ARTIFICIAL INTELLIGENCE SOFTWARE DEVELOPMENT

CST8510 Week 4

Dr. Hari M Koduvely

Agenda for Today

- ❑ Theory: 5:30PM – 7:30PM
  - Choosing the Right ML Algorithm
  - Distributed Training
  - Auto ML
- ❑ Lab: 7:30PM – 9:30PM
  - Standup Meetings

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

- ❑ Avoid human biases in selecting models

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

1. Do not use only State-of-the-Art (SOTA) models
   - *Do not jump straight away to SOTA models*
   - *SOTA models are typically evaluated in academic settings*
   - *Using standard datasets*
   - *SOTA models many not be the best for your dataset*
   - *They may be more expensive to train*
   - *May have more latency during inference*

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

- ❑ Start with the simplest models
  - *Simple models are easy to comprehend and to explain the predictions.*
  - *They are also easy to deploy.*
  - *Early deployment helps in many validations.*
  - *Simple models will help one to debug more complex models.*
  - *They will give a baseline to compare more complex models.*

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

- ❑ Evaluate performance at different time points
  - ▪ Use Learning Curve

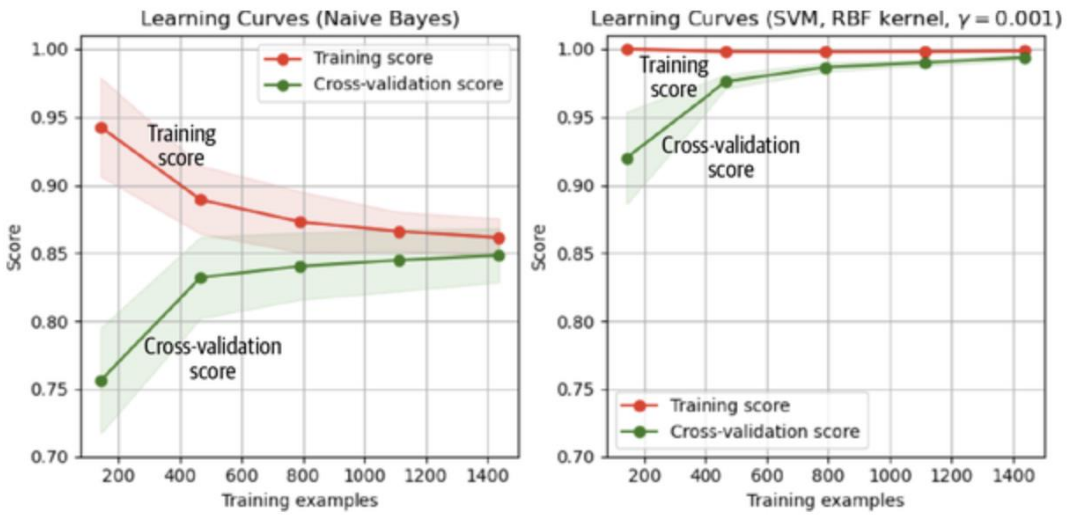


Figure 6-1. The learning curves of a naive Bayes model and an SVM model. Source: scikit-learn

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

- ❑ Evaluate trade-offs
  - False Positive vs False Negative Trade off
  - Accuracy vs Computational Cost
  - Latency vs Accuracy

Choosing the Right ML Algorithm

Six tips for choosing the right ML Algorithm for your problem.

- ❑ Understand your model's assumptions
    - All models are some approximations of reality
    - "All models are wrong, but some are useful." - George Box 1976
    - Some common set of assumptions
        - Normality
        - IID
        - Smoothness
        - Tractability
        - Boundaries
        - Conditional independence

Choosing the Right ML Algorithm

datacamp

Top Machine Learning Algorithms

		ALGORITHM	DESCRIPTION	APPLICATIONS	ADVANTAGES	DISADVANTAGES
Supervised Learning	Linear Models	Linear Regression	A simple algorithm that models a linear relationship between inputs and a continuous numerical output variable	USE CASES 1. Stock price prediction 2. Predicting housing prices 3. Predicting customer lifetime value	1. Explainable method 2. Interpretable results by its output coefficients 3. Faster to train than other machine learning models	1. Assumes linearity between inputs and output 2. Sensitive to outliers 3. Can underfit with small, high-dimensional data
		Logistic Regression	A simple algorithm that models a linear relationship between inputs and a categorical output (1 or 0)	USE CASES 1. Credit risk score prediction 2. Customer churn prediction	1. Interpretable and explainable 2. Less prone to overfitting when using regularization 3. Applicable for multi-class predictions	1. Assumes linearity between inputs and outputs 2. Can overfit with small, high-dimensional data
		Ridge Regression	Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients closer to zero. Can be used for classification or regression	USE CASES 1. Predictive maintenance for automobiles 2. Sales revenue prediction	1. Less prone to overfitting 2. Best suited where data suffer from multicollinearity 3. Explainable & interpretable	1. All the predictors are kept in the final model 2. Doesn't perform feature selection
		Lasso Regression	Part of the regression family — it penalizes features that have low predictive outcomes by shrinking their coefficients to zero. Can be used for classification or regression	USE CASES 1. Predicting housing prices 2. Predicting clinical outcomes based on health data	1. Less prone to overfitting 2. Can handle high-dimensional data 3. No need for feature selection	1. Can lead to poor interpretability as it can keep highly correlated variables
	Tree-Based Models	Decision Tree	Decision Tree models make decision rules on the features to produce predictions. It can be used for classification or regression	USE CASES 1. Customer churn prediction 2. Credit score modeling 3. Disease prediction	1. Explainable and interpretable 2. Can handle missing values	1. Prone to overfitting 2. Sensitive to outliers
		Random Forests	An ensemble learning method that combines the output of multiple decision trees	USE CASES 1. Credit score modeling 2. Predicting housing prices	1. Reduces overfitting 2. Higher accuracy compared to other models	1. Training complexity can be high 2. Not very interpretable
		Gradient Boosting Regression	Gradient Boosting Regression employs boosting to make predictive models from an ensemble of weak predictive learners	USE CASES 1. Predicting car emissions 2. Predicting ride hailing fare amount	1. Better accuracy compared to other regression models 2. It can handle multicollinearity 3. It can handle non-linear relationships	1. Sensitive to outliers and can therefore cause overfitting 2. Computationally expensive and has high complexity
		XGBoost	Gradient Boosting algorithm that is efficient & flexible. Can be used for both classification and regression tasks	USE CASES 1. Churn prediction 2. Claims processing in insurance	1. Provides accurate results 2. Captures non linear relationships	1. Hyperparameter tuning can be complex 2. Does not perform well on sparse datasets
		LightGBM Regressor	A gradient boosting framework that is designed to be more efficient than other implementations	USE CASES 1. Predicting flight time for airlines 2. Predicting cholesterol levels based on health data	1. Can handle large amounts of data 2. Computational efficient & fast training speed 3. Low memory usage	1. Can overfit due to leaf-wise splitting and high sensitivity 2. Hyperparameter tuning can be complex
Unsupervised Learning	Clustering	K-Means	K-Means is the most widely used clustering approach—it determines K clusters based on euclidean distances	USE CASES 1. Customer segmentation 2. Recommendation systems	1. Scales to large datasets 2. Simple to implement and interpret 3. Results in tight clusters	1. Requires the expected number of clusters from the beginning 2. Has troubles with varying cluster sizes and densities
		Hierarchical Clustering	A "bottom-up" approach where each data point is treated as its own cluster—and then the closest two clusters are merged together iteratively	USE CASES 1. Fraud detection 2. Document clustering based on similarity	1. There is no need to specify the number of clusters 2. The resulting dendrogram is informative	1. Doesn't always result in the best clustering 2. Not suitable for large datasets due to high complexity
		Gaussian Mixture Models	A probabilistic model for modeling normally distributed clusters within a dataset	USE CASES 1. Customer segmentation 2. Recommendation systems	1. Computes a probability for an observation belonging to a cluster 2. Can identify overlapping clusters 3. More accurate results compared to K-means	1. Requires complex tuning 2. Requires setting the number of expected mixture components or clusters
	Association	Apriori algorithm	Rule based approach that identifies the most frequent itemset in a given dataset where prior knowledge of frequent itemset properties is used	USE CASES 1. Product placements 2. Recommendation engines 3. Promotion optimization	1. Results are intuitive and interpretable 2. Exhaustive approach as it finds all rules based on the confidence and support	1. Generates many uninteresting itemsets 2. Computationally and memory intensive. 3. Results in many overlapping item sets

[Datacamp ML Algorithm Selection Cheat Sheet]

Choosing the Right ML Algorithm

- Imagine you're working with a large dataset that has a mix of numeric and categorical data, and your goal is to predict a continuous outcome. Which machine learning algorithms would you consider and why?

Choosing the Right ML Algorithm

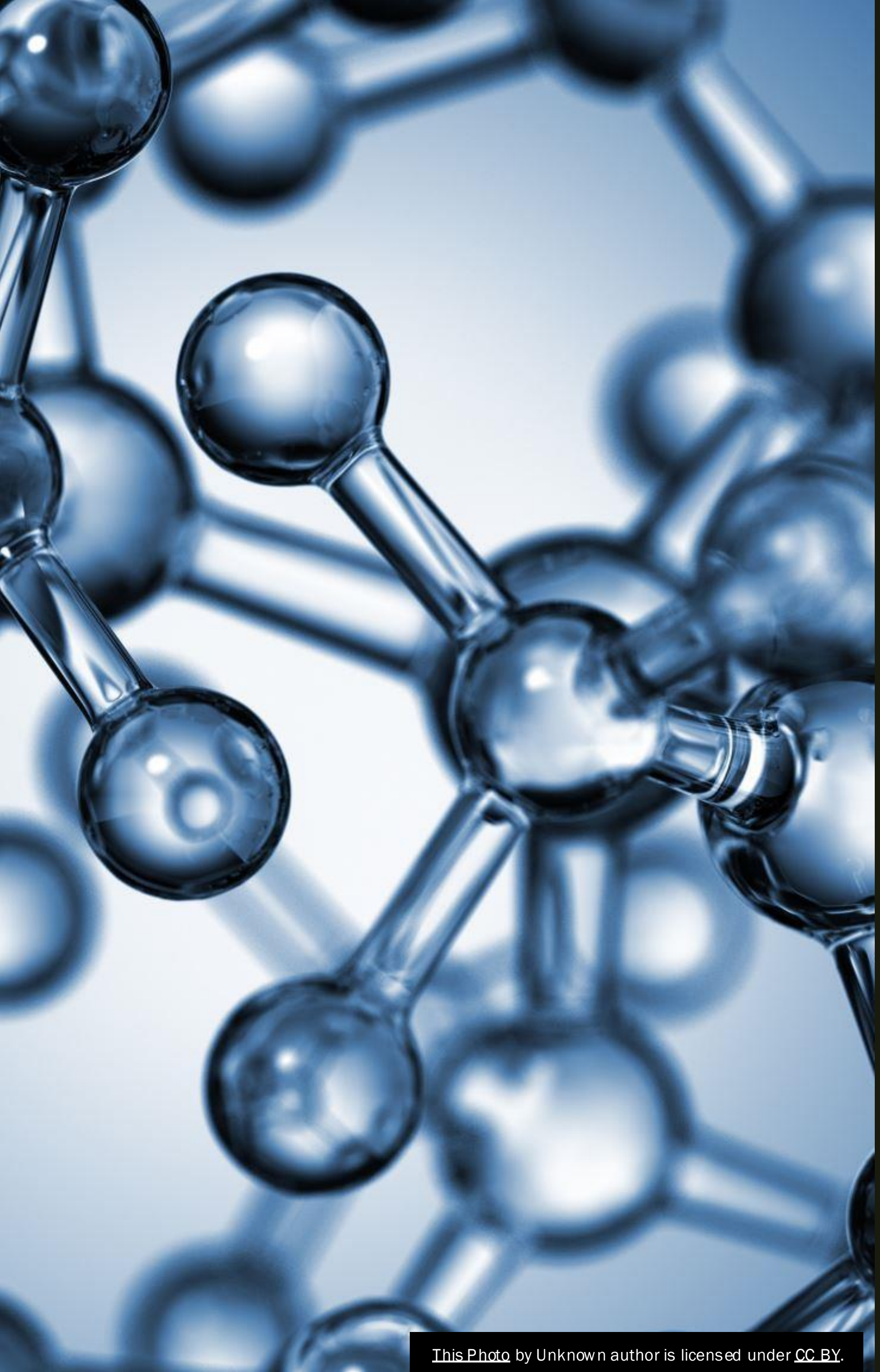
- Imagine you're working with a large dataset that has a mix of numeric and categorical data, and your goal is to predict a continuous outcome. Which machine learning algorithms would you consider and why?
- algorithms like Random Forest or Gradient Boosting Machines (GBM) are suitable as they handle mixed data types well and are good for regression tasks. They can also handle large datasets effectively.

Distributed Training

- When distributed training become necessary?

Distributed Training

- ❑ When distributed training become necessary?
- ❑ In cases where training data doesn't fit into memory
- ❑ Examples:
  - Large Language Models (GPT, LaMDA etc.)
  - Medical Images (CT Scans, MRI Images)
  - Genomic Sequences

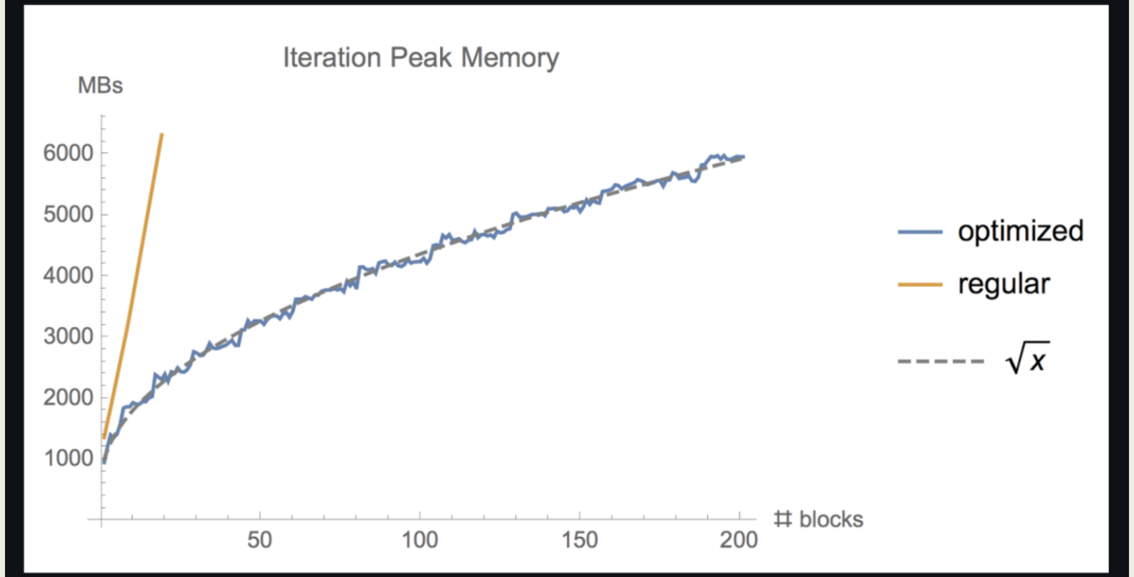


Distributed Training

- ❑ Preprocessing steps also requires parallel computation
- ❑ Example: Apache Spark, Hadoop
- ❑ Examples:
  - Large Language Models (GPT, LaMDA etc.)
  - Medical Images (CT Scans, MRI Images)
  - Genomic Sequences

Memory Optimization Methods

- Gradient Checkpointing
  - Is a technique used in training deep neural networks to manage the high memory requirements
  - Mark a subset of neural network activations as checkpoints and store them in memory after the forward pass. Checkpoint nodes are recomputed at most once and are stored in memory only until no longer required.
  - For feed-forward networks, the optimal strategy is to mark every sqrt(n)-th node as a checkpoint
  - Feed-forward models were able to fit more than 10x larger models
  - At only a 20% increase in computation time

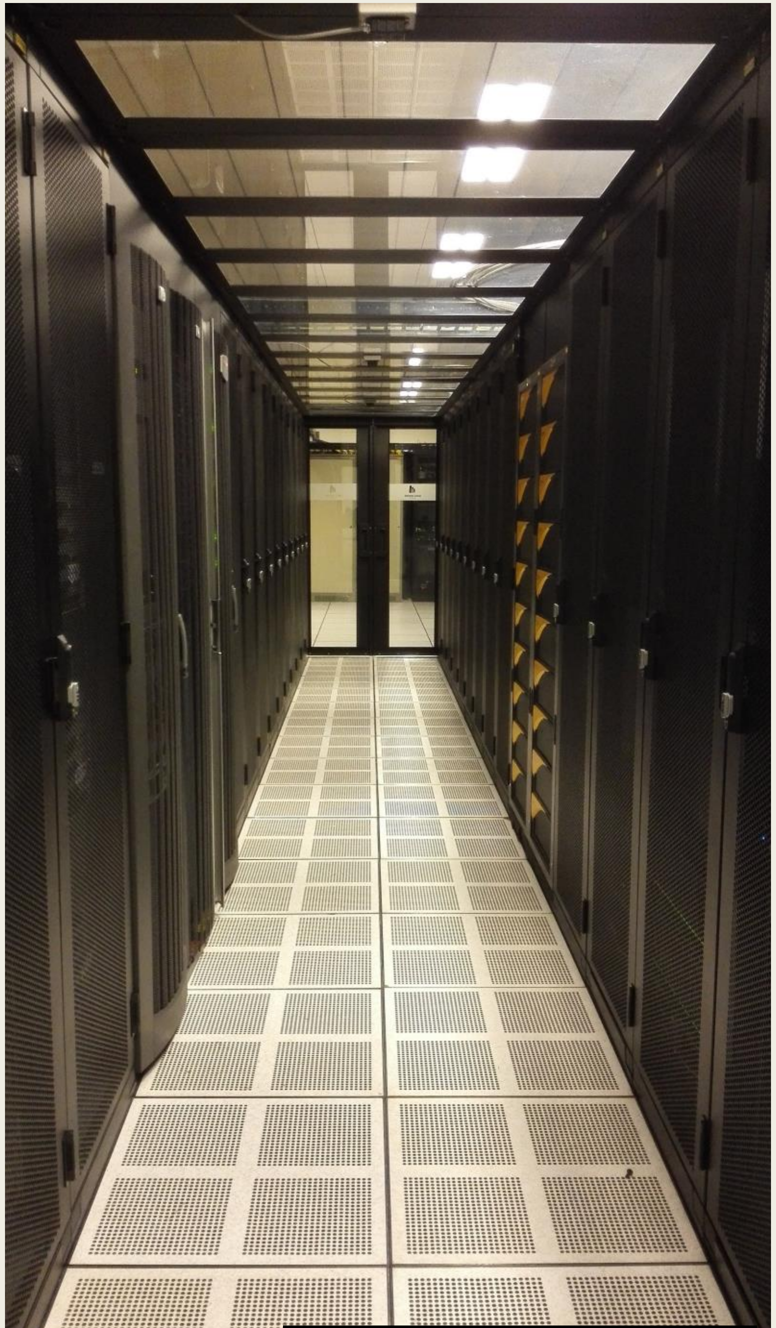


Memory used while training a ResNet model with large batch size, using the regular tf.gradients function and using our memory-optimized gradient implementation

Source https://github.com/cybertronai/gradient-checkpointing

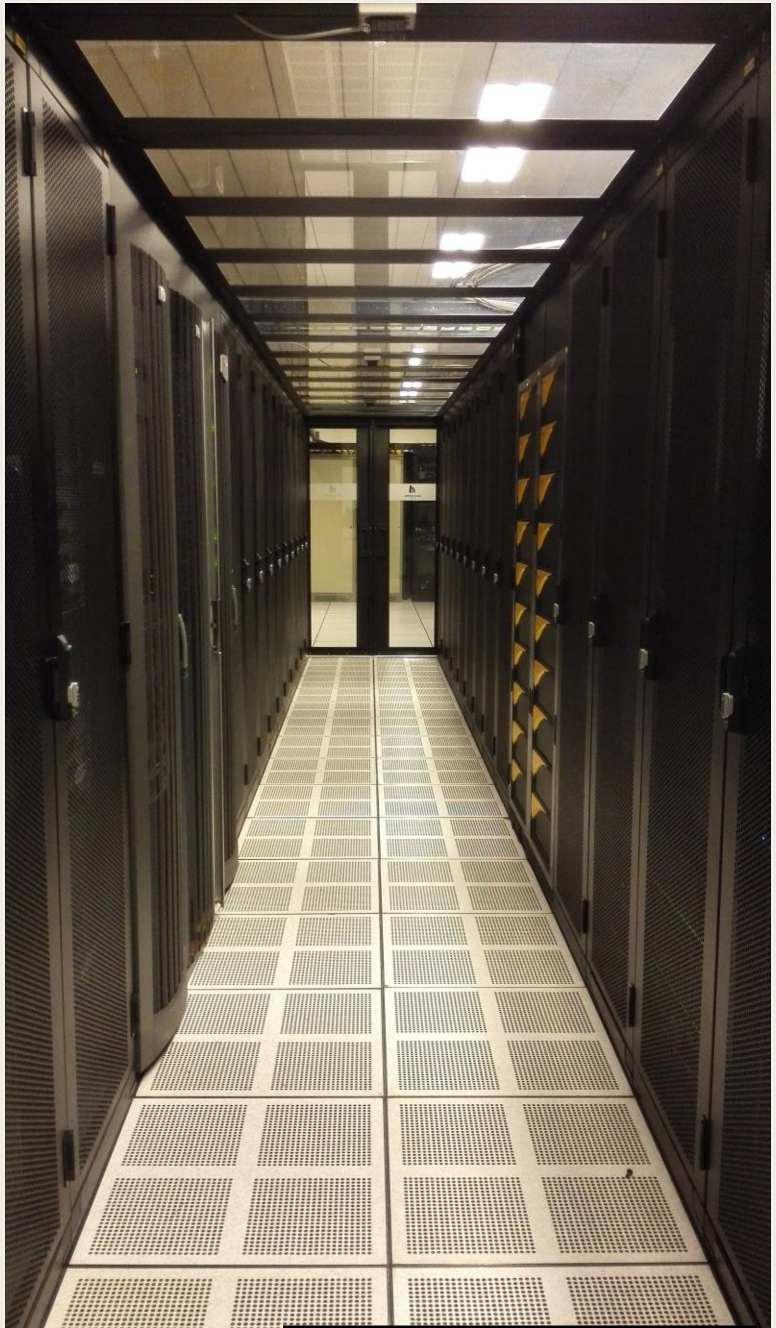
Strategies for Parallelization

- ❑ Data Parallelism
- ❑ Model Parallelism
- ❑ Pipeline Parallelism



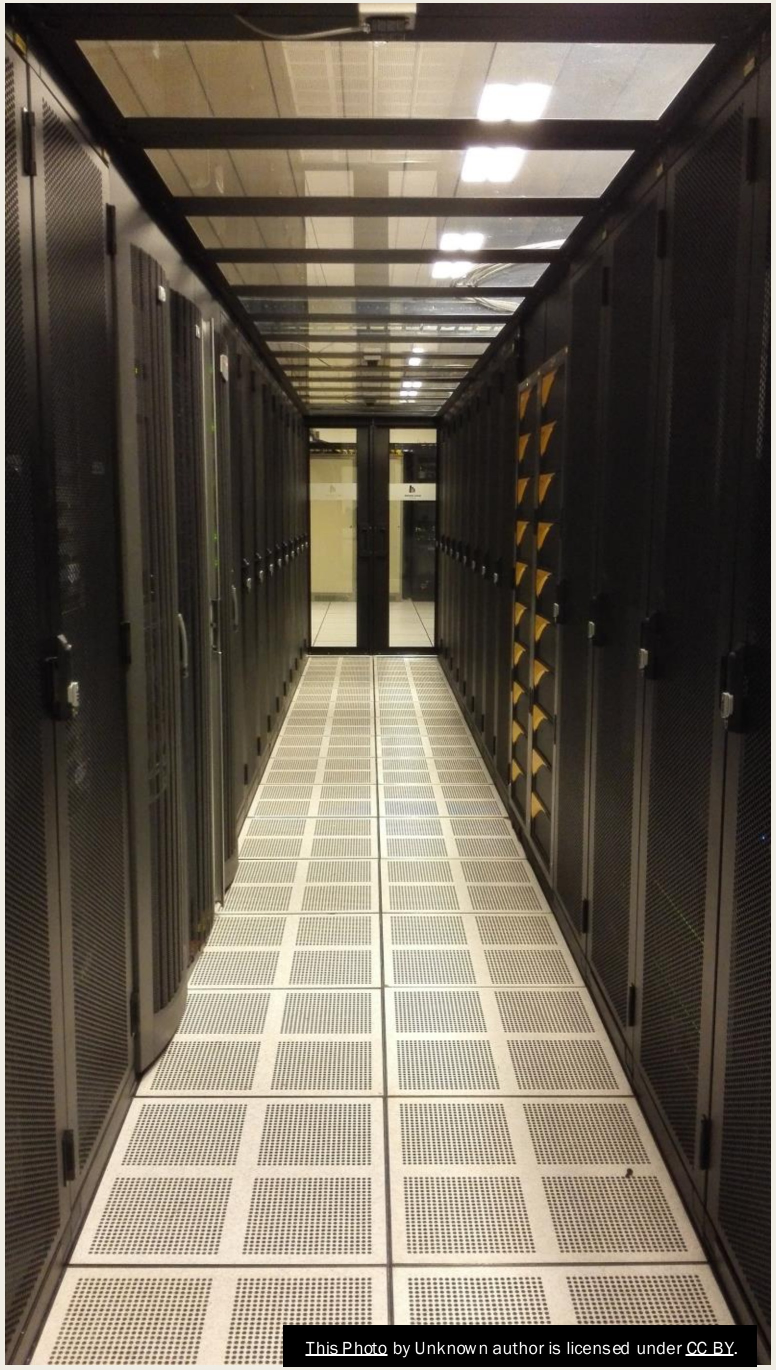
Data Parallelism

- ❑ Split the Data to multiple machines
- ❑ Train the same copy of the model on each machine
- ❑ Accumulate the gradients from multiple machines



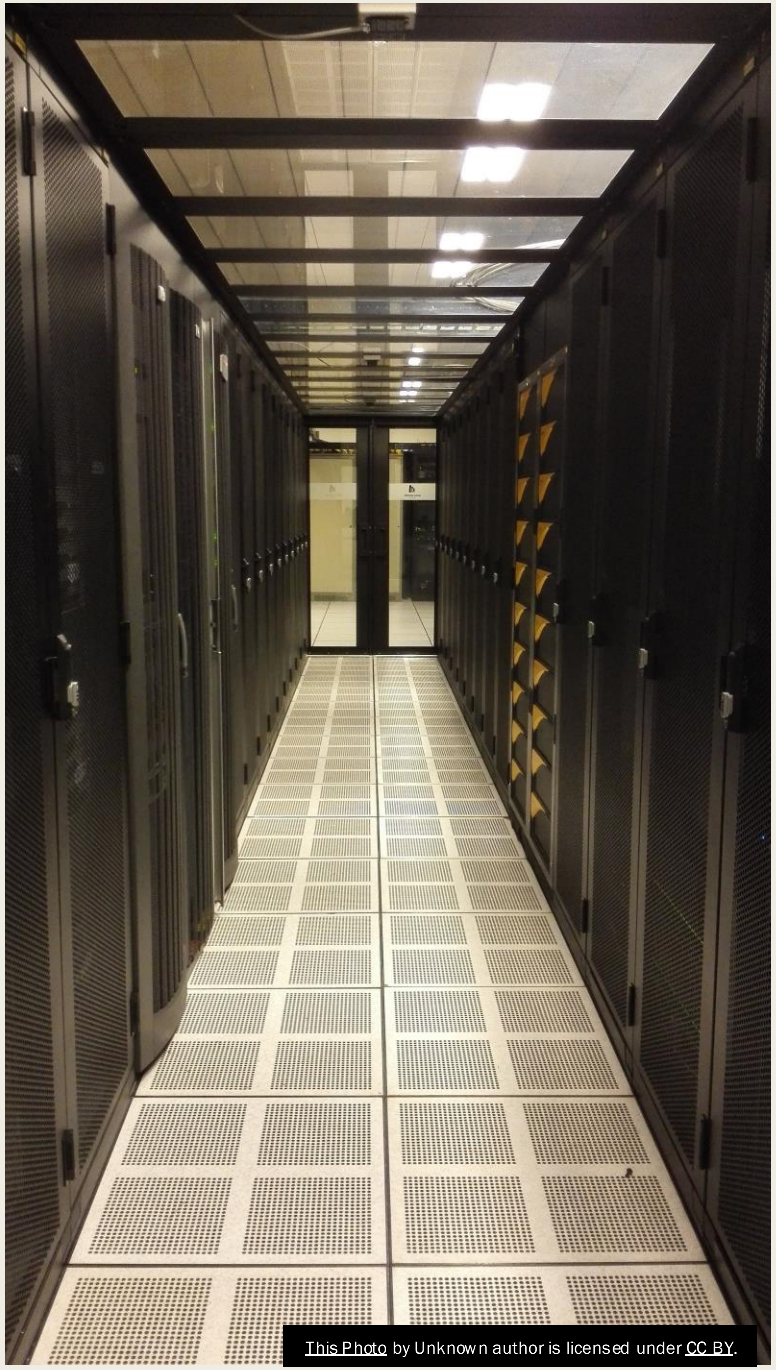
Data Parallelism

- ❑ Challenge is to accurately and efficiently accumulates gradients from different machines.
- ❑ Two modes of gathering gradients
  - *Synchronous Mode*
  - *Asynchronous Mode*



Data Parallelism

- ❑ Synchronous Mode will produce *Straggler Problem*.
- ❑ Also, it grows with the number of machines.
- ❑ Will lead to slowdown of entire system.
- ❑ Waste resources.
- ❑ Can be reduced using load balancing, dynamic allocation of resources etc.

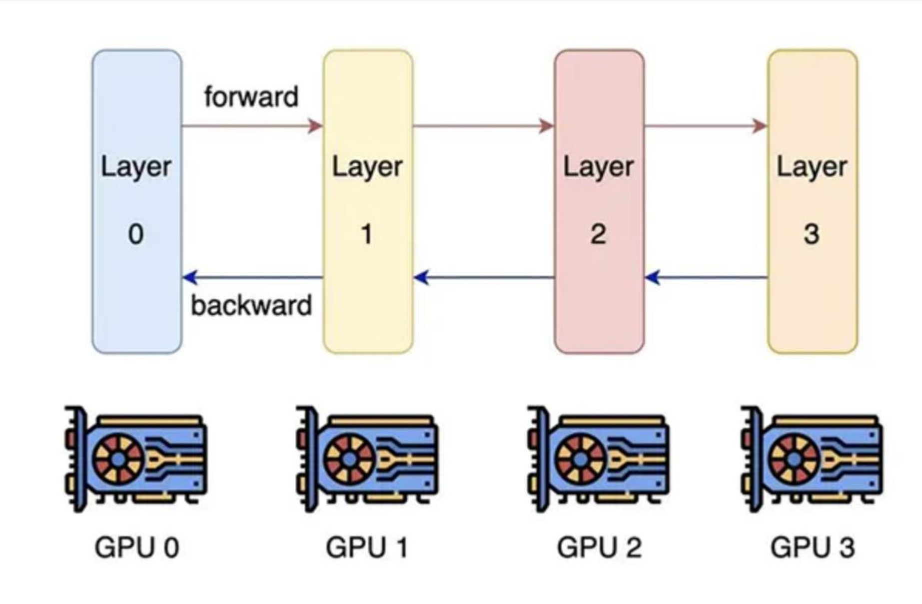


Data Parallelism

- ❑ Asynchronous Mode leads to *Gradient Staleness* problem.
- ❑ Weights changes by gradients from just one machine.
- ❑ When the number of parameters is large, gradient updates tends to be sparse.
- ❑ Gradient staleness becomes less of a problem in this scenario.

Model Parallelism

- Different components of the model are trained under different machines.



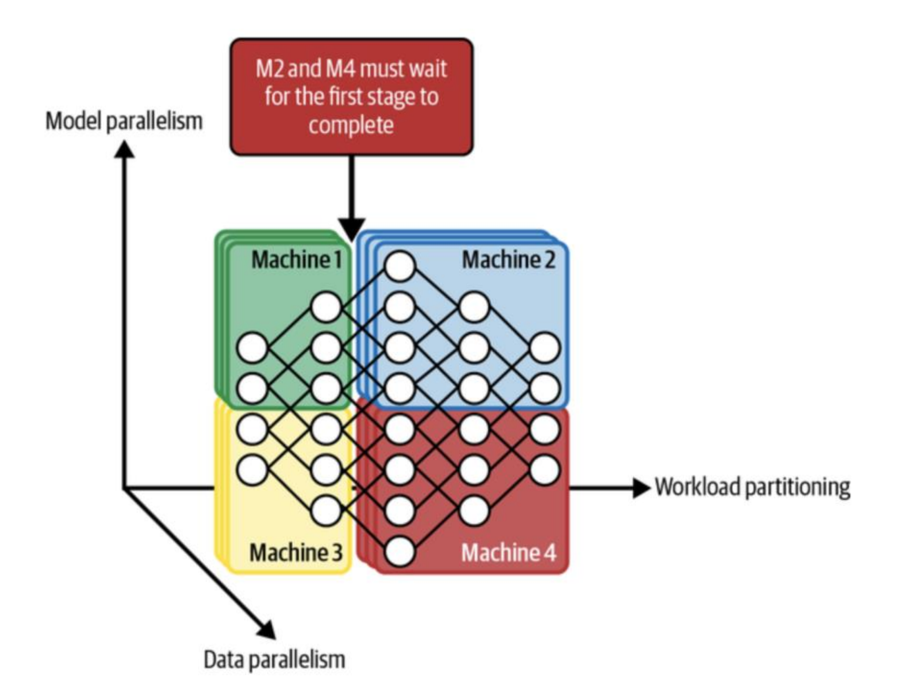


Figure 6-7. Data parallelism and model parallelism. Source: Adapted from an image by Jure Leskovec[21]

Pipeline Parallelism

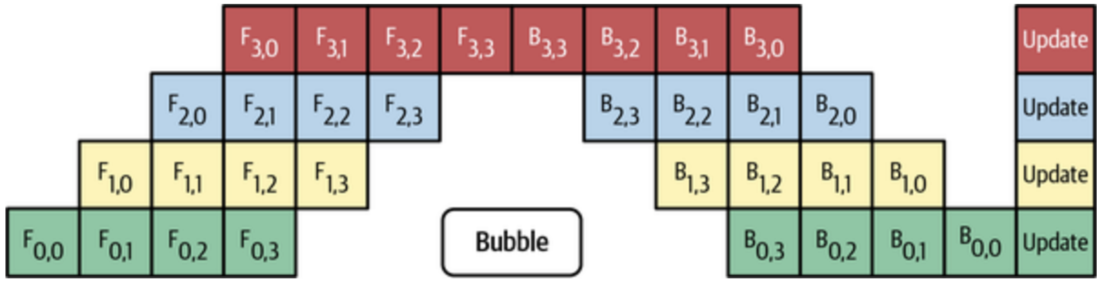


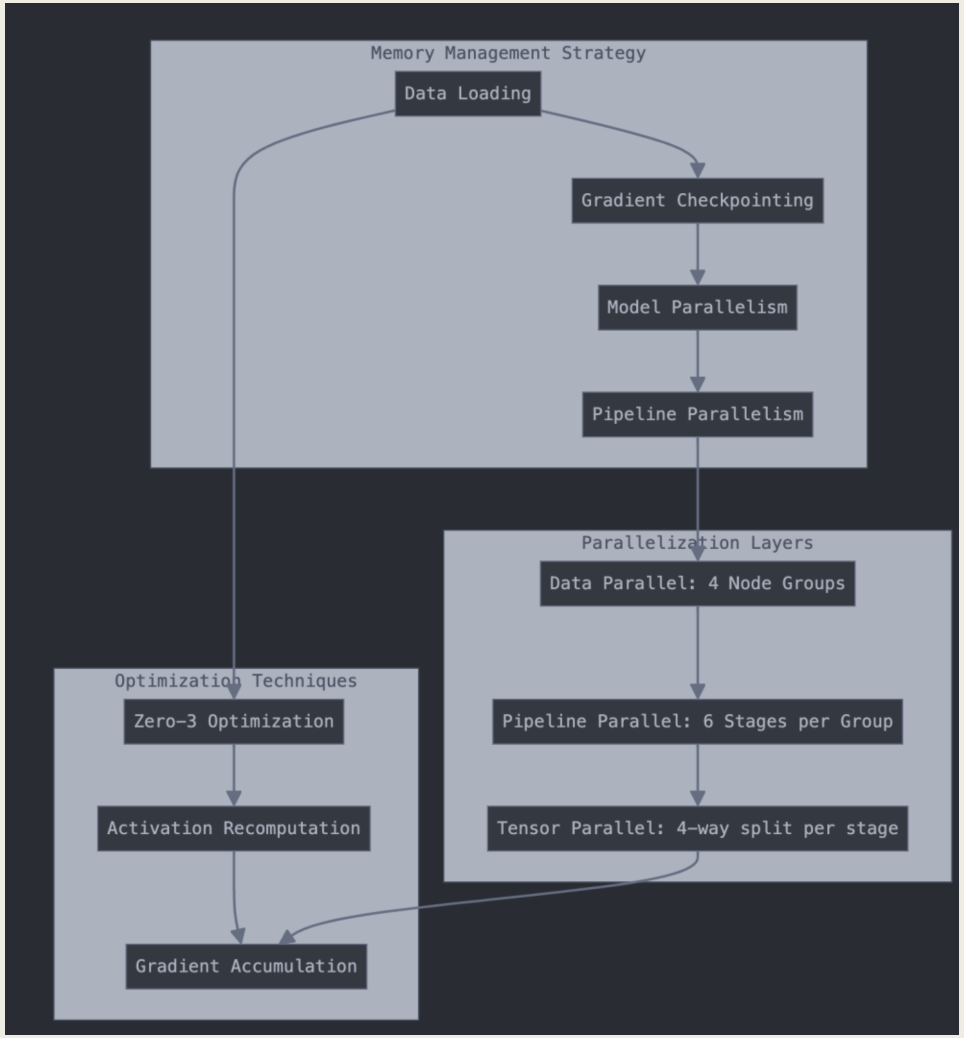
Figure 6-8. Pipeline parallelism for a neural network on four machines; each machine runs both the forward pass (F) and the backward pass (B) for one component of the neural network. Source: Adapted from an image by Huang et al.[22]

- ❑ Break the computation of each machine to multiple parts.
- ❑ When machine 1 completes its first part, pass the results to machine 2
- ❑ Machine 1 then start computing its second part
- ❑ Figure: 4 layers of a NN computed using 4 machines

Pipeline Parallelism

Use case of Training Llama 2 70B Model

Goolge Colab Notebook



Distributed Model Training with PyTorch

- ❑ Two Approaches:
  - o Distributed Data Parallel (DDP)
  - o Fully Sharded Data Parallel (FSDP)
- ❑ In DDP training, each process/worker owns a replica of the model and processes a batch of data
- ❑ Model weights and optimizer states are replicated across all workers
- ❑ Uses all-reduce to sum up gradients over different workers

Distributed Model Training with PyTorch

- ❑ Two Approaches:
  - o Distributed Data Parallel (DDP)
  - o Fully Sharded Data Parallel (FSDP)
- ❑ In FSDP training model parameters, optimizer states and gradients

  Are sharded across GPUs
- ❑ This makes training of very large models feasible

Distributed Model Training with PyTorch

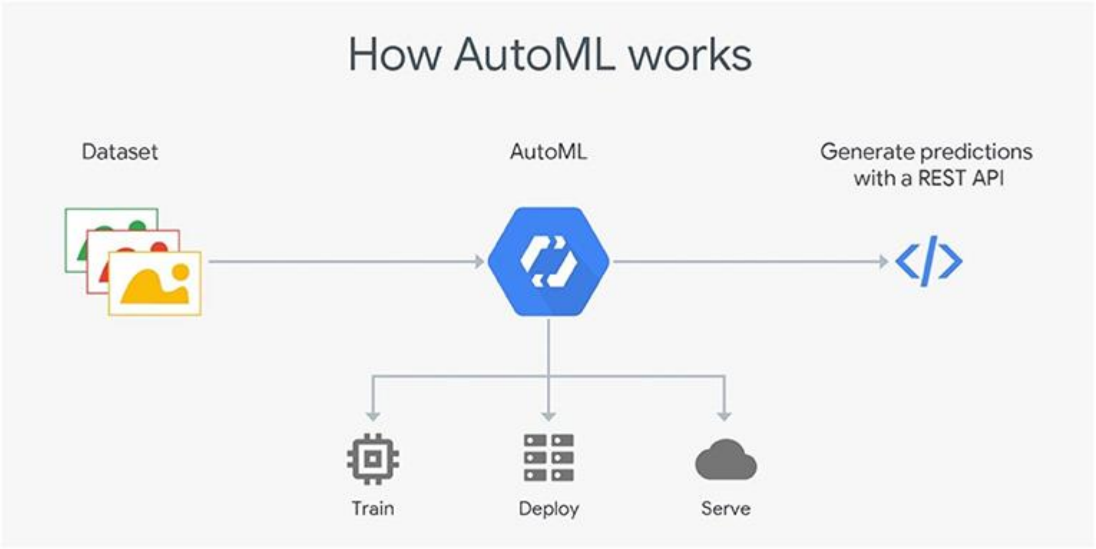
- Exercise:

  Use the code provided with the lecture notes to train a Neural Network Classification model using FSDP on the GPU Cluster

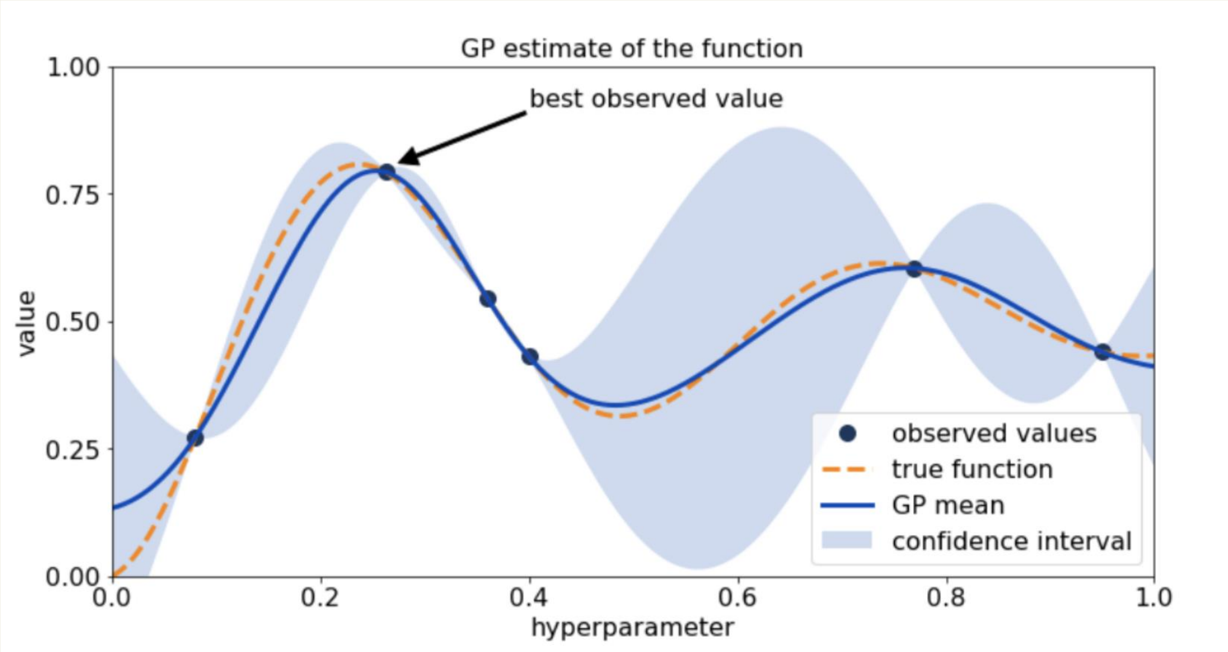
- Reference:
  - FSDP Blog from Meta
  - Fair Scale Open-Source Library
  - FSDP Tutorial - PyTorch

Auto ML

- ❑ Refers to the process of automating the end-to-end process of applying ML to real-world problems.
- ❑ Two flavors:
  - ❑ Soft Auto ML: Hyperparameter Tuning
  - ❑ Hard AutoML: Architecture search and learned optimizer



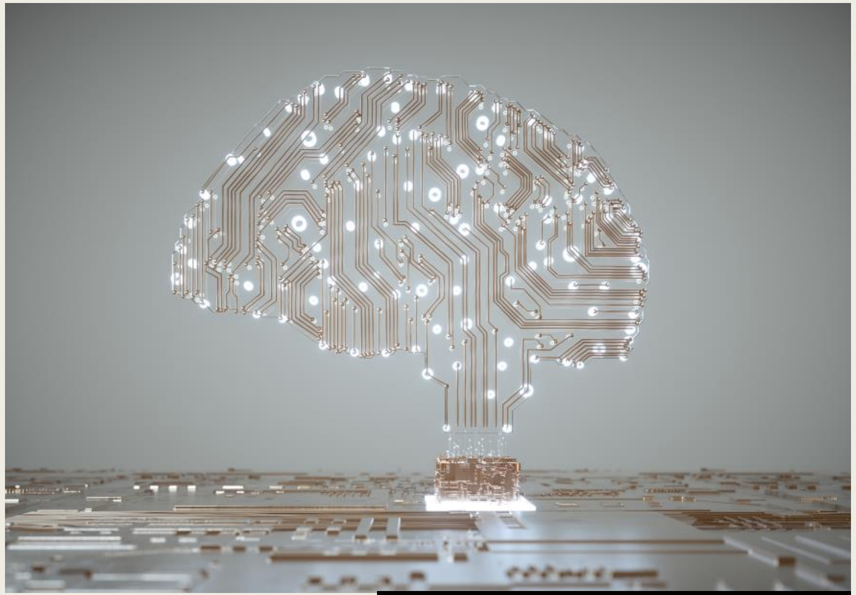
Soft Auto ML – Hyper Parameter Tuning



Source: CERN

https://cds.cern.ch/record/2702355

- ❑ Popular ML Frameworks comes with built in tuners
  - Auto-sklearn
  - Keras Tuner
- ❑ Popular methods
  - Grid search
  - Random search
  - Bayesian optimization



Hard Auto ML: Neural Architecture Search (NAS)

Consists of 3 components

- ❑ A Search Space
  - Library of NN components (e.g., 3 X 3 convolutions, pooling layers, skip connections).
- ❑ A Search Strategy
  - Exploration – Try novel architectures
  - Exploitation – Tweak proven architectures
- ❑ A Performance Estimation Strategy
  - Measures how good the performance is using k-fold cross validation

Hard Auto ML: Neural Architecture Search (NAS)

- Reinforcement Learning-Based NAS:
  - A Controller (usually an RNN or a Transformer model) acts as an Agent.
  - Suggests a model description as a "string".
  - This model is built and its performance is evaluated.
  - The value of the performance metric is given back to controller as a reward.
  - After receiving the reward, the controller suggests a new model.
  - Repeating this several times results in a highly optimized model description from the controller (optimizing long term cumulative rewards)
  - Example: NASNet - Beat human designed models on ImageNet

Hard Auto ML: Neural Architecture Search (NAS)

- Evolutionary Algorithms: Applies principles of biological evolution, such as mutation, crossover, and selection, to evolve network architectures over time.
  - Start with a population of random model architectures.
  - Kill the models having performance lower than a threshold.
  - Mutate (tweak) the models having higher performance.
  - Repeat this process.
  - Example: **AmoebaNet.** It proved that "evolution" could find high-performing architectures that human intuition might never have considered

Hard Auto ML: Neural Architecture Search (NAS)

- Differentiable/Gradient-Based NAS (DARTS):
  - Instead of treating the search as a series of separate guesses, it turns the architecture into a single, massive mathematical equation.
  - Creates a "Supernet" where every possible path exists at once with different weights.
  - Using gradient descent, the model slowly "turns down the volume" on bad paths and "turns up the volume" on good paths.
  - Reduced the search time from thousands of GPU-hours to just **a few hours**

In a standard neural network, an edge between two layers is a fixed operation (like a Convolution). In DARTS, every edge is a **weighted sum** of all candidate operations (3×3 conv, 5×5 conv, max-pool, etc.).

For an input x, the output of a connection is calculated as:

$$\bar{o}(x) = \sum_{i \in \text{Candidates}} \frac{\exp(\alpha_i)}{\sum_j \exp(\alpha_j)} o_i(x)$$

- **$o_i(x)$:** The actual mathematical operation (the "candidate").
- **α_i:** A learnable **architecture parameter** (the "strength" of that operation).
- **Softmax:** We use a softmax function so that the weights of all candidates always add up to 100%.

[What is Neural Architectural Search]

Summary of Today's Learning

- ❑ Approaches for choosing the right algorithm for your ML problem.
- ❑ Methods for Distributed Training of ML Models.
- ❑ Introduction to Auto ML.

Useful Links

- Distributed full fine-tuning of Llama2 on Kubernetes
- Fine-tune Llama 2 with Limited Resources
- TinyLlama/TinyLlama-1.1B-Chat-v1.0
- mistralai/Mixtral-8x7B-Instruct-v0.1
- ADVANCED MODEL TRAINING WITH FULLY SHARDED DATA PARALLEL (FSDP)