phrase, such as “weather”
- Prior inputs, such as “feeling” and “under”, would be represented as a hidden state in the third timestep to predict the output in the sequence – “the”.



RNN in depth

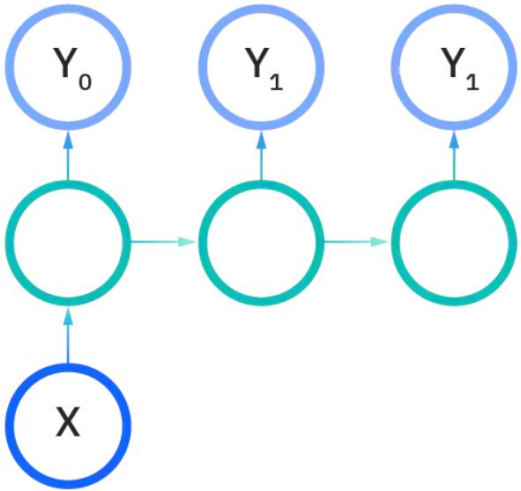
- RNN share parameters across each layer of the network
- RNN have the same weight parameter within each layer of the network, CNN have different weights across each node
- RNN weights are adjusted in the process of backpropagation and gradient descent (which are out of the scope of this lecture) to facilitate reinforcement learning.
- Recurrent neural networks leverage backpropagation through time (BPTT) algorithm to determine the gradients

Types of RNN

One to One

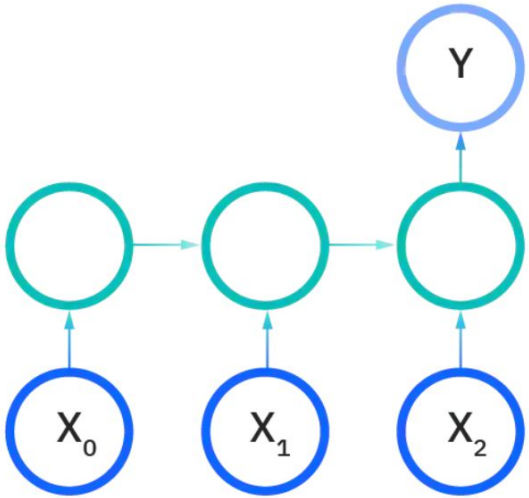
Example: Traditional RNN

One to Many



Example: Image Captioning

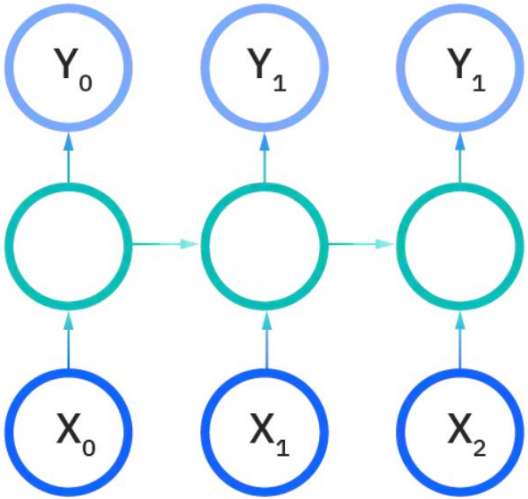
Many to one

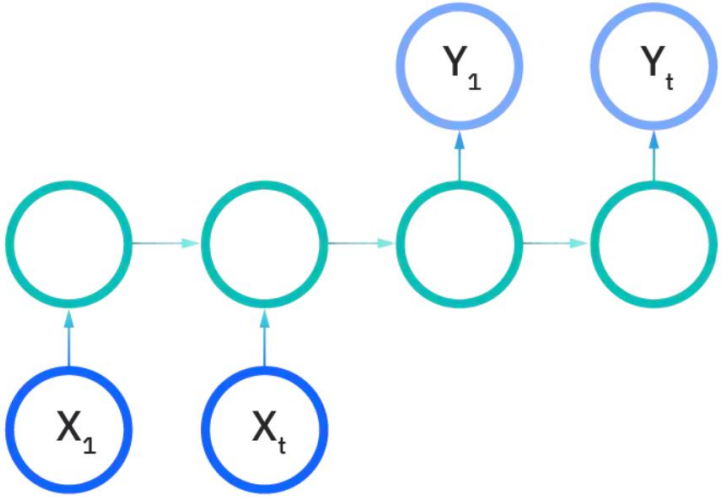


Example: Sentiment Analysis

Types of RNN

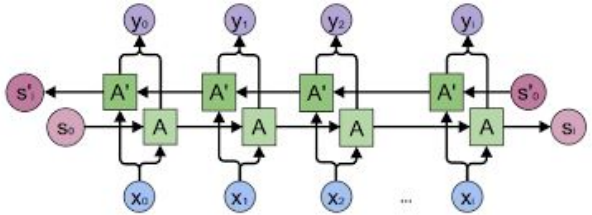
Many to Many



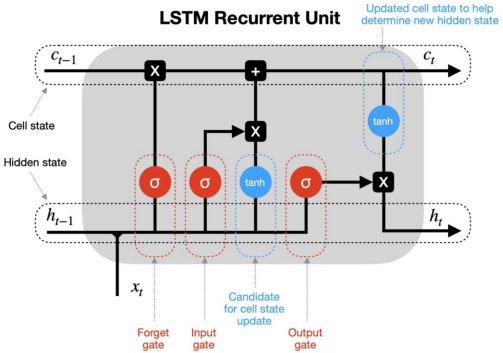


Example: Machine Translation

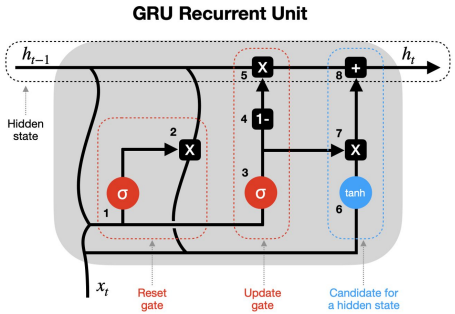
Variant RNN architecture



Bidirectional recurrent neural networks (BRNN)



Long short-term memory (LSTM)

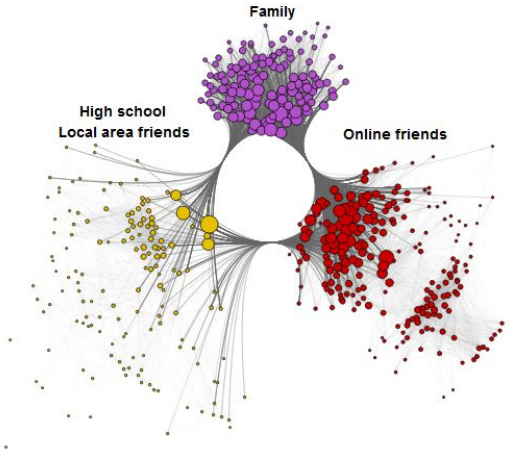


Gated recurrent units (GRUs)

Graph Neural Networks

GNNs

- Graph represents a relation (edge) between the nodes.
  - directed or undirected
- Applications examples: social networks, biology, recommendation systems....



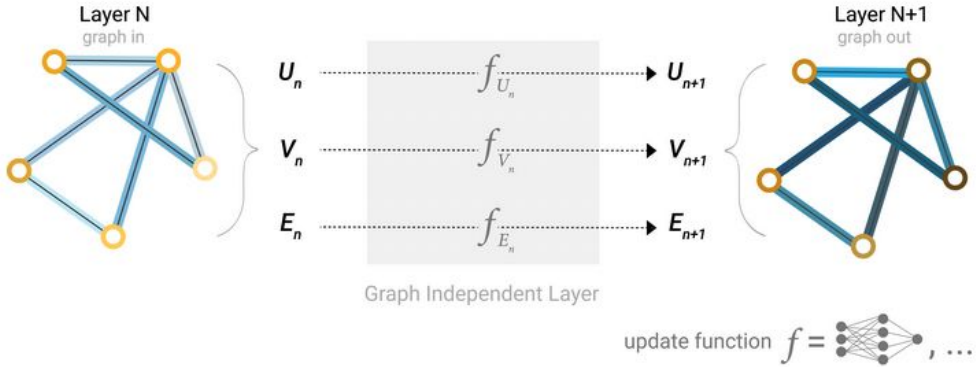
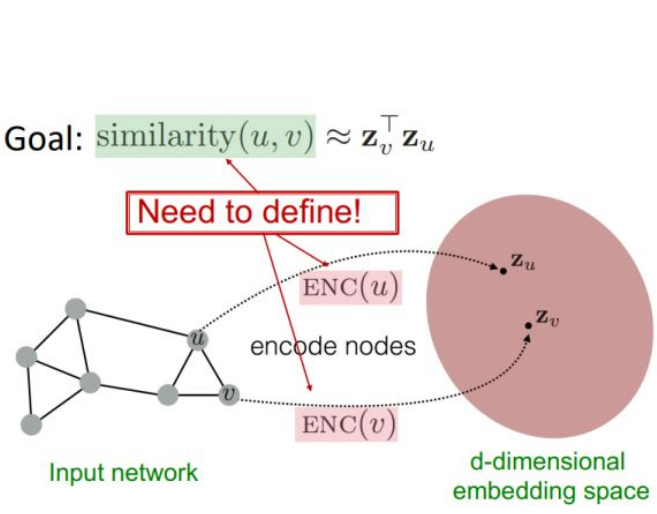


Image source: https://distill.pub/2021/gnn-intro/

GNNs



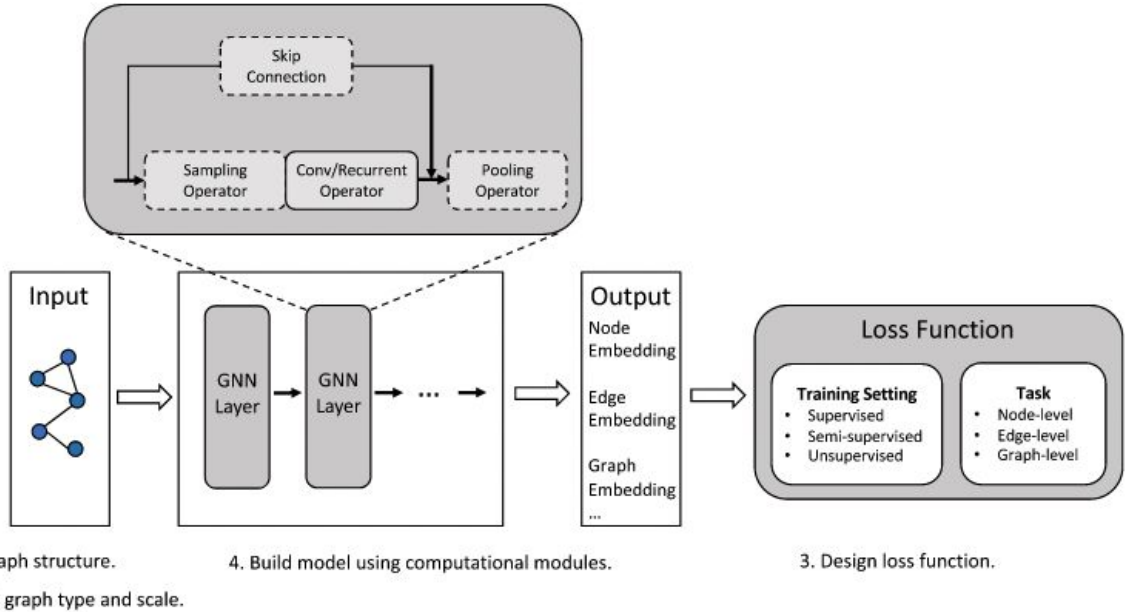
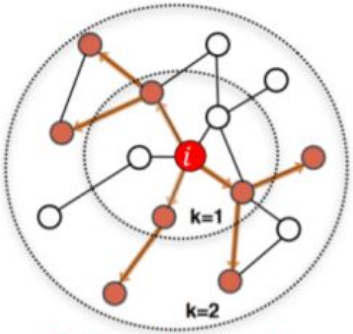
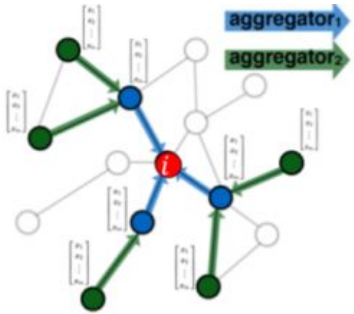


Image source: https://arxiv.org/ftp/arxiv/papers/1812/1812.08434.pdf

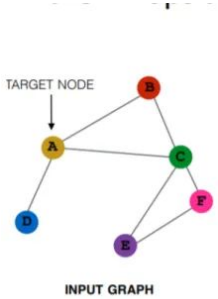
GNNs

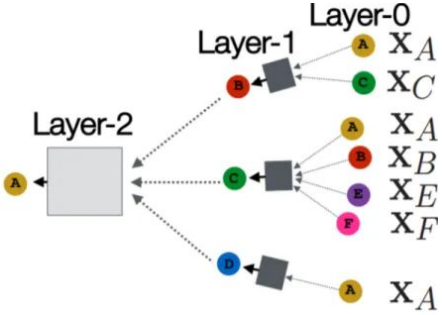


Determine node computation graph



Propagate and transform information





$$h_v^k = \sigma(W_k \sum \frac{h_u^{k-1}}{| N(v)} + B_k h_v^{k-1}) \text{ where } k = 1, ..., k-1$$

Image source: https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

GNN Applications

- Traffic prediction
  - Ex. Google Maps
- Physics and particle discovery
  - Ex. Fermilab for particle discovery
- Drug Discovery
  - Ex. Open Catalyst Project
- Recommender Systems
  - Ex. UberEats

Generative Adversarial Networks

Generative Adversarial Networks (GANs)

- Generative
  - Learn a generative model
- Adversarial
  - Trained in an adversarial setting
- Networks
  - Use Deep Neural Networks

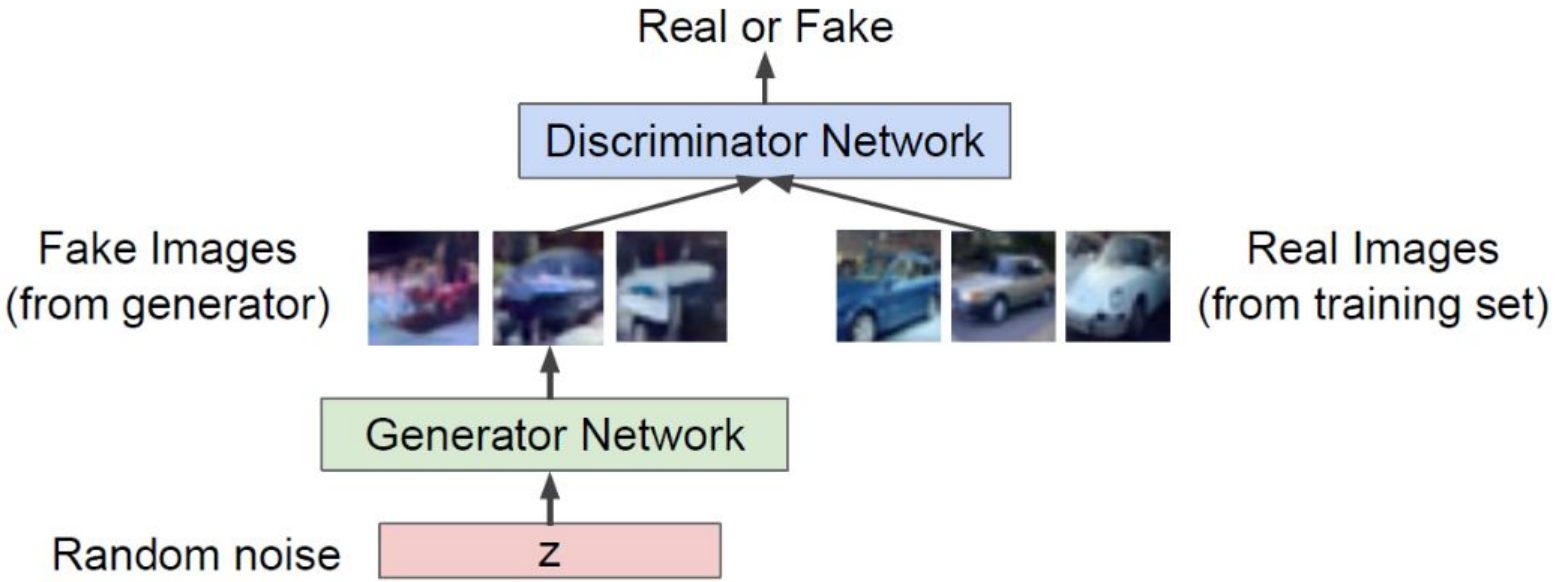
Sub-models
- Generator model
- Discriminator

What are GANs?

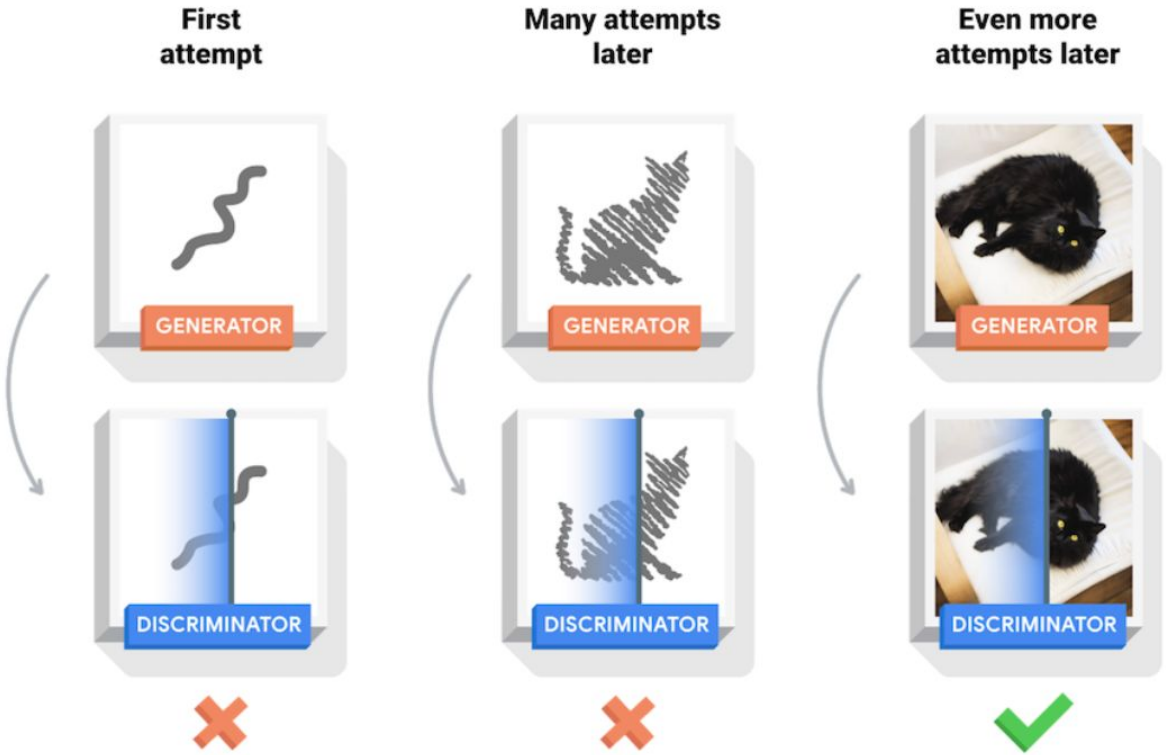
- GANs introduce the concept of adversarial learning, as they lie in the rivalry between 2 NN.

- These techniques have enabled researchers to create realistic-looking but entirely computer-generated photos of people's faces
- GANs can create:
  - 2D images
  - 3D images
  - Videos
  - Text
  - Sounds
  - Etc.

GAN's formulation

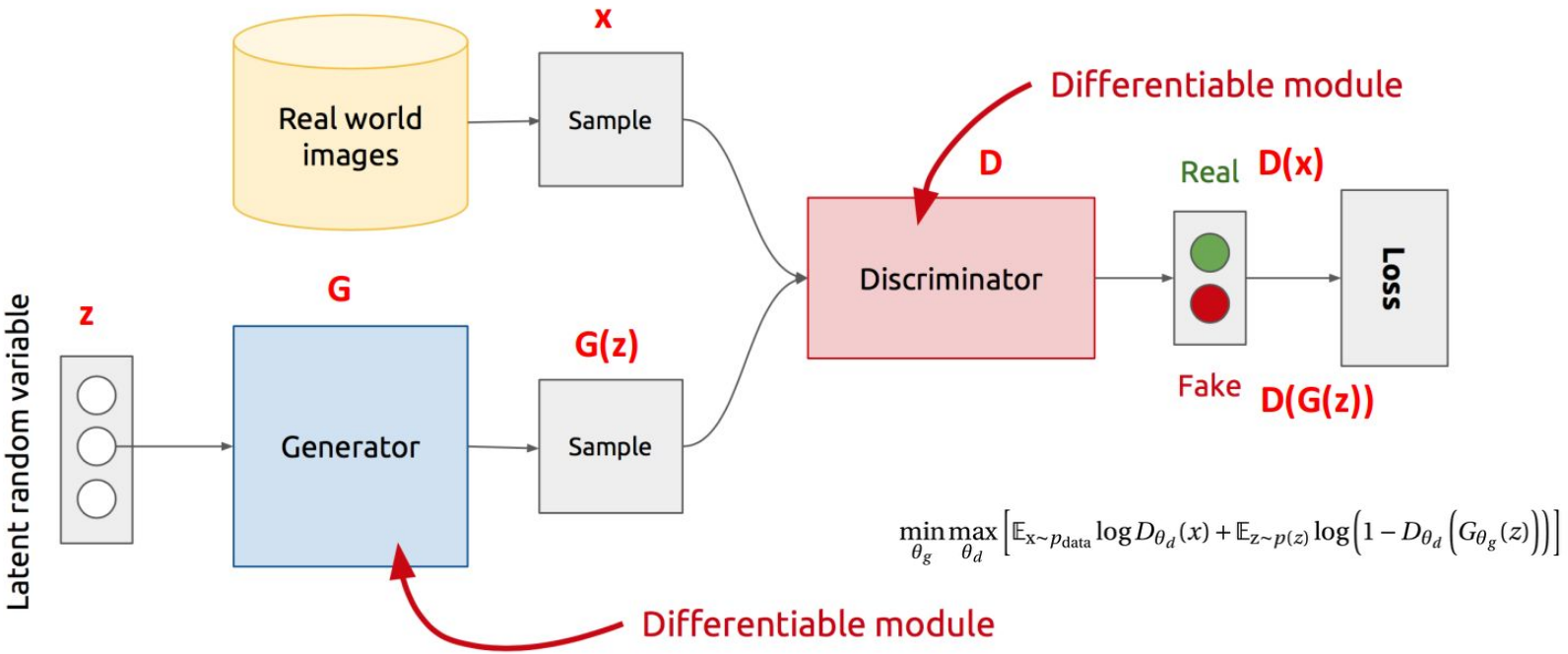


When the Generator “fools” the Discriminator, the mission is accomplished!



GANs Architecture

- Z represents some random noise



$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_d} \left(G_{\theta_g}(z) \right) \right) \right]$$

GANs Pros and Cons

- Pros
  - Plenty of existing work
  - Sampling (generation) is straightforward
  - Robust to Overfitting since Generator never sees the training data
  - Good at capturing modes of the distribution

- Cons
  - Probability distribution is not straightforward
  - Training is Hard

Breakout

5 min
(3-4 per room)

What applications of GANs have you seen, or can you think of?

Designate *one person to share* from your breakout room

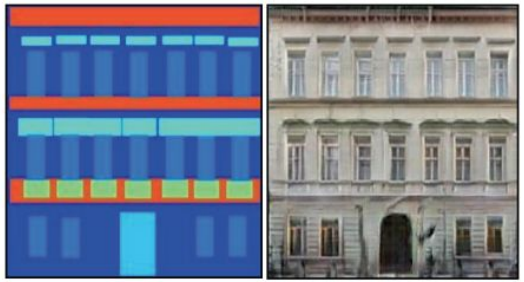
Applications of conditional GANs

Image to Image translation

Labels to Street Scene

input output

Labels to Facade



input output

BW to Color

input output

Aerial to Map

input output

Day to Night

input output

Edges to Photo

input output

Applications of conditional GANs

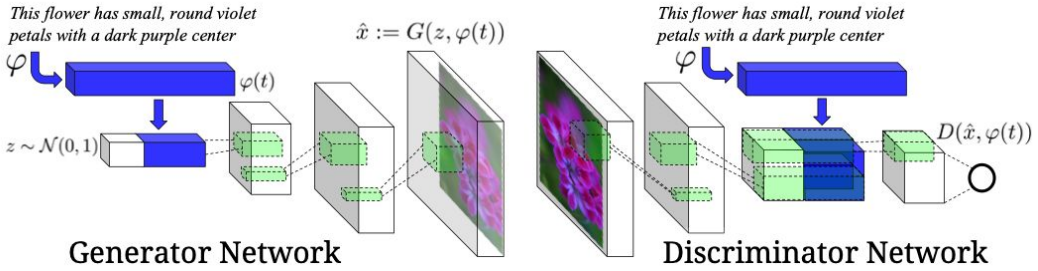
Text to Image Synthesis

this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

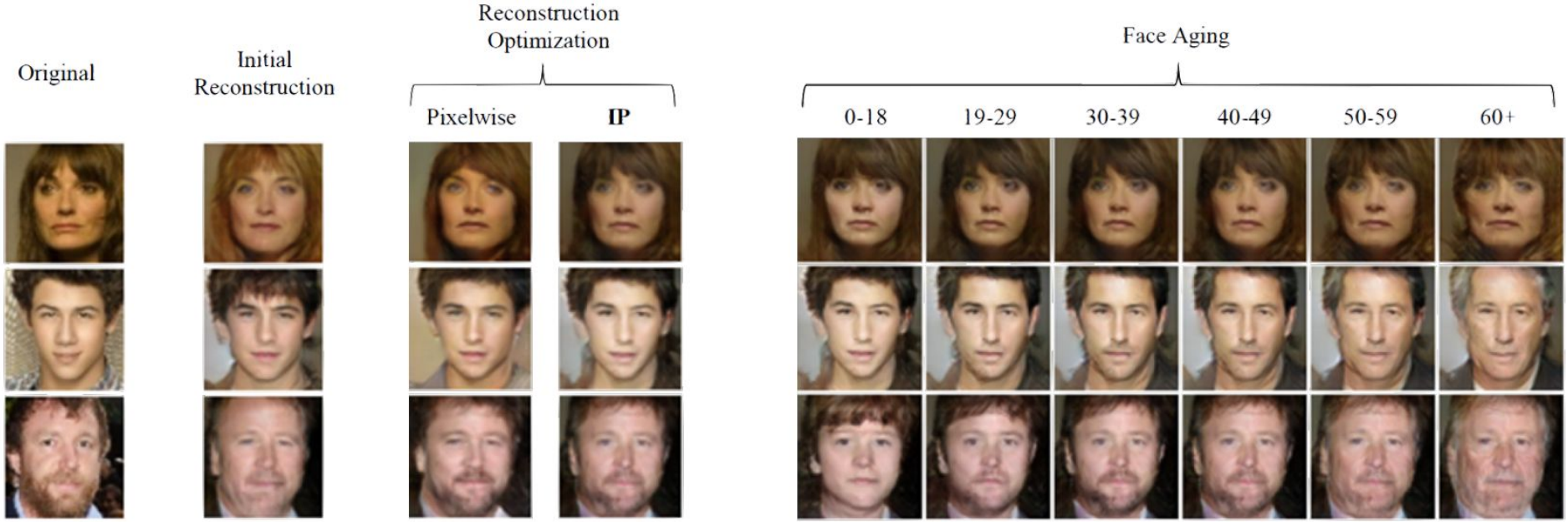
this white and yellow flower have thin white petals and a round yellow stamen



Paper

Applications of conditional GANs

Face Aging

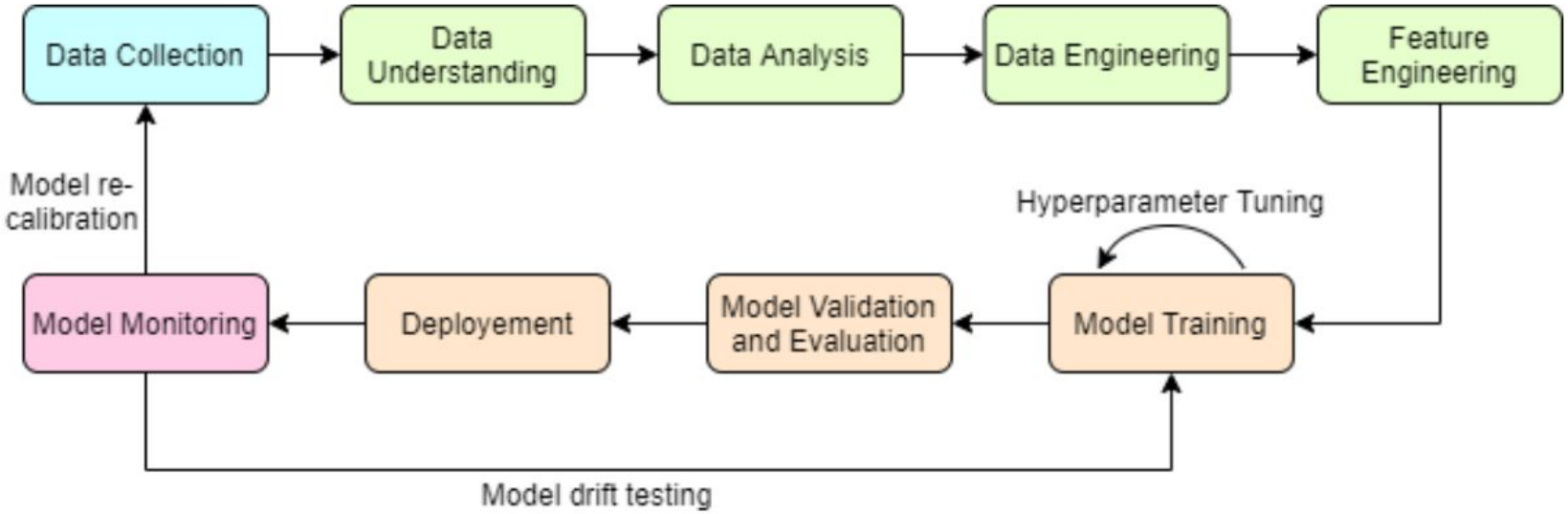


Paper Code

AutoML

AutoML

Automated Machine Learning (AutoML)



AutoML

- Automated Machine Learning (AutoML)
- It helps in automating critical components of the ML pipeline
- This pipeline consists of:
  - Data Understanding
  - Data Engineering
  - Feature Engineering
  - Model Training
  - Hyperparameter Tuning
  - Model Monitoring and etc.

"Efficient and Robust Automated Machine Learning."

Google AutoML

Data Version Control

Data Version Control

- Live data systems are continuously ingesting newer data points
- Different users carry out different experiments on the same datasets
- This leads to multiple versions of the same dataset
- This also means there is not a single source

Data Version Control - DVC

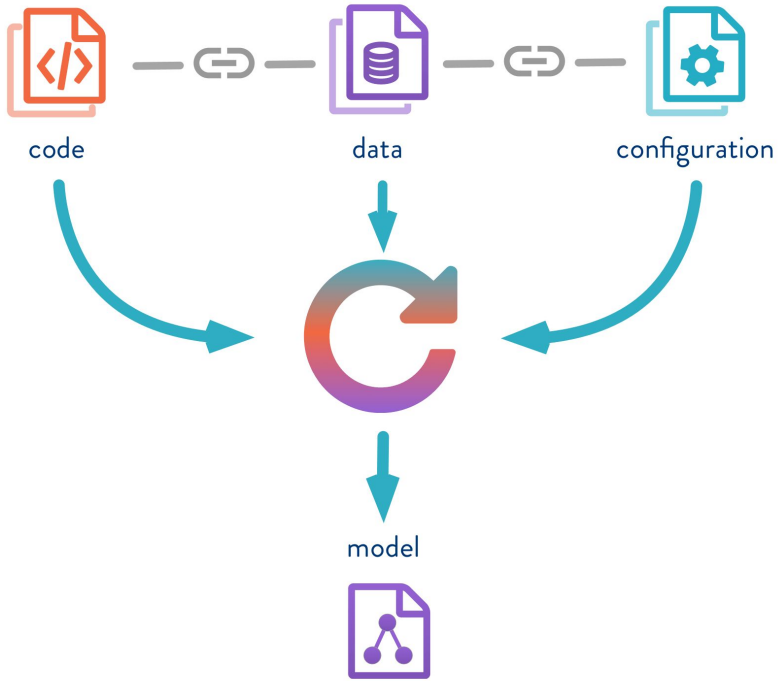
- In software engineering, the solution to version control is Git
- Git allows to:
  - Commit changes
  - Create different branches from source
  - Merge back the branches
- DVC is purely the same paradigm, but for datasets

Data Version Control - Solutions

- DVC is a system that involves tracking the datasets by registering changes on a particular dataset
- There are multiple DVC solutions (free and paid)
- We will be exploring “DVC”, a open-source packages widely used in data science

DVC

- DVC tracks ML models and Datasets
- It is built to make ML models sharable and reproducible
- It is designed to handle:
  - Large files
  - Datasets
  - ML models
  - Metrics
  - Code



DVC - Features

Git-compatible

- DVC runs on top of any git repository
- It is compatible with any standard Git server/provider (GitHub, GitLab, etc.)
- Data file contests can be shared by network-accessible storage or any supported cloud solution

DVC - Features

Accessibility

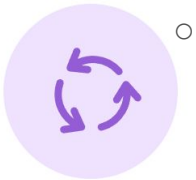
- You can extend DVB capabilities and your ML experimentation workflows directly into your IDE

- As an example, DVC works with Visual Studio Code where you can:
    - Manage your data
    - Run experiments
    - Compare metrics
    - Visualize plots

DVC - Features

Reproducible

- A single ‘dvc repo’ command reproduces experiments end-to-end
- DVC guarantees reproducibility by consistently maintaining a combination of:
  - Input data
  - Configuration
  - Code that was initially used to run an experiment



Data Version Control - DVC

Model Version Control

What You’re Realizing

MLflow

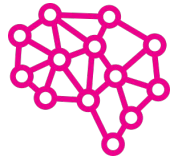
- MLflow is an open-source platform
- It manages the ML lifecycle
- It includes:
  - Experimentation
  - Reproducibility
  - Deployment
  - Central model registry

MLflow - Tracking

- MLflow Tracking is used to track/record the experiments
- Steps:
  - Store the logging parameters
  - Store the metrics
  - Store the output files
  - Visualize the results of all the experiments on the localhost

MLflow - Project

- MLflow Project is a component employed for packaging data science code
- It aims for making the code:
  - Reusable
  - Reproducible



ML Flow | Integrations

TensorFlow · PyTorch · Keras · Spark · scikit-learn · 🤗

MONAI · RAPIDS · H2O.ai · python · R · Java

mleap · ONNX · GLUON · XGBoost · LightGBM · spaCy

fast.ai · statsmodels · CatBoost · PyCaret · ALGORITHMIA · ZenML

OPTUNA · RAY · CONDA · kubernetes · docker · Amazon SageMaker

Azure Machine Learning · Google Cloud · databricks

MLflow - Models

- MLflow Models is used to package machine learning models
- There are multiple methods to save and load MLflow models
- I.e. train a sklearn model and later log it as an MLflow artifact for the current run using “log_model”

MLflow - Registry

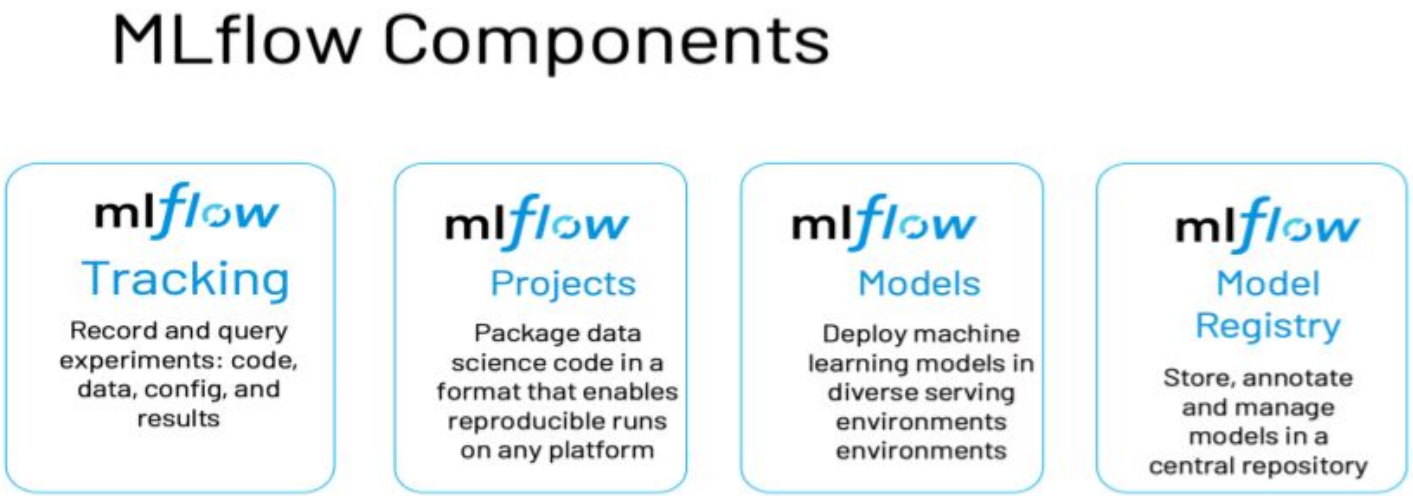
- MLflow Registry is a tool specialized in adding the model to the model registry
- It allows providing the chronology of the models produced from staging to production
- It permits adding description to the models at each given run

Relating to Data Version Control

- In a machine learning environment, we want to have several versions of the same “model” trained on different versions of the same dataset
- Model re-training to include newer data points
- The above processes require proper audit and versioning, otherwise, this would create a tangled web of datasets and experiments

MLFlow

- open source platform to manage model experimentation, reproducibility, deployment and to keep track of the models in the model registry



Feedback on Lecture and Concepts?