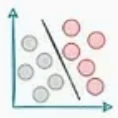
Pandas ↔ Polars ↔ SQL ↔ PySpark



blog.DailyDoseofDS.com

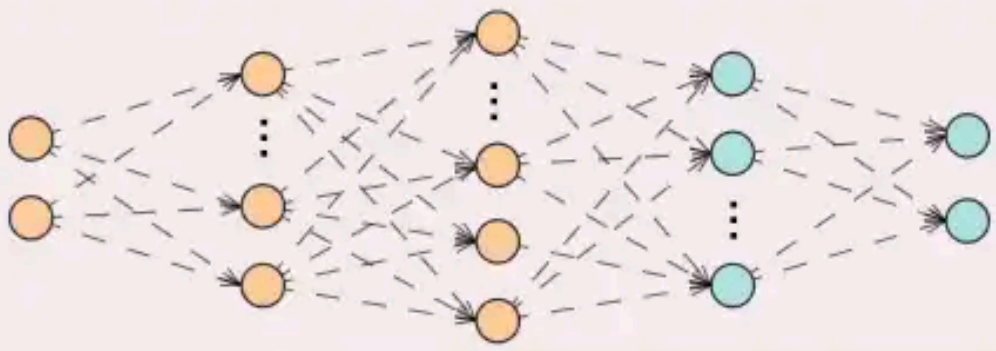
Operation	Pandas	Polars	SQL	PySpark
Import	import pandas as pd	import polars as pl	-	from pyspark.sql import SparkSession spark = SparkSession.builder.appName("ABCD")
Read CSV	df = pd.read_csv(file)	df = pl.read_csv(file)	LOAD DATA INFILE 'data.csv' INTO TABLE table FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 ROWS;	df = spark.read .csv("data.csv")
Print first 10 (or k) rows	df.head(10)	df.head(10)	SELECT * FROM table LIMIT 10;	df.show(10)
Dimensions	df.shape	df.shape	SELECT count(*) FROM table;	df.count()
			SELECT count(*) FROM INFORMATION_SCHEMA.COLUMNS where TABLE_NAME = 'table';	len(df.columns)
Datatype	df.dtypes	df.dtypes	DESCRIBE table;	df.printSchema()
Select column(s)	df[["col1", "col2"]]	df[["col1", "col2"]]	SELECT column FROM table;	df.select("col1", "col2")
Filter Data	df[df.column > 10]	df[df.column > 10]	SELECT * FROM table where column>10;	df.filter(df["column"]>10)
		df.filter(pl.col("column") > 10)		
Sort	df.sort_values("column")	df.sort("column")	SELECT * FROM table ORDER BY column;	df.orderBy("column")
Fill NaN	df.column.fillna(0)	df.column.fill_nan(0)	UPDATE table SET column=0 WHERE column IS NULL;	df.na.fill(0)
Join	pd.merge(df1, df2, on ="col", how="inner")	df1.join(df2, on="col", how="inner")	SELECT * FROM table1 JOIN table2 ON (table1.col = table2.col);	df1.join(df2, on="col", how="inner")
Concatenate	pd.concat((df1, df2))	pl.concat((df1, df2))	SELECT * FROM table1 UNION ALL table2;	df1.union(df2)
Group	df.groupby("column"). agg_col.mean()	df.groupby("column"). agg(pl.mean("agg_col"))	SELECT column, avg(agg_col) FROM table GROUP BY column;	df.groupBy("column"). agg(avg("agg_col"))
Unique values	df.column.unique()	df.column.unique()	SELECT DISTINCT column FROM table;	df.select("column"). distinct()
Rename column	df.rename(columns = {"old_name": "new_name"})	df.rename(mapping = {"old_name": "new_name"})	ALTER TABLE table RENAME COLUMN old_name TO new_name;	df.withColumnsRenamed({"old_name": "new_name"})
Delete column	df.drop(columns = ["column"])	df.drop(name = ["column"])	ALTER TABLE table DROP COLUMN column;	df.drop("col1", "col2")

Next...

4 Strategies for Multi-GPU Training

blog.DailyDoseofDS.com

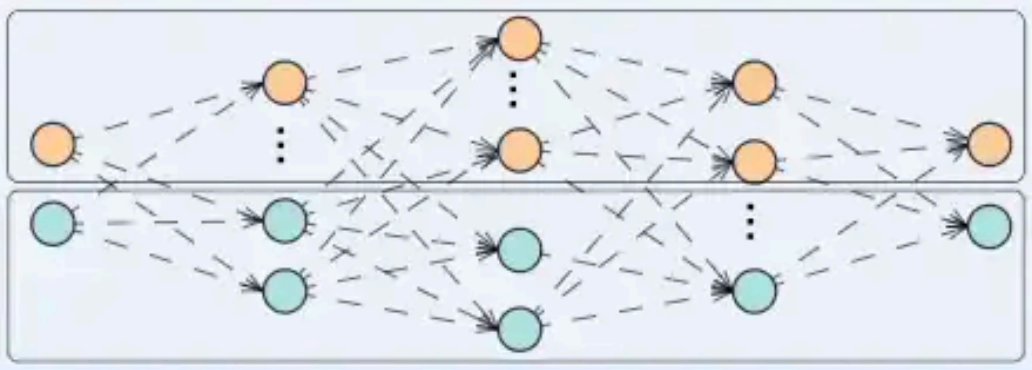
Model parallelism



Layer on 1st GPU

Layer on 2nd GPU

Tensor parallelism



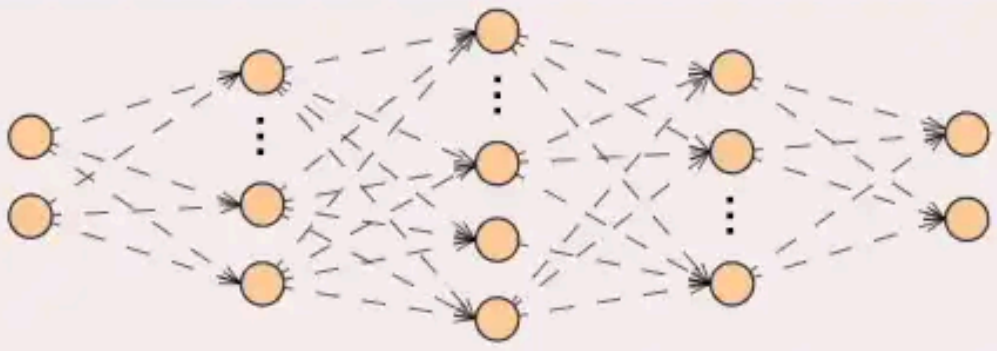
Neurons on 1st GPU

Neurons on 2nd GPU

Data parallelism

Data subset #1

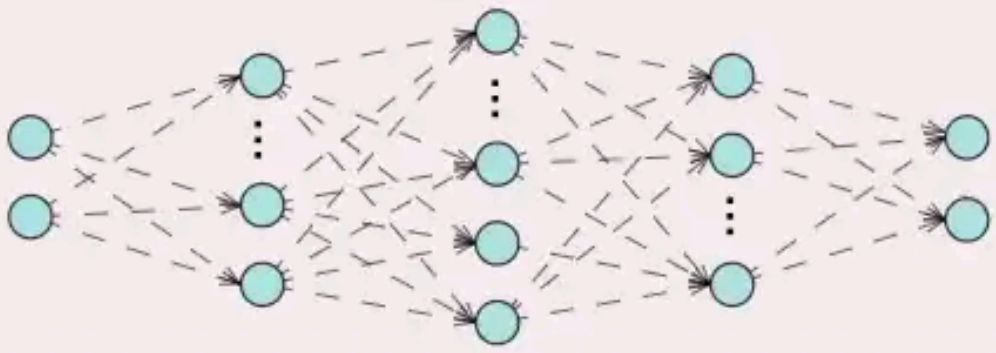
x1	x2	x3
...	...	...
...	...	...



Model copy on 1st GPU

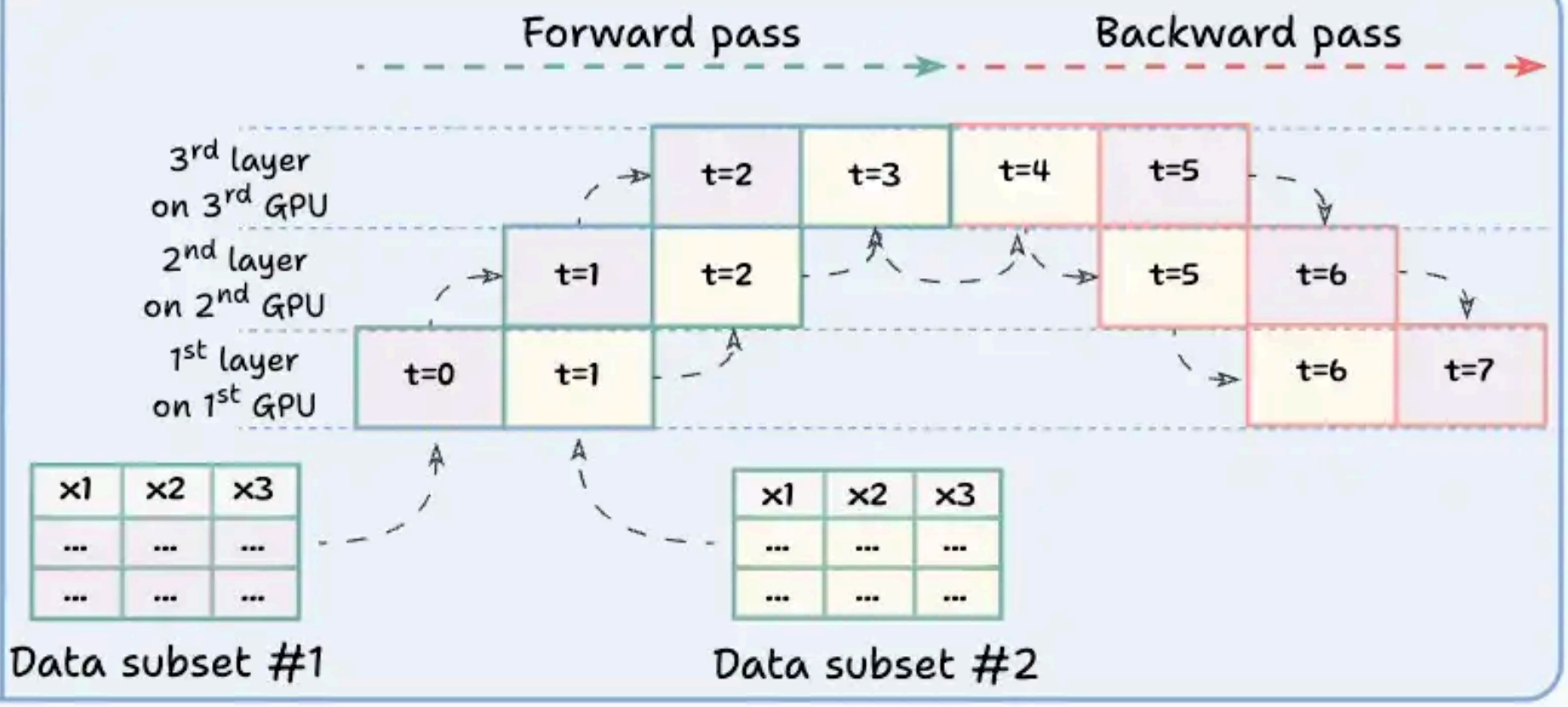
Data subset #2

x1	x2	x3
...	...	...
...	...	...



Model copy on 2nd GPU

Pipeline parallelism

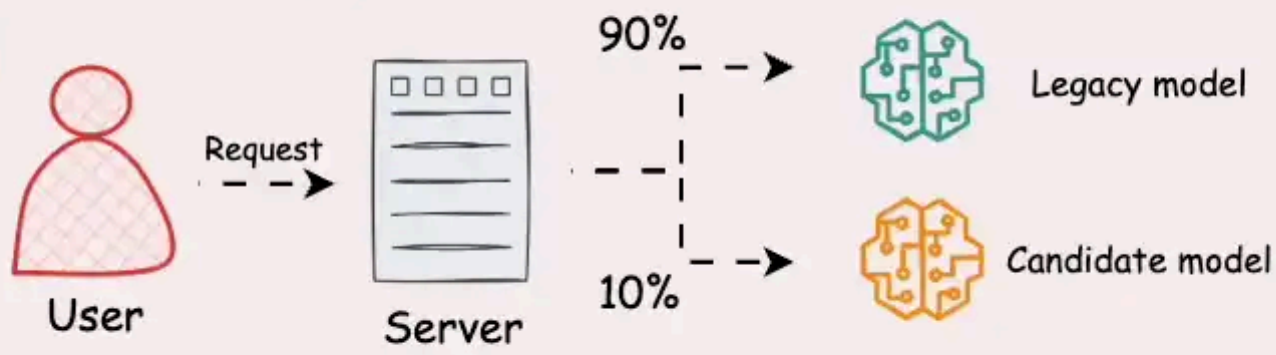


Next...

4 Ways to Test ML Models in Production

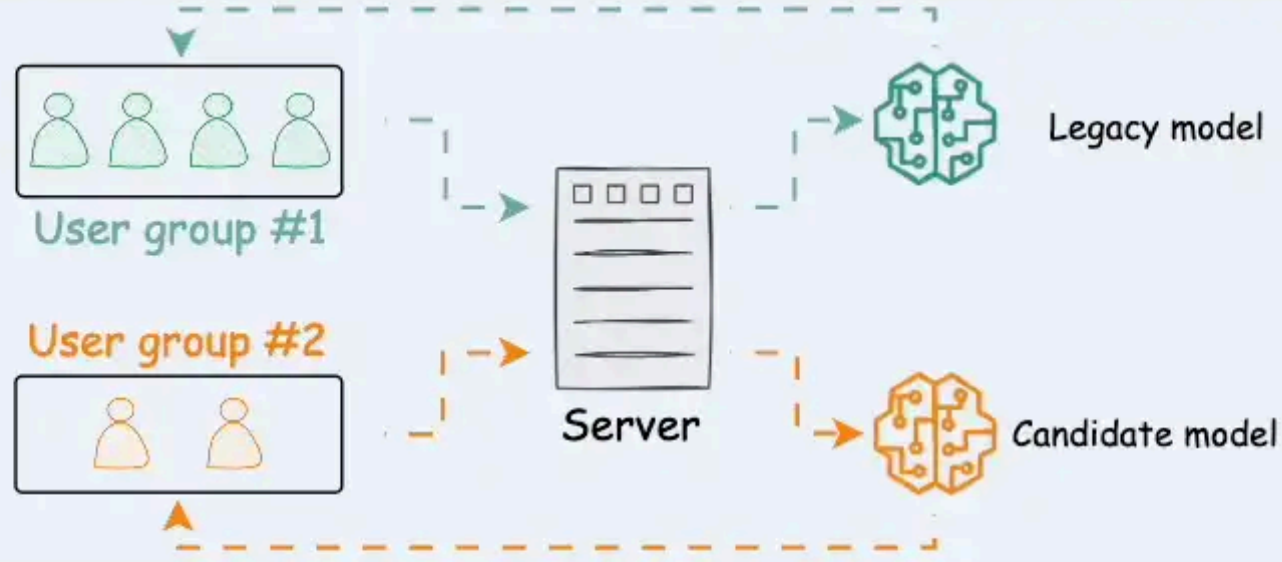
blog.DailyDoseofDS.com

1) A/B Testing



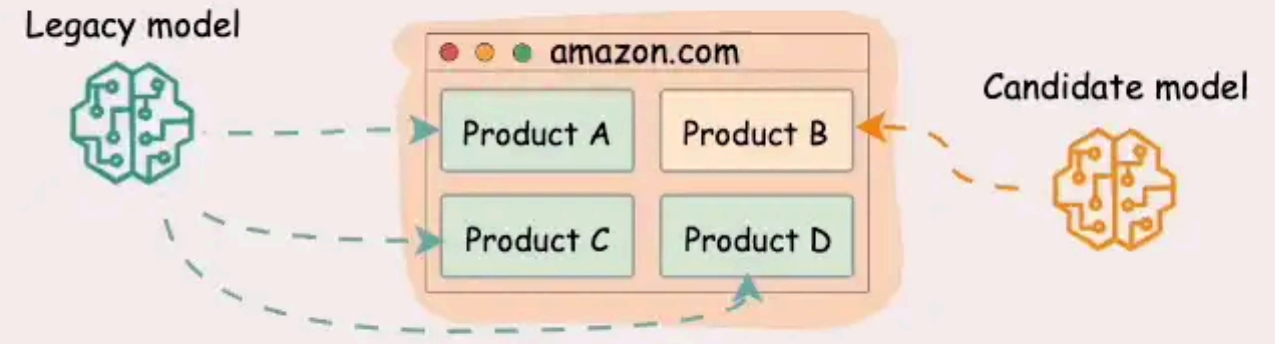
Send some requests to the candidate model.

2) Canary Testing



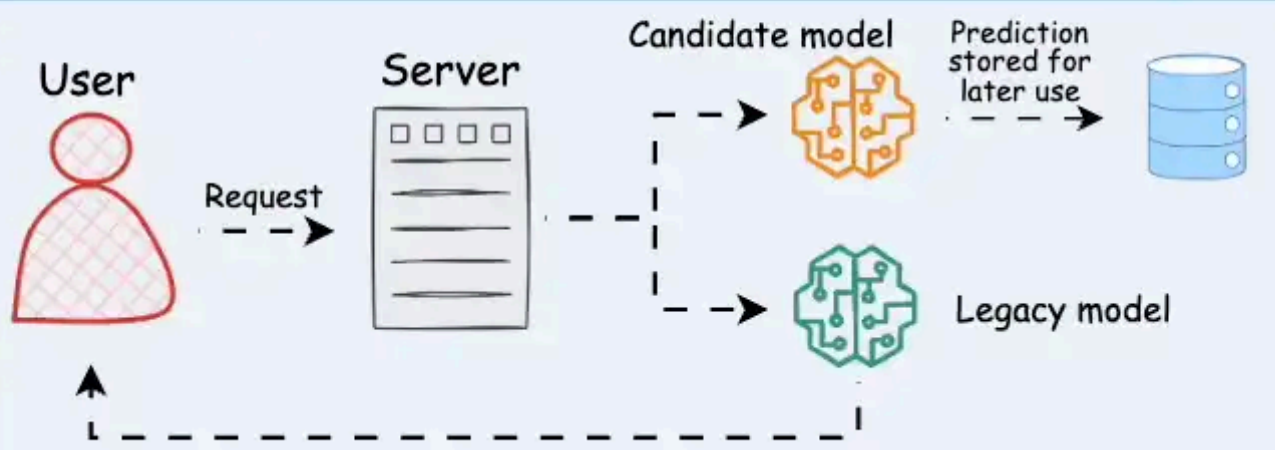
Release the candidate model only to a few users.

3) Interleaved Testing



Use both models for predictions.

4) Shadow Testing



Store the predictions from candidate model for later use/ inspection.

Next...

15 Ways to Optimize Neural Network Training

blog.DailyDoseofDS.com

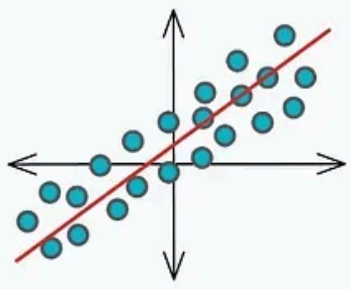
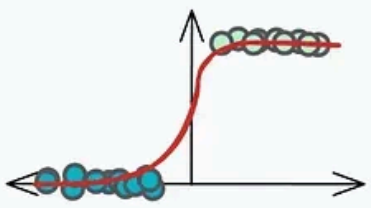
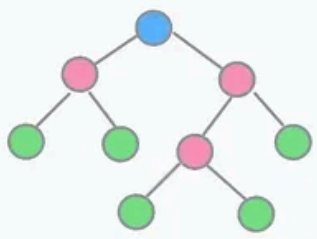
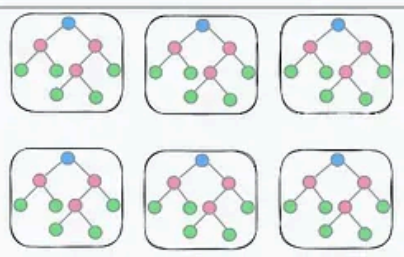
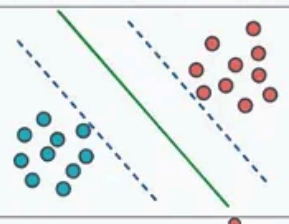
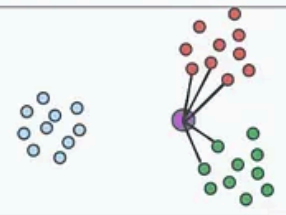
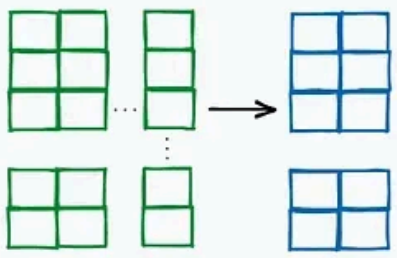
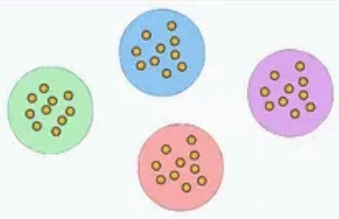
- ☑ Use efficient optimizers—AdamW, Adam, etc.
- ☑ Utilize hardware accelerators (GPUs/TPUs).
- ☑ Max out the batch size.
- ☑ Use Bayesian Optimization if hyperparameter search space is big.
- ☑ Set max_workers in the DataLoader.
- ☑ Set pin_memory in DataLoader.
- ☑ Use mixed precision training.
- ☑ Use He or Xavier initialization for faster convergence (usually helps).
- ☑ Use activation checkpointing to optimize memory (run-time will go up).
- ☑ Utilize multi-GPU training through Model/Data/Pipeline/Tensor parallelism.
- ☑ For large models, use DeepSpeed, FSDP, YaFSDP, etc.
- ☑ Normalize data after transferring to GPU (for numerical data, like pixels).
- ☑ Use gradient accumulation (may have marginal improvement at times).
- ☑ Always use DistributedDataParallel, not DataParallel.
- ☑ torch.rand(2, 2, device = ...) creates a tensors directly on the GPU.
- ☒ torch.rand(2,2).cuda() first creates on the CPU, then transfers to GPU.

Always profile your code to identify and to identify eliminate performance bottlenecks.

Next...

Time Complexity of 10 Most Popular ML Algorithms

blog.DailyDoseofDS.com

		Training	*Inference*
	Linear Regression (OLS)	$O(nm^2 + m^3)$	$O(m)$
	Linear Regression (SGD)	$O(n_{epoch}nm)$	$O(m)$
	Logistic Regression (Binary)	$O(n_{epoch}nm)$	$O(m)$
	Logistic Regression (Multiclass OvR)	$O(n_{epoch}nmc)$	$O(mc)$
	Decision Tree	$O(n \cdot log(n) \cdot m)$ $O(n^2 \cdot m)$* **Worst case**	$O(d_{tree})$
	Random Forest Classifier	$O(n_{trees} \cdot n \cdot log(n) \cdot m)$	$O(n_{trees} \cdot d_{tree})$
	Support Vector Machines (SVMs)	$O(n^2m + n^3)$	$O(m \cdot n_{SV})$
	k-Nearest Neighbors	—	$O(nm)$
$P(B\|A) = \frac{P(\mathrm{B} \cap A)}{P(A)}$	Naive Bayes	$O(nm)$	$O(mc)$
	Principal Component Analysis (PCA)	$O(nm^2 + m^3)$	—
	t-SNE	$O(n^2m)$	—
	KMeans Clustering	$O(iknm)$	??

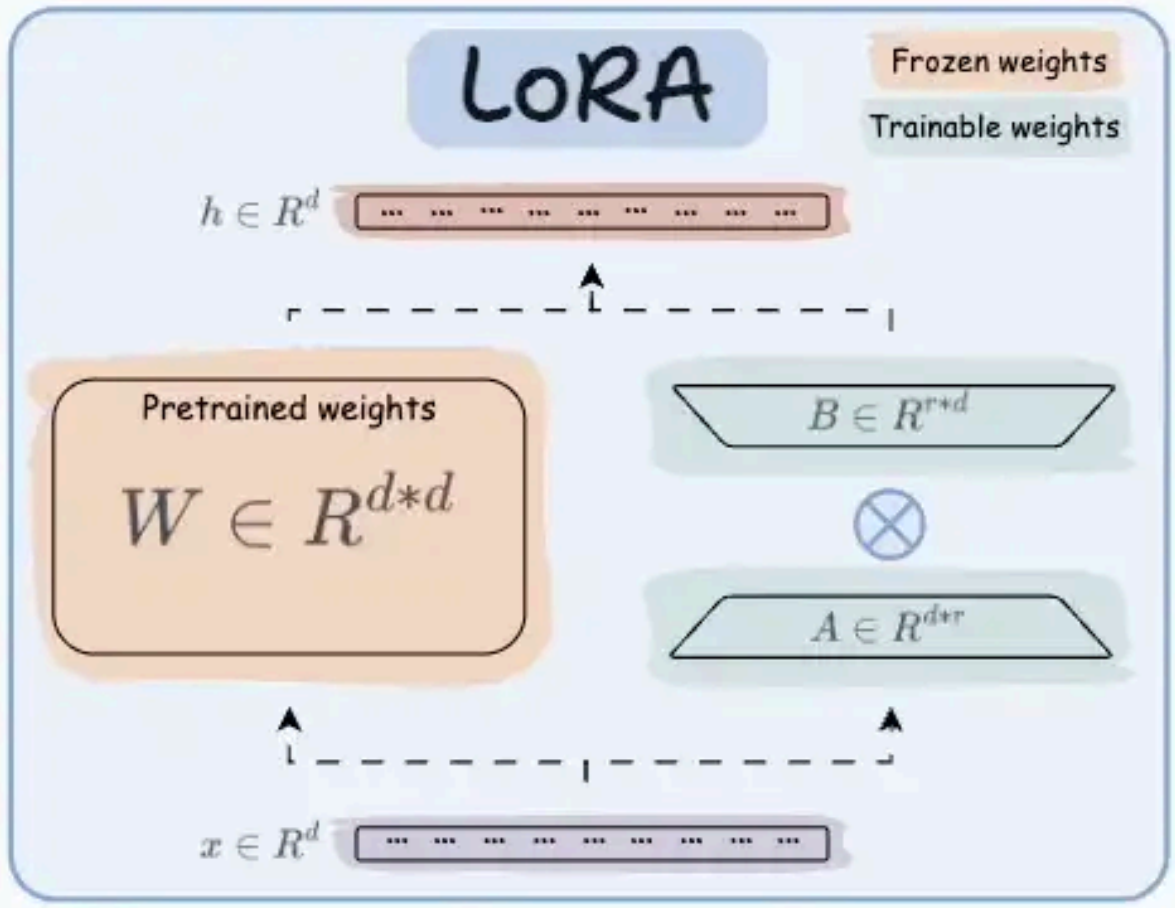
n: samples | m: dimensions | n_{epoch}: epochs | c: classes | d_{tree}: depth

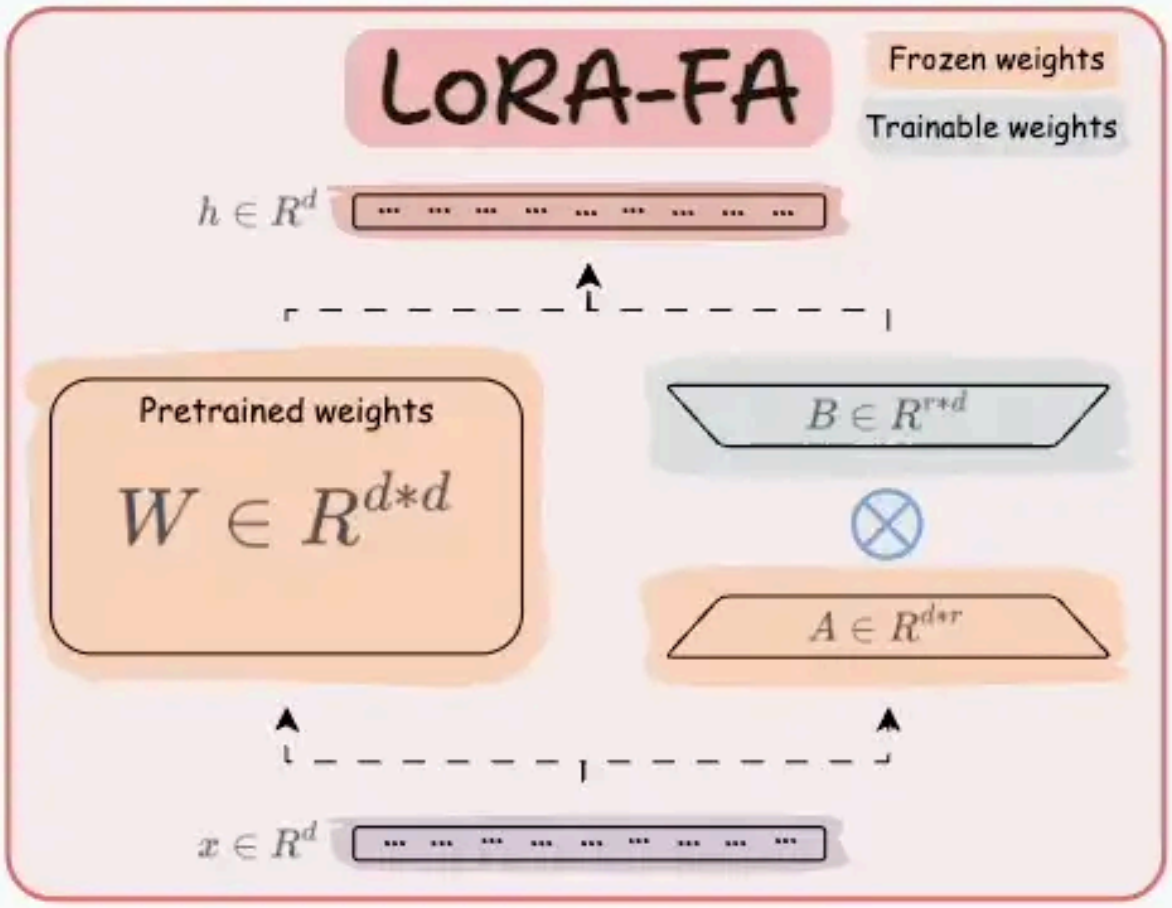
n_{SV}: Support vectors | k: clusters | i: iterations

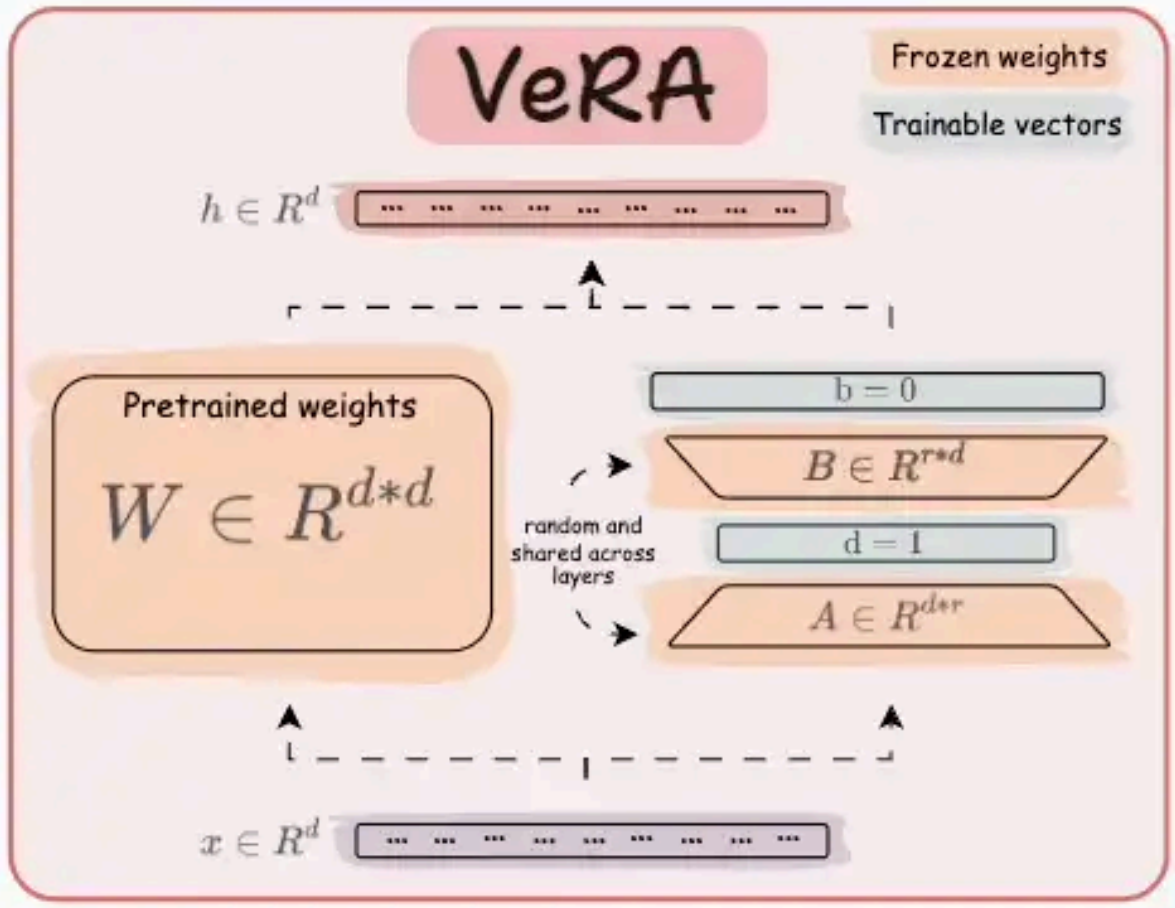
Next...

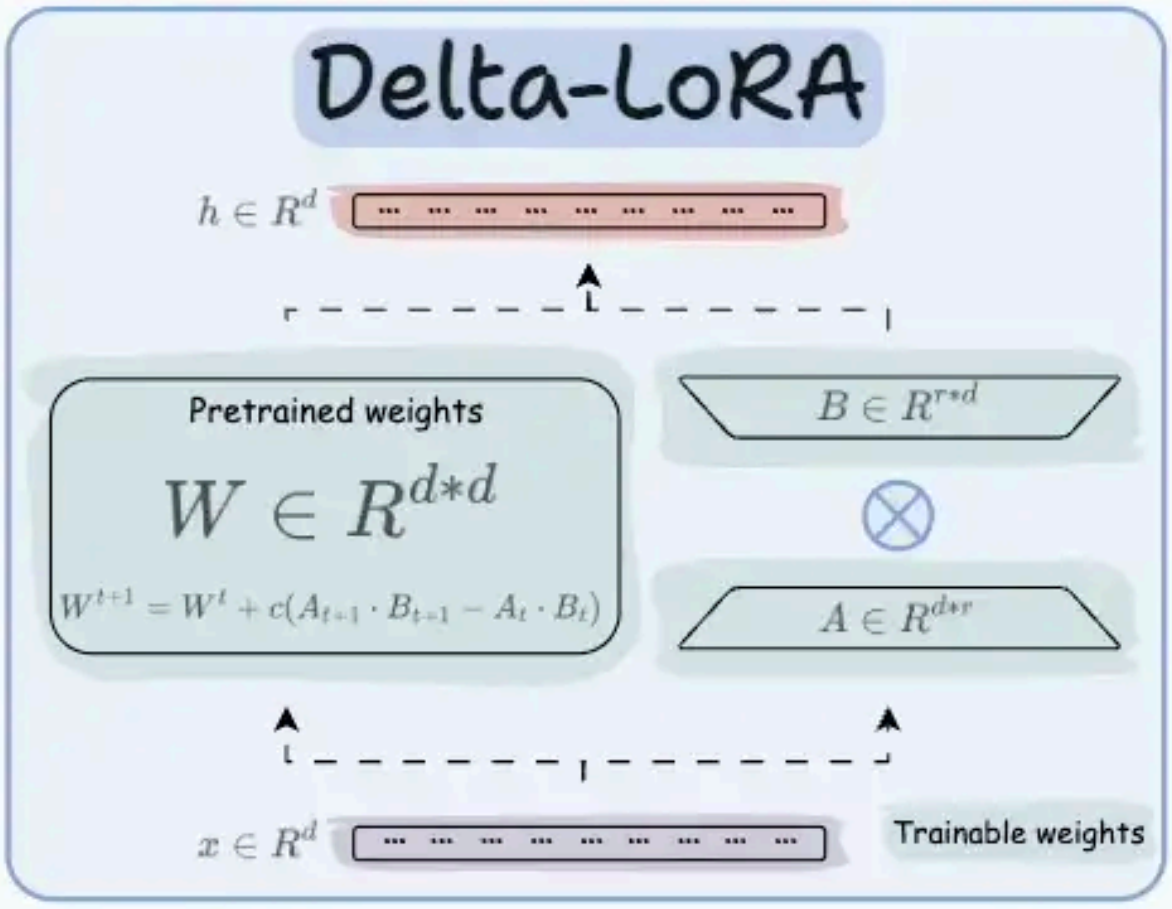
5 Techniques to fine-tune LLMs

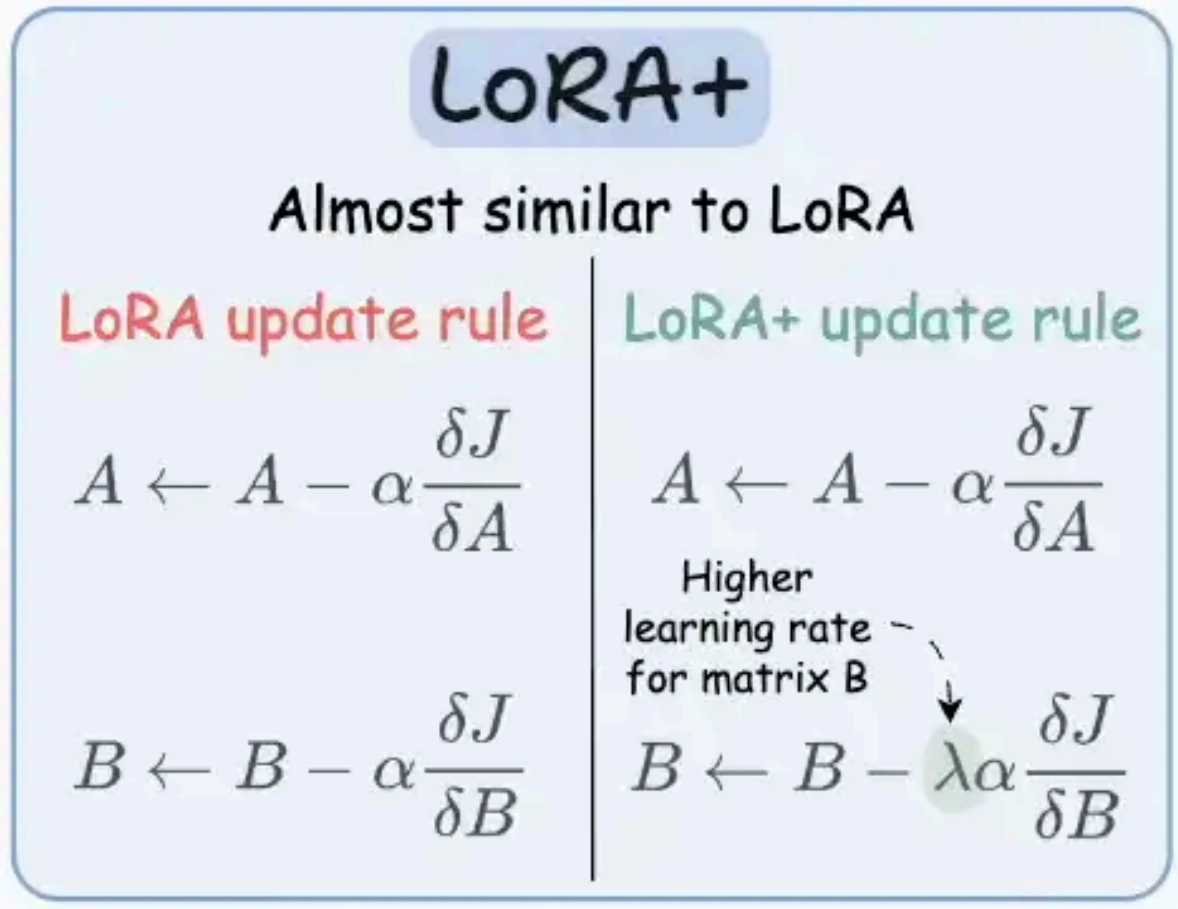
blog.DailyDoseofDS.com





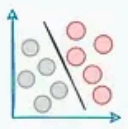






Next...

40 NumPy Methods That Data Scientists Use 95% of the Time



blog.DailyDoseofDS.com

NumPy Array Creation Methods

Method	Description
np.array(<list>)	NumPy array from Python list
np.array(<list-of-lists>)	NumPy array from list of lists
np.array(<pandas-series>)	NumPy array from PD Series
df.values	NumPy array from DataFrame
np.zeros(<size>)	NumPy array of all zeros
np.ones(<size>)	NumPy array of all ones
np.eye(<size>)	Identity NumPy array
np.arange(<start>,<stop>,<step>)	Equally spaced NumPy array with specific step
np.linspace(<start>,<stop>,<count>)	Equally spaced NumPy array with specific size
np.random.randint(<low>,<high>,<size>)	NumPy array of random ints
np.random.random(<size>)	NumPy array of random floats

NumPy Array Manipulation Methods

Method	Description
array.reshape(<new-shape>)	Reshape NumPy Array
array.transpose() OR array.T	Transpose NumPy Array
np.concatenate(<np-arrays>, <axis>)	Concatenate NumPy Arrays
np.flatten(<Nd-np-array>)	Flatten a NumPy Array
np.unique(<np-array>, <axis>)	Find unique elements
array.tolist()	NumPy Array to List

Search Methods

Method	Description
np.argmax(<np-array>, <axis>)	Max Element Index
np.argmin(<np-array>, <axis>)	Min Element Index
np.where(<condition>,<true-return-value>, <false-return-value>)	Conditional Search and Replacement
np.nonzero(<np-array>)	Index of non-zero elements

Mathematical Operations

Method	Description
np.sin(<np-array>)	Trigonometric Functions
np.cos(<np-array>)	
np.tan(<np-array>)	
np.floor(<np-array>)	Element-wise floor value
np.ceil(<np-array>)	Element-wise ceiling value
np.rint(<np-array>)	Round to nearest int
np.round_(<np-array>, <decimal-places>)	Round to decimal places
np.exp(<np-array>)	Element-wise exponent
np.log(<np-array>)	Element-wise logarithm
np.sqrt(<np-array>)	Element-wise square root
np.sum(<np-array>, <axis>)	Sum along an axis
np.mean(<np-array>, <axis>)	Mean along an axis
np.std(<np-array>, <axis>)	Std. dev along an axis

Matrix and Vector Operations

Method	Description
np.dot(<np-array1, np-array2>)	Dot Product
np.matmul(<np-array1, np-array2>)	Matrix Multiplication
np-array1 @ np-array2	
np.linalg.norm(<np-array>)	Vector Norm

Sorting Methods

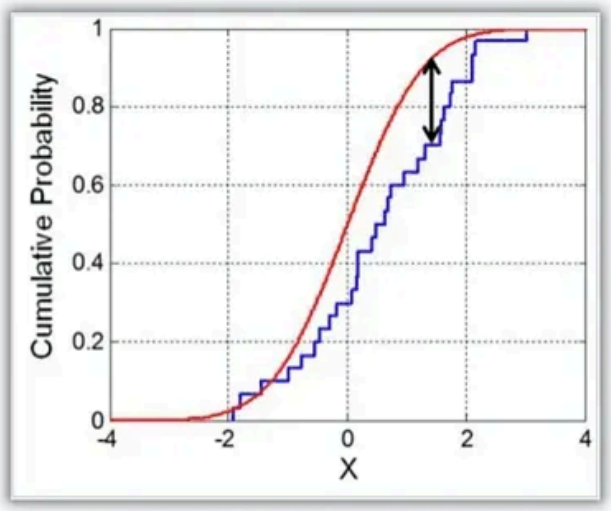
Method	Description
np.sort(<np-array>, <axis>)	Sort Array
np.argsort(<np-array>, <axis>)	Return the order of indices that sort the array

Next...

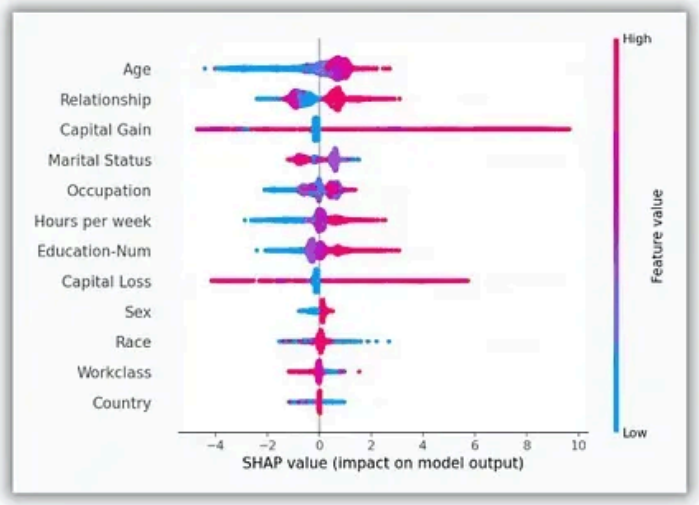
11 Most Important Plots in Data Science

blog.DailyDoseofDS.com

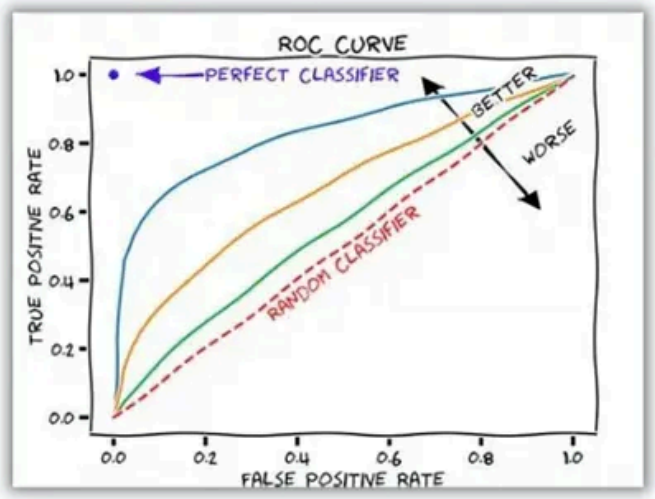
1 KS Plot



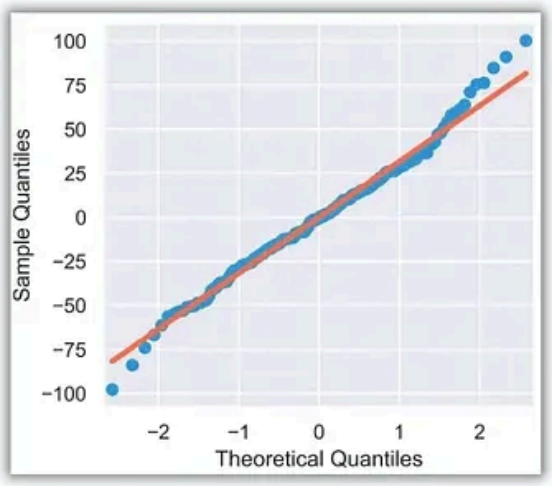
2 SHAP Plot



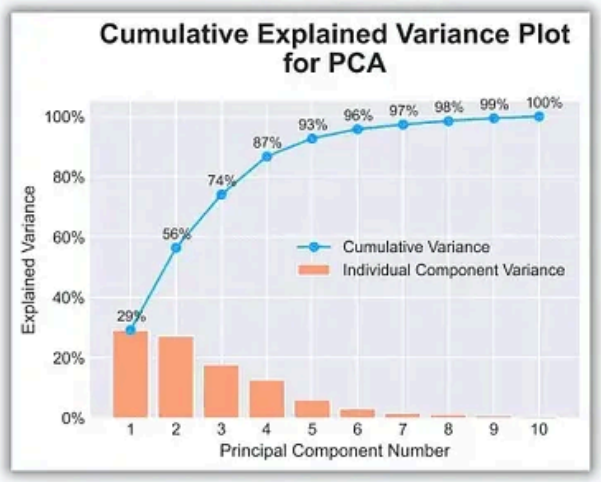
3 ROC Curve



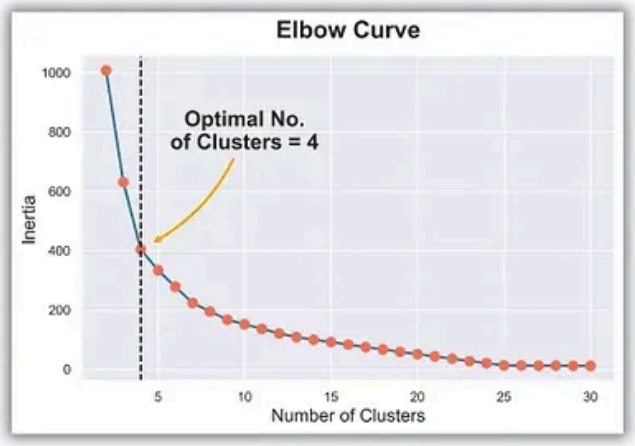
4 QQ Plot



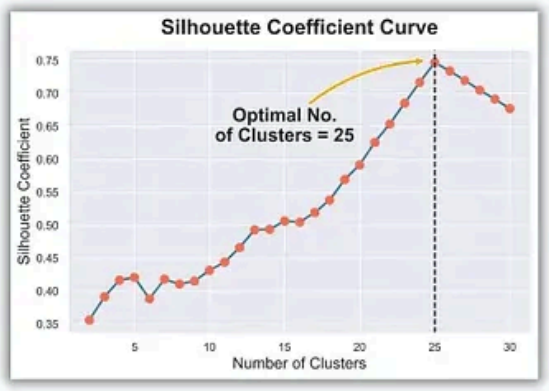
5 Cumulative Explained Variance



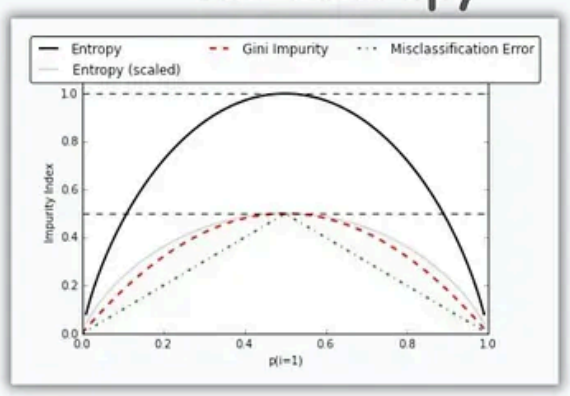
6 Elbow Curve



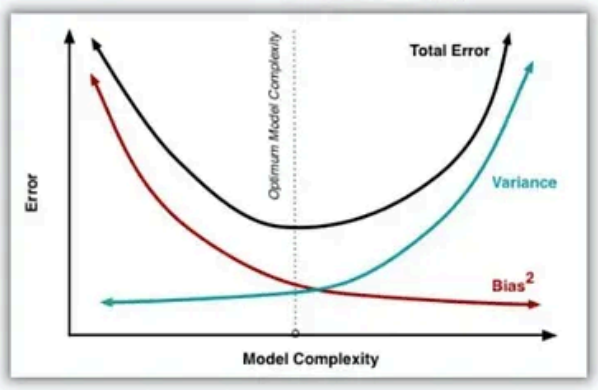
7 Silhouette Curve



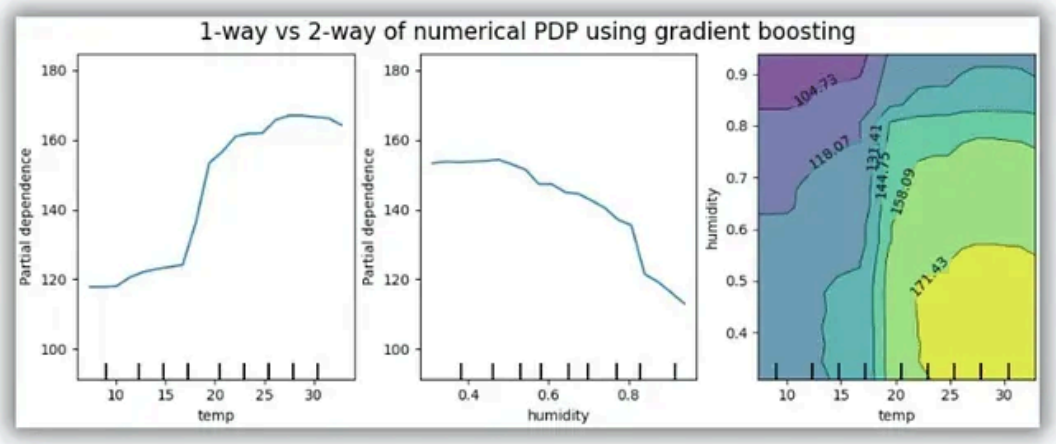
8 Gini impurity vs. Entropy



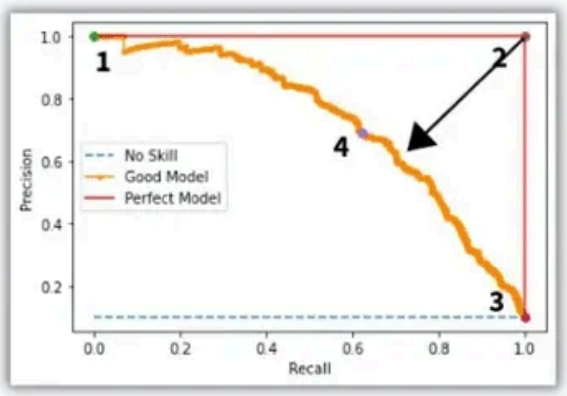
9 Bias-Variance Tradeoff



10 Partial Dependency Plot



11 Precision Recall Plot



Next...

10 Most Common Loss Functions in Machine Learning

blog.DailyDoseofDS.com

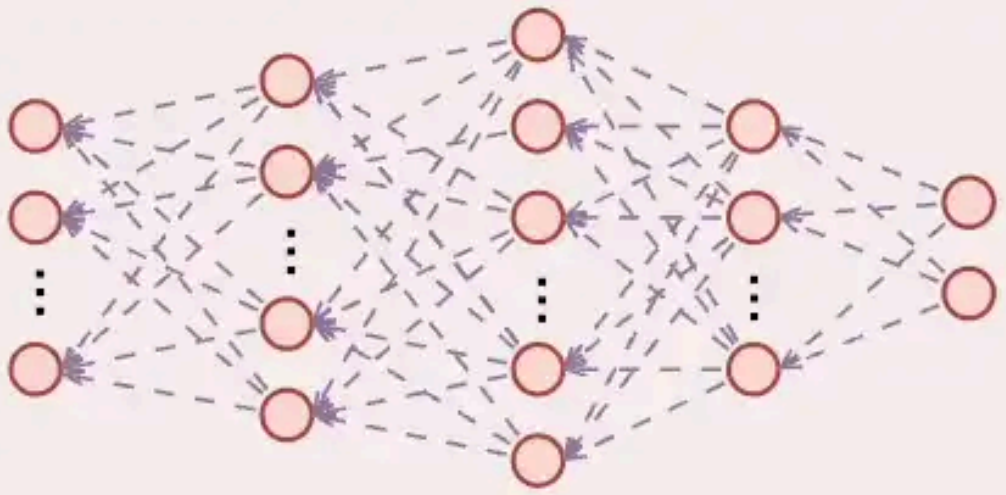
Loss Function Name	Description	Function
Regression Loss Functions		
Mean Bias Error	Captures average bias in prediction. But is rarely used for training.	$\mathcal{L}_{MBE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))$
Mean Absolute Error	Measures absolute average bias in prediction. Also called L1 Loss.	$\mathcal{L}_{MAE} = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - f(x_i)\rvert$
Mean Squared Error	Average squared distance between actual and predicted. Also called L2 Loss.	$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$
Root Mean Squared Error	Square root of MSE. Loss and dependent variable have same units.	$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$
Huber Loss	A combination of MSE and MAE. It is parametric loss function.	$\mathcal{L}_{\text{Huberloss}} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & : \lvert y_i - f(x_i)\rvert \le \delta \\ \delta(\lvert y_i - f(x_i)\rvert - \frac{1}{2}\delta) & : otherwise \end{cases}$
Log Cosh Loss	Similar to Huber Loss + non-parametric. But computationally expensive.	$\mathcal{L}_{\text{LogCosh}} = \frac{1}{N}\sum_{i=1}^{N} log(cosh(f(x_i) - y_i))$
Classification Loss Functions (Binary + Multi-class)		
Binary Cross Entropy (BCE)	Loss function for binary classification tasks.	$\mathcal{L}_{BCE} = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(x_i)) + (1 - y_i) \cdot log(1 - p(x_i))$
Hinge Loss	Penalizes wrong and right (but less confident) predictions. Commonly used in SVMs.	$\mathcal{L}_{\text{Hinge}} = max(0, 1 - (f(x) \cdot y))$
Cross Entropy Loss	Extension of BCE loss to multi-class classification.	$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}.log(f(x_{ij}))$ $N : samples; M : classes$
KL Divergence	Minimizes the divergence between predicted and true probability distribution	$\mathcal{L}_{KL} = \sum_{i=1}^{N} y_i \cdot log(\frac{y_i}{f(x_i)})$

Next...

Full Fine Tuning, LoRA and RAG

blog.DailyDoseofDS.com

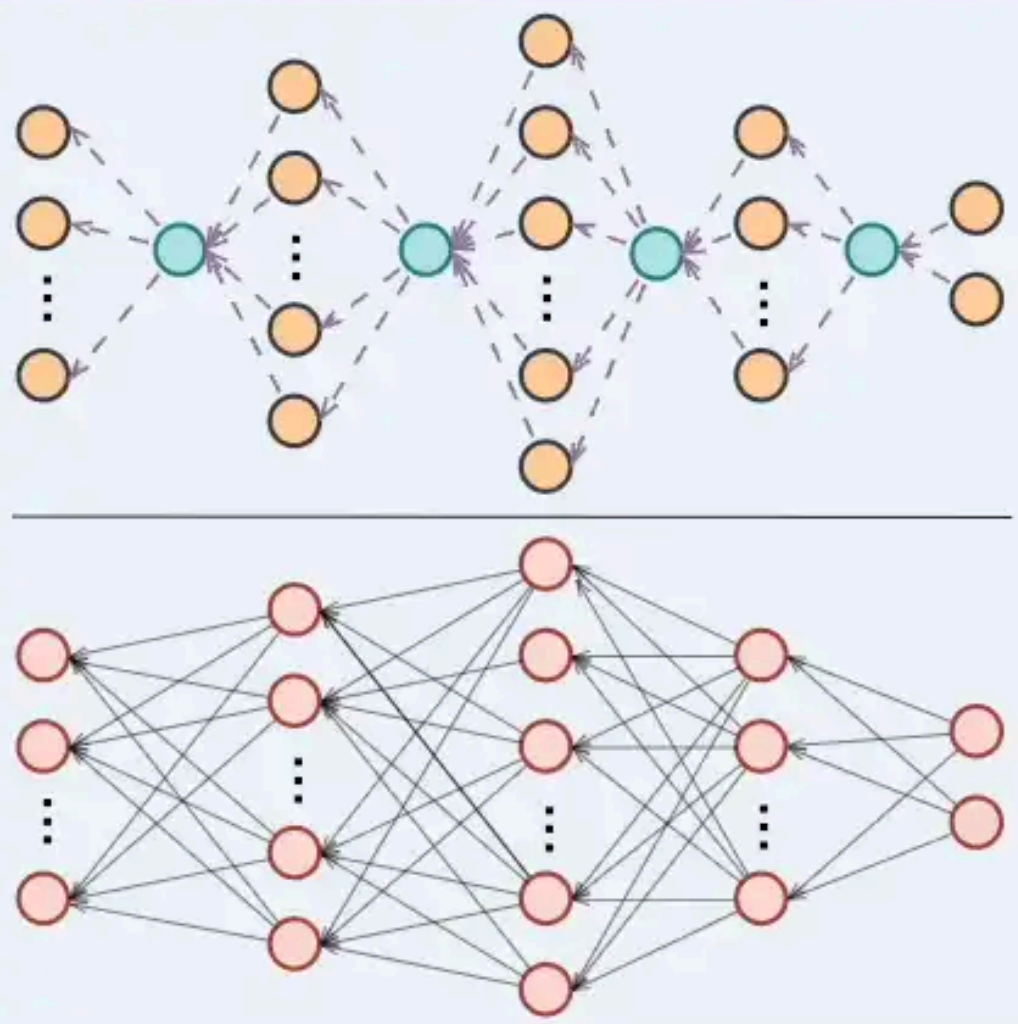
Full Fine Tuning



Gradient flow

Full pre-trained network

LoRA Fine Tuning



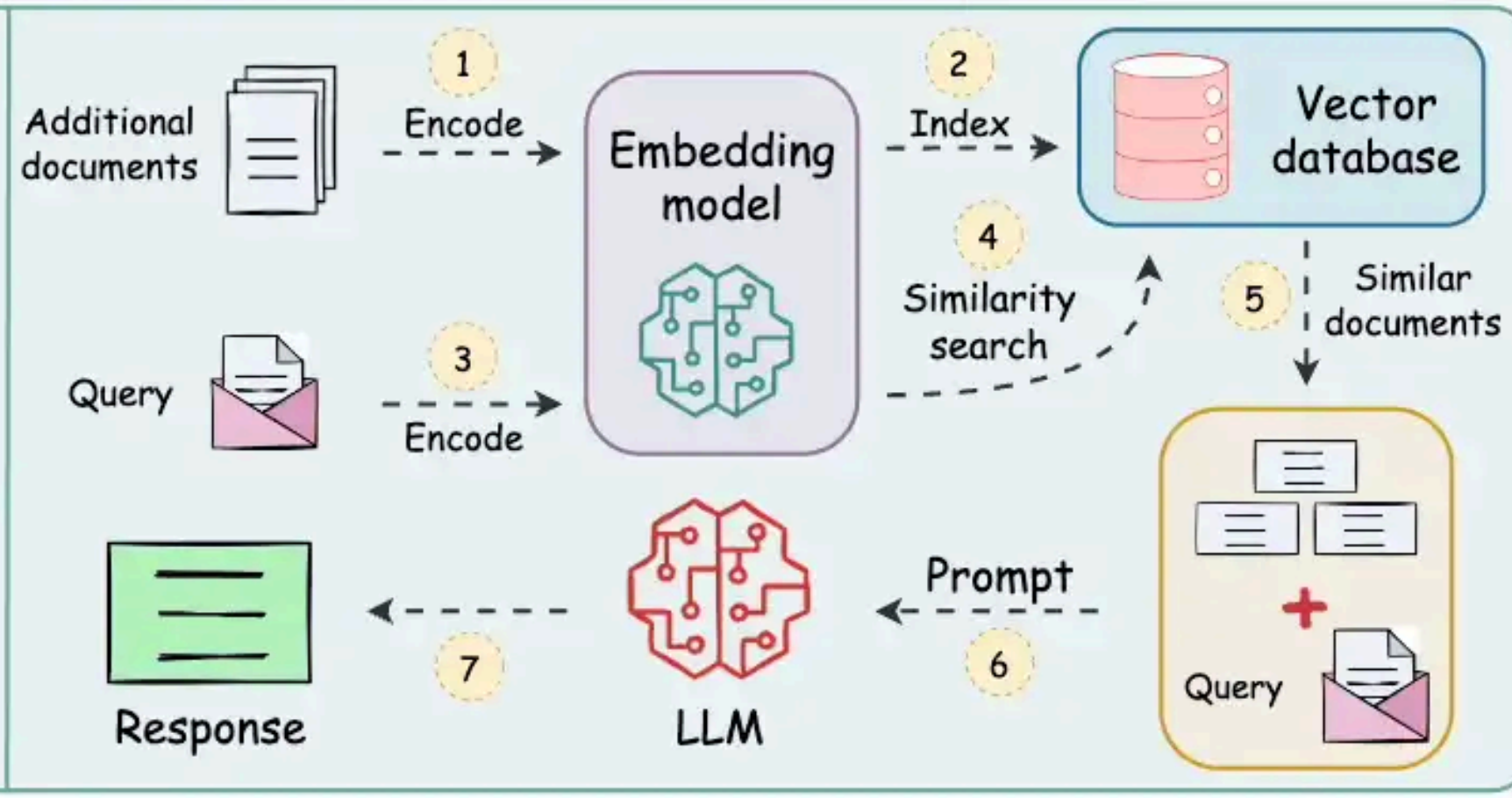
Gradient flow

Additional LoRA layers

No gradient flow

Full pre-trained network

RAG

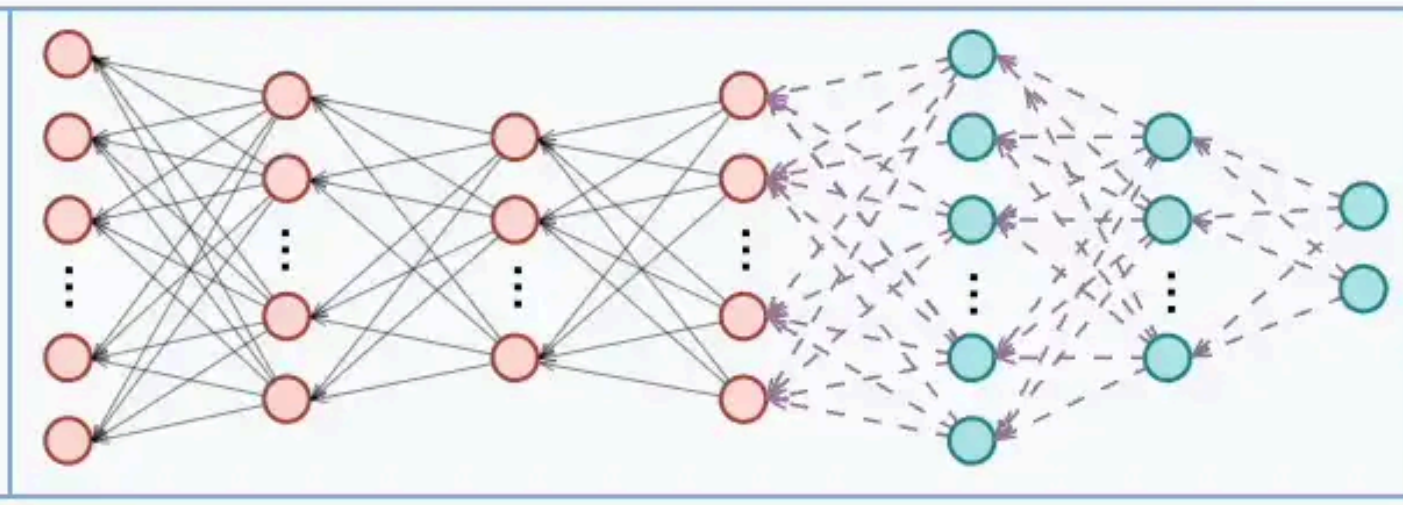


Next...

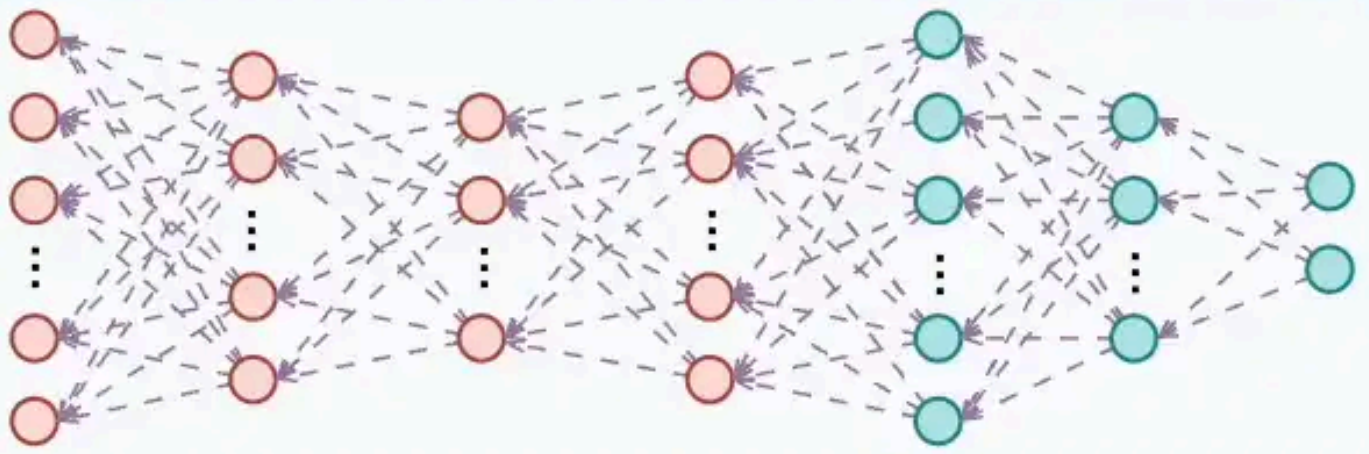
Transfer Learning vs. Fine Tuning vs. Multitask Learning vs. Federated Learning

blog.DailyDoseofDS.com

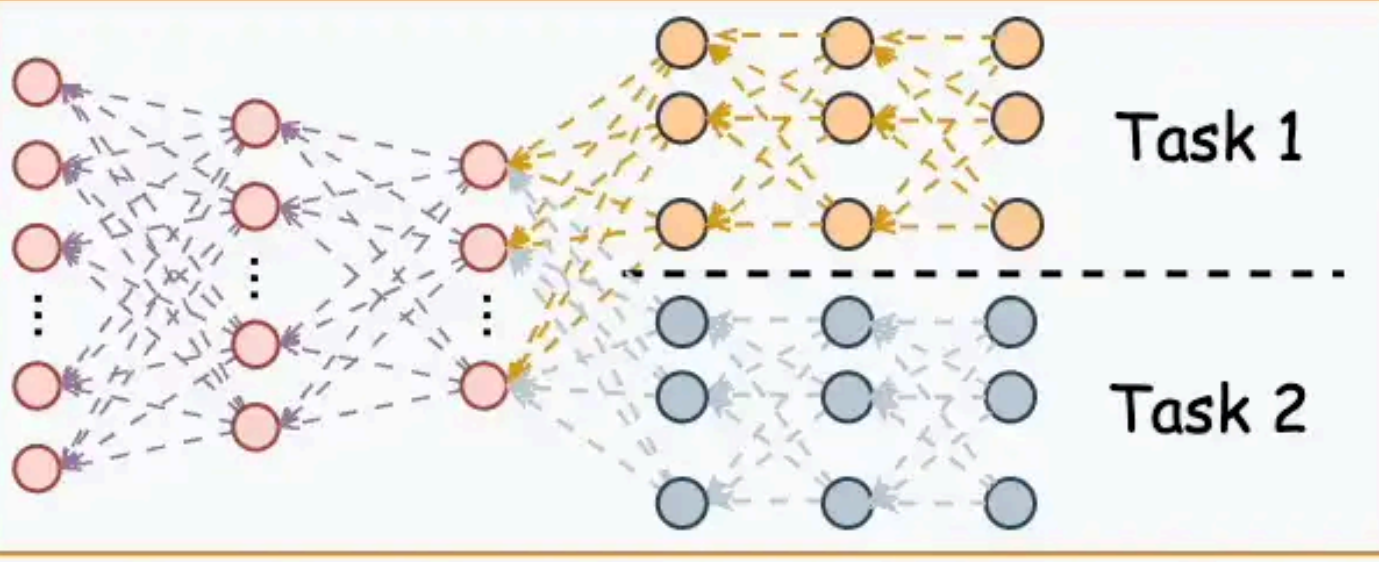
Transfer Learning



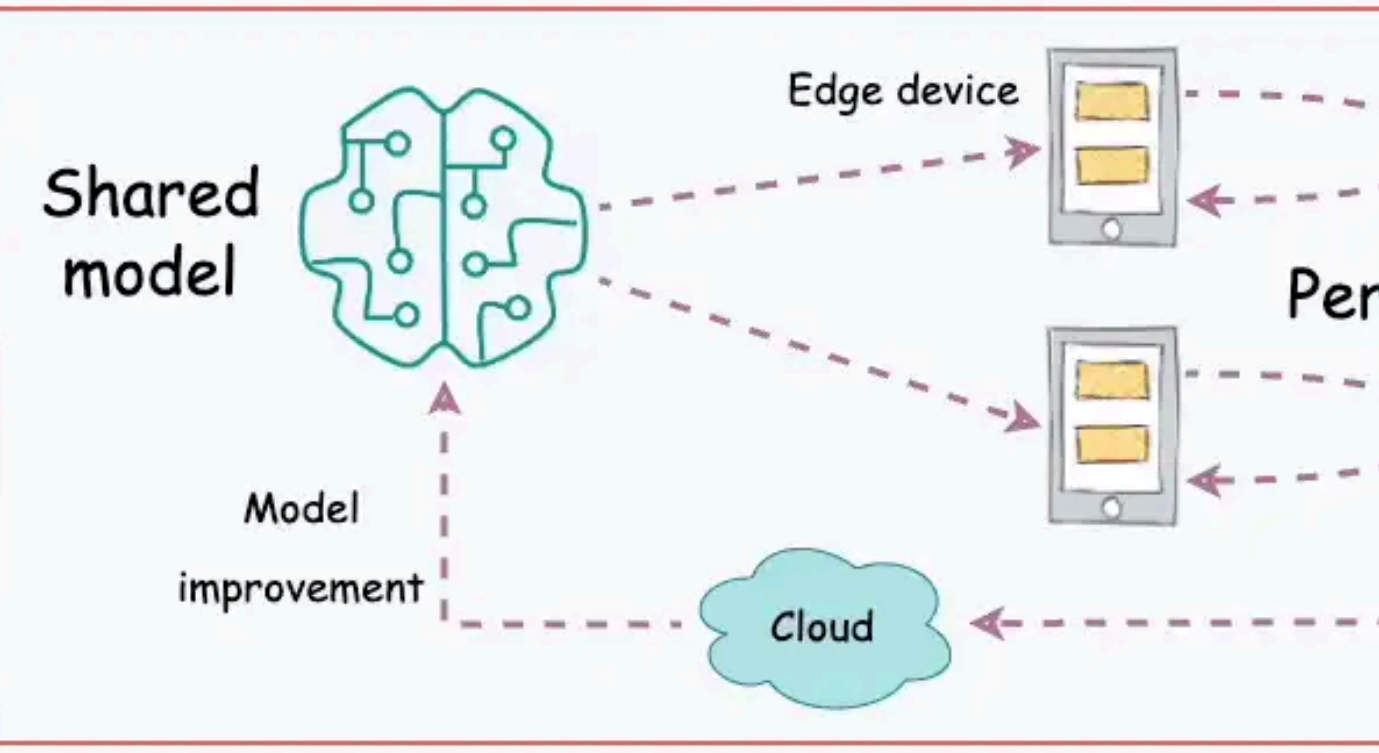
Fine Tuning



Multitask Learning



Federated Learning



Next...