Full Stack AI Projects

AI Specialization

Henry Ruiz, PhD.
GDE ML
@devharuiz
https://haruiz.github.io/

What is this course about?

How to :

Build and design applications and products with AI.

- A **machine learning model** can only begin to **add value to an organization** when that model's insights routinely become available to the users for which it was built

Create and Lead ML groups

Establish a working relationship between software engineers, infrastructure managers and data scientists.

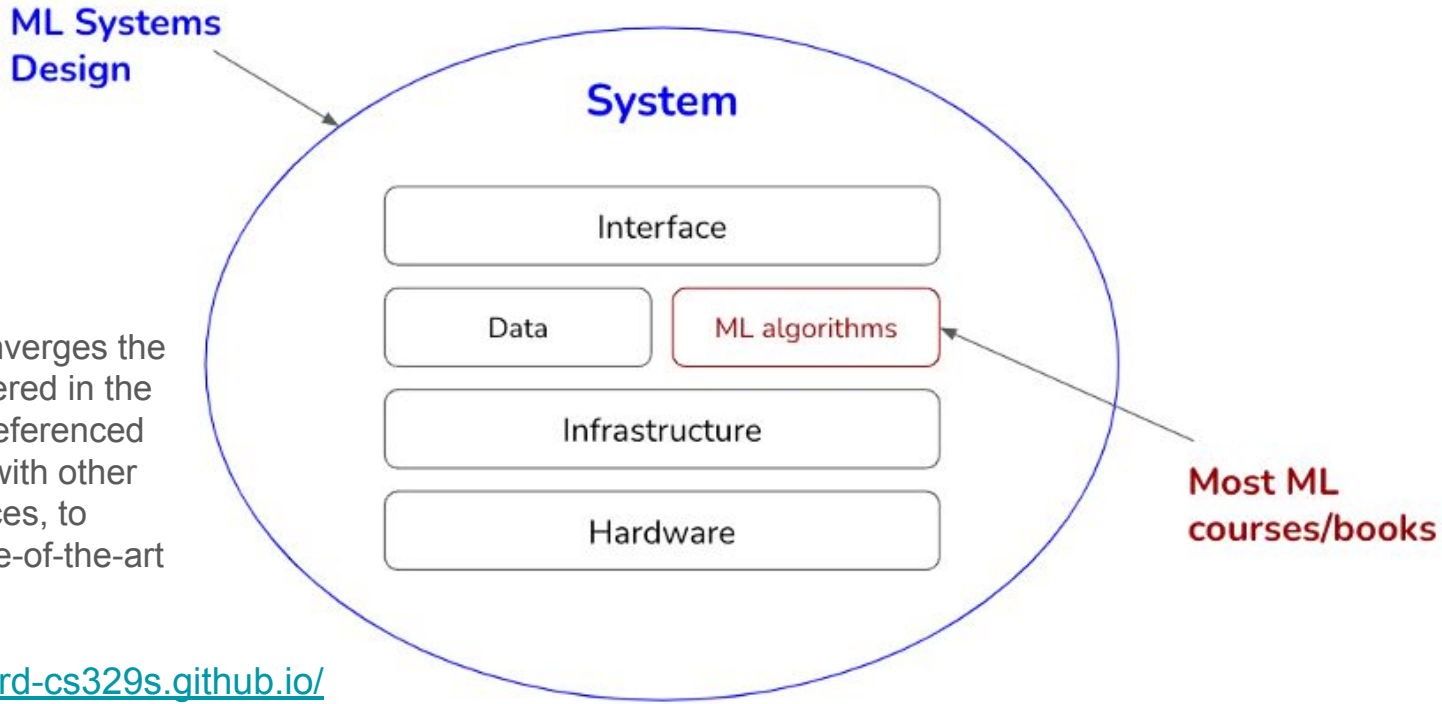
- **Software engineers** who want to build robust and responsible systems meeting the specific challenges of working with ML components
- **Data scientists** who want to facilitate getting a prototype model into production

Describe and Understand some of the most common ML deployment scenarios

Evaluate, testing and Monitoring ML systems

ML in production

ML Systems design



This class converges the materials covered in the two courses referenced below, along with other online resources, to provide a state-of-the-art course:

https://stanford-cs329s.github.io/

https://fullstackdeeplearning.com/

What is this course NOT about

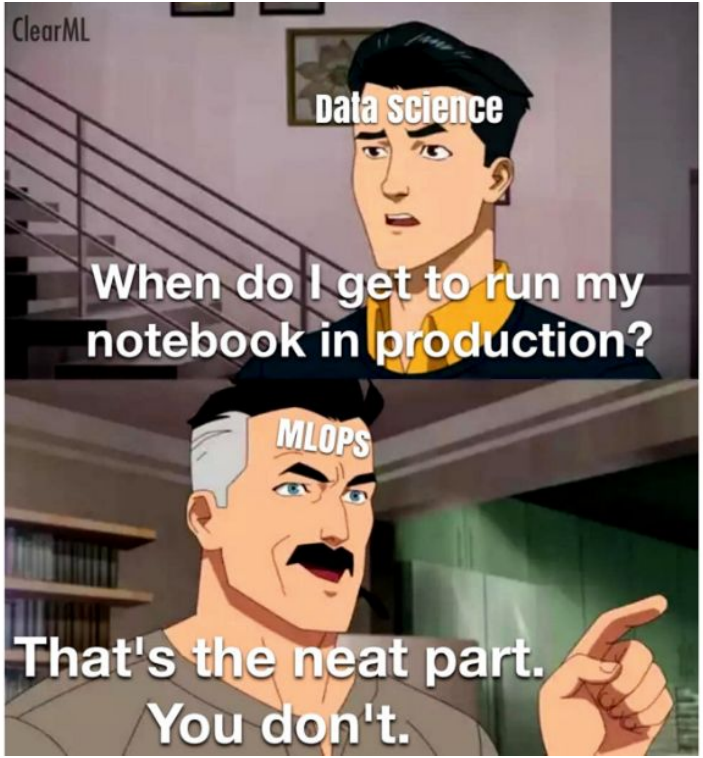
- Machine learning/deep learning algorithms
  - CS 229: Machine Learning
  - CS 230: Deep Learning
  - CS 231N: Convolutional Neural Networks for Visual Recognition
  - CS 224N: Natural Language Processing with Deep Learning
- Computer systems
  - CS 110: Principles of Computer Systems
  - CS 140E: Operating systems design and implementation
- UX design
  - CS 147: Introduction to Human-Computer Interaction Design
  - DESINST 240: Designing Machine Learning: A Multidisciplinary Approach

What is machine learning system design?

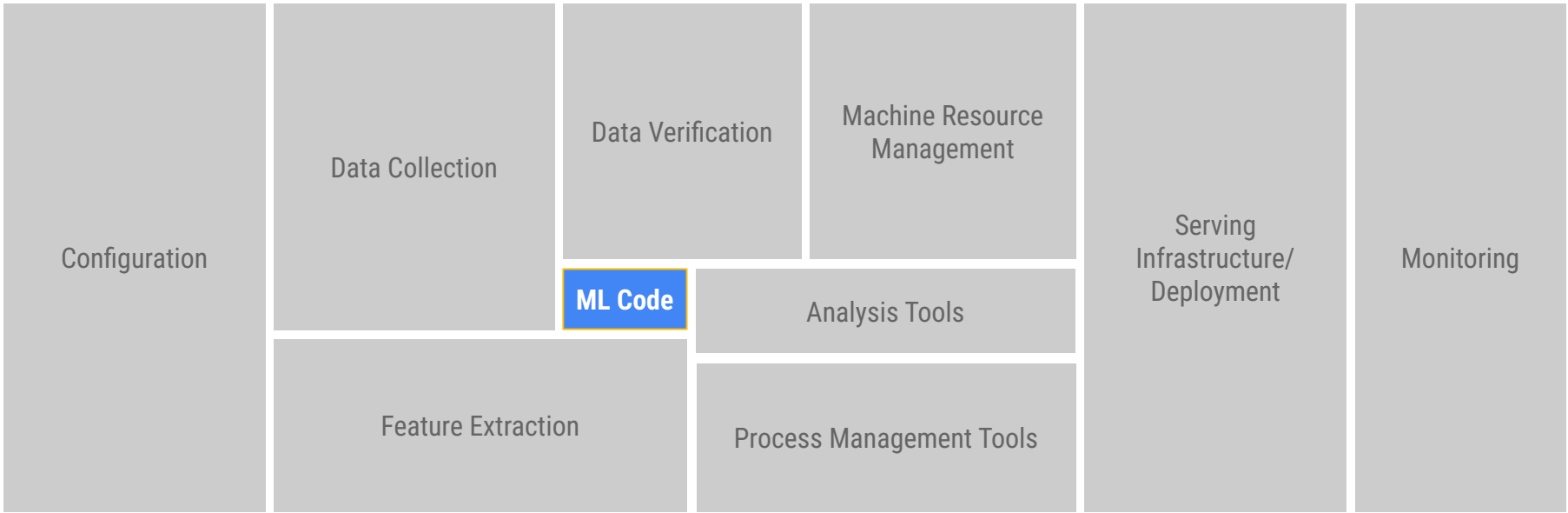
The process of **defining the <u>interface, algorithms, data, infrastructure, and hardware</u>** for a machine learning system to satisfy specified requirements.

The course should help you to address some of the below questions ...

You've trained a model, now what?
What are different components of an ML system?
How to engineer features?
How to evaluate your models, both offline and online?
What's the difference between online prediction and batch prediction?
How to serve a model on production?
How to continually monitor and deploy changes to ML systems?



… a production solution requires so much more

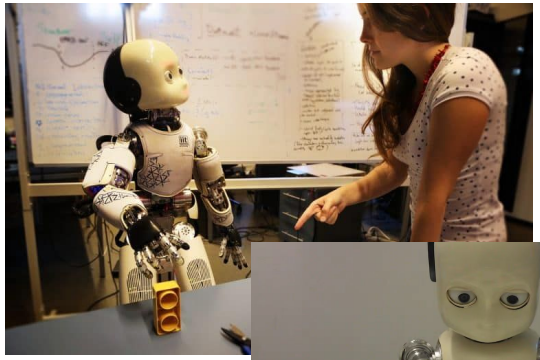


Model's deployment course could potentially be the continuation

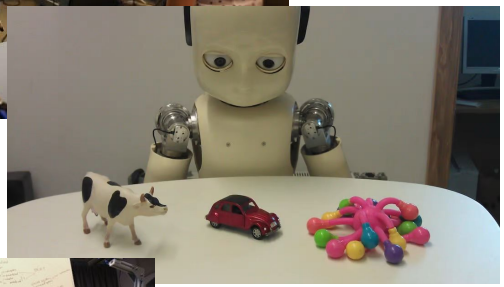
Model deployment is one of the most important steps in the ML pipeline. We have spent a lot of time and effort playing around with different models, training and tuning our model hyperparameters, so after evaluating its performance and obtaining that long-awaited score, now is the time to release it to the world, exposing this to real use.

It sounds like graduation time!!.

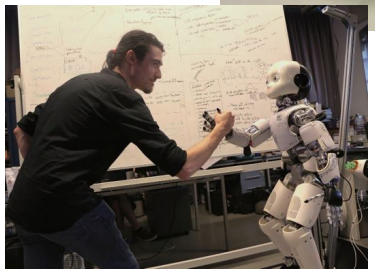
Training



Testing



Evaluation

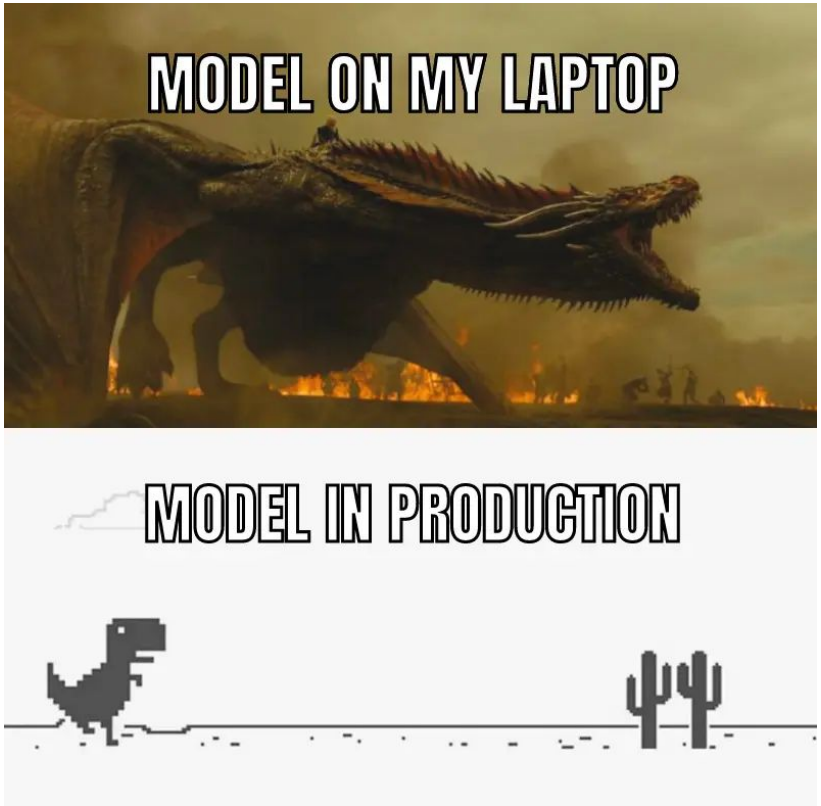


Deployment

Model Deployment

A machine learning model can only begin to add value to an organization when that model's insights routinely become available to the users for which it was built.

The process of taking a trained ML model and making its predictions available to users or other systems is known as deployment.



Course Methodology

Three hours of classes in two sections every week

Thursdays 6:30 pm – 9:30 pm

Saturday 11:00 am – 2:00 pm

1 Final project and a paper discussion

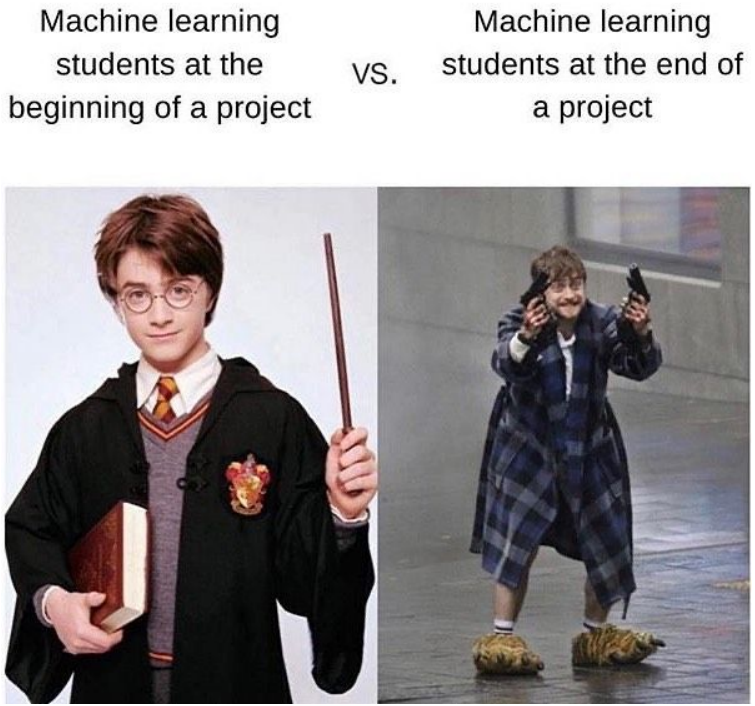
5 hours of lecture, 1 working on the project (optional)

Setting up Machine Learning Projects

Like any other software solution, ML systems require a **well-structured methodology** to maximize the success rate of the implementation.

ML algorithms are the less challenging part. The hard part is **making algorithms work with other software infrastructure components to solve real-world problems.**

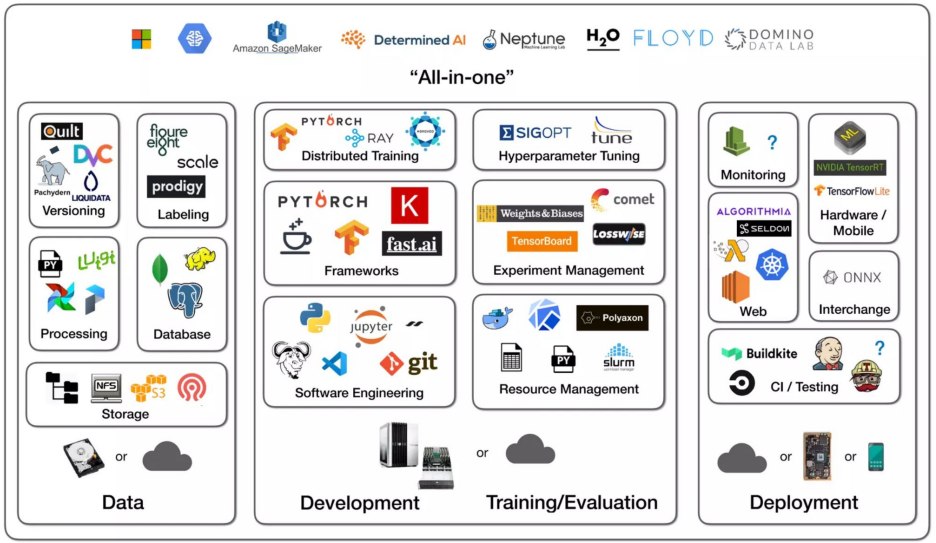
Non-ML components cause 60/96 failures, and 60% of the models never make it into production.



Infrastructure & Tooling

The number of **available tools to work with ML seems endless.**

Selecting the appropriate tools depends on:
the kind of problem, **type of solution, deployment scenario, capacity building**, team experience, cost, hardware and software infrastructure, etc.




Google
Cloud Platform
Azure
amazon
web services™
IBM Watson™
FLOYD
scikit
learn
TensorFlow
PyTorch
NumPy
Pandas
DASK
Numba
astropy
gprlib
OPTUNA
SciPy
OpenCV

Data Management

ML is all about data

Data can be represented and stored differently.

We will explore different storage and data management solutions in this part of the course.

Machine Learning Teams

Machine Learning talents are expensive and scarce.

ML teams have diverse roles.

Managing and leading ML and Data Science teams require unique skills.

Here we will learn more about these roles' importance and impact within the organization.

Training, Debugging and Design patterns

In engineering disciplines, **design patterns capture best practices and solutions to commonly occurring problems.**

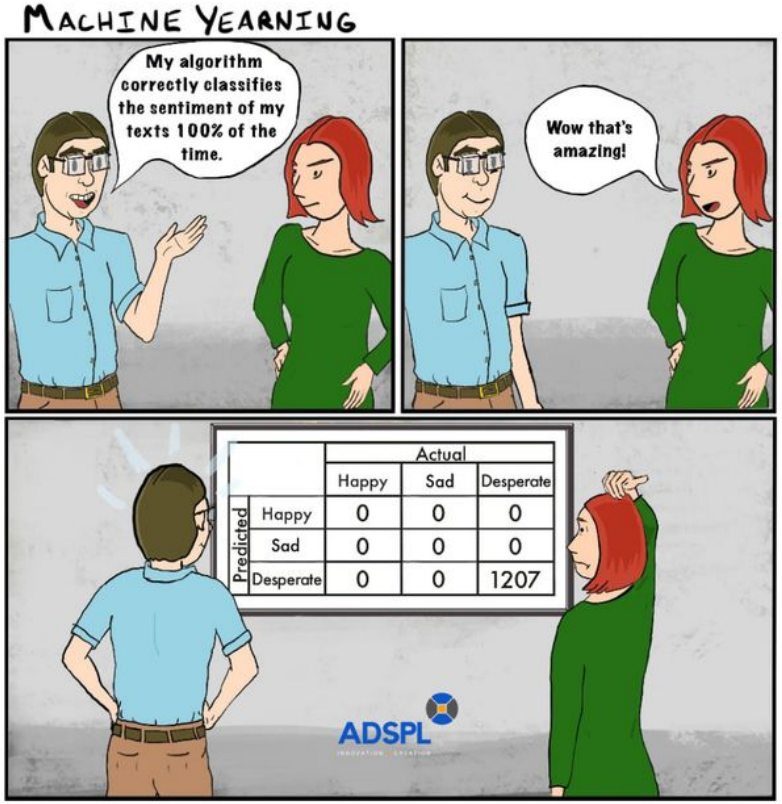
They codify the knowledge and experience of experts into advice that all practitioners can follow.

This course will introduce some design patterns or repeatable solutions to issues in ML engineering for training and debugging ML systems.

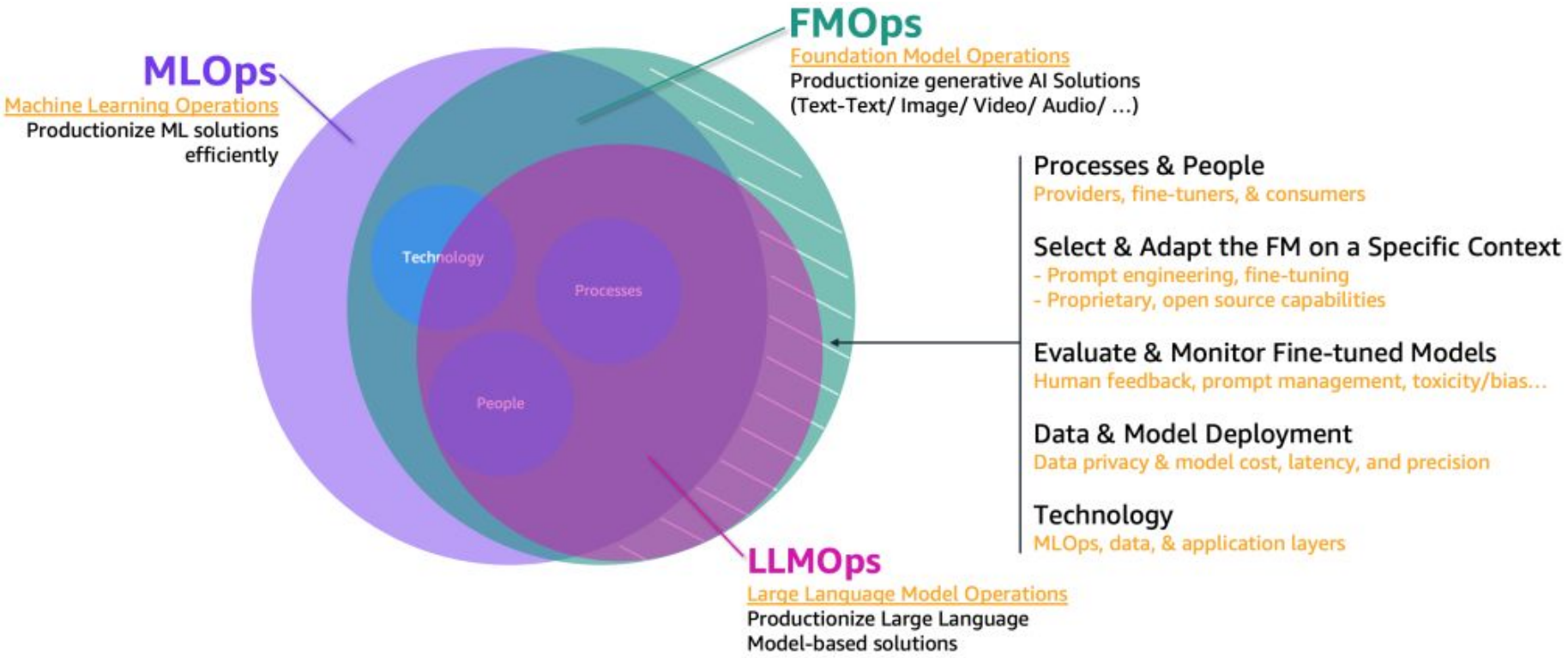
Testing and Deployment

A machine learning model can only begin to add value to an organization when that **model's insights routinely become available to the users for which it was built.**

Let's learn together about **troubleshooting and deploying ML models in production.**

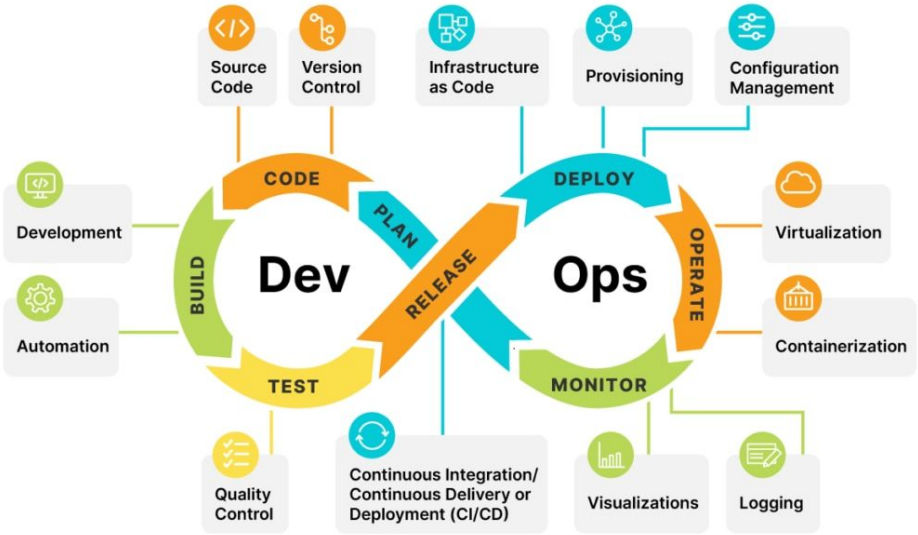


DevOps, MLOps, LLMOps, FMOps



DevOps

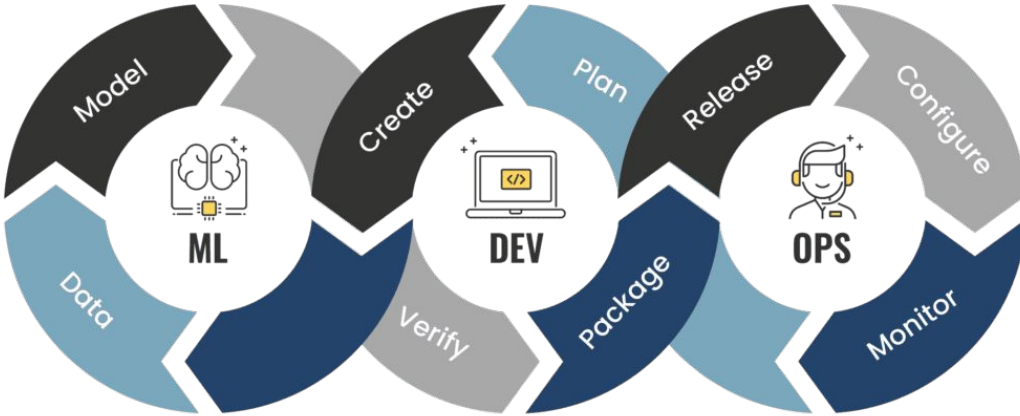
combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity and quality



MLOps

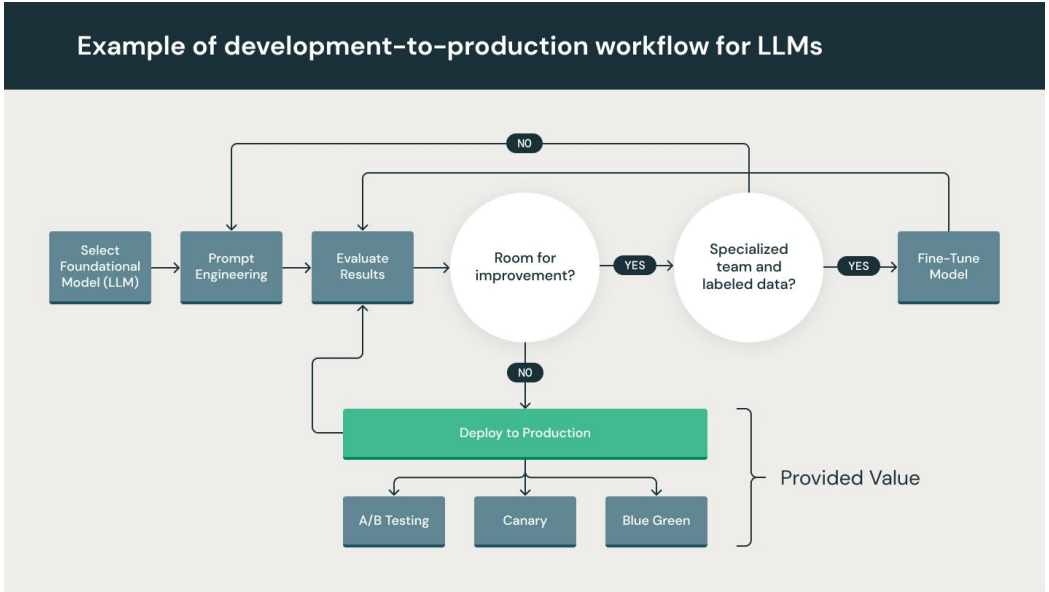
MLOps is **a methodology for ML engineering that unifies ML system development** (the ML element) with **ML system operations** (the Ops element).

This course will discuss how MLOps maximize the capacities and resources of ML teams by providing a set of standardized processes and technology capabilities for building, deploying, and operationalizing ML systems rapidly and reliably.



LLMOps

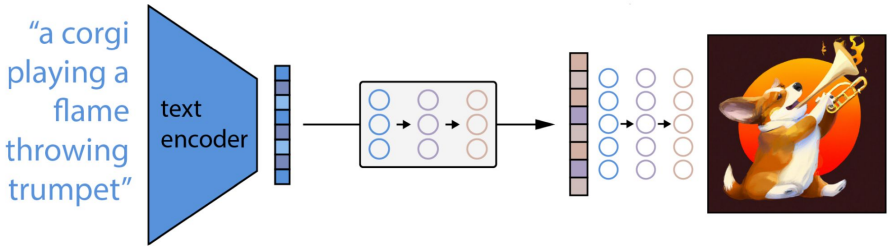
Large Language Model Ops (LLMOps) encompasses the practices, techniques and tools used for the operational management of large language models in production environments.



https://www.databricks.com/glossary/llmops

Research Areas

In this section, the current advances in AI will be discussed, such as **Attention-based models, Transformers, Diffusion models, and Multimodal models**



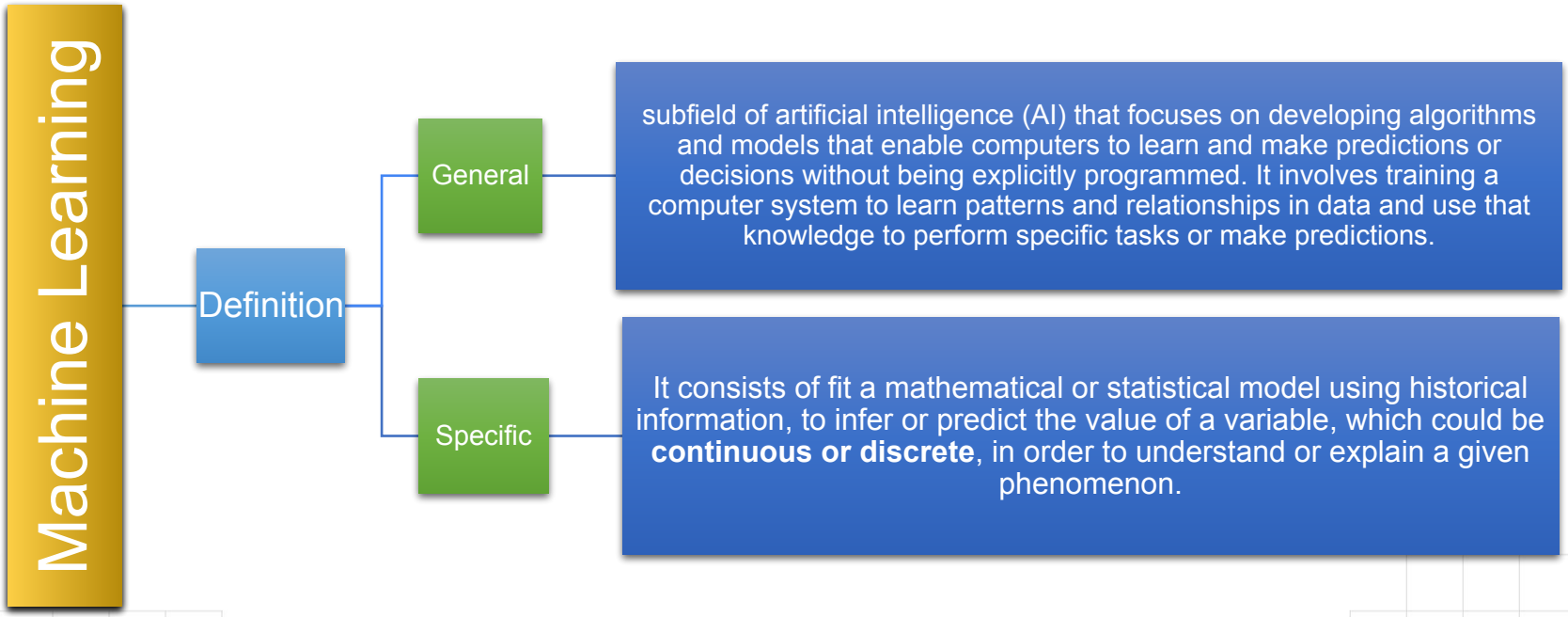
ML basic concepts review

What is machine Learning?

AI vs Machine Learning vs Deep Learning

AI	• is the broader **umbrella under all** • technique which allows the machines to act like humans by replicating their behavior and nature. It includes genetic algorithms, search, optimization, and planning
Machine Learning	• Is a subset of artificial intelligence • It allows the machines to learn and make predictions based on historical information
Deep Learning / Neural networks (Brain Inspired)	• Deep learning is a particular kind of machine learning • Uses neural networks to simulate human like decision making

What is Machine Learning?



Machine Learning

Levels of knowledge

Machine Learning practitioner	Machine Learning Research
• Tasks Oriented: • Queries databases. • Cleaning data. • Writing scripts to transform data and probe algorithms. • Play around with libraries. • Make all easy. • Find the best model writing custom code.	• Research Oriented: • Read papers. • Implement the algorithm from scratch. • Translate Math into code. • Reducing algorithms. • Use beauty math to develop their own models.

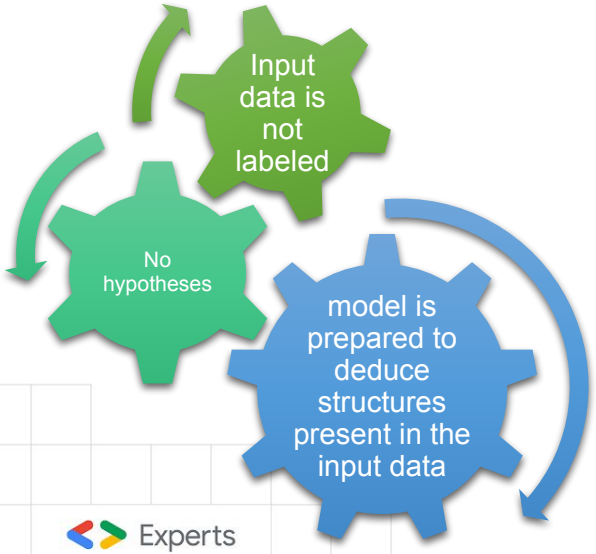
Experts

Types of Machine Learning

Unsupervised Learning

Data driven

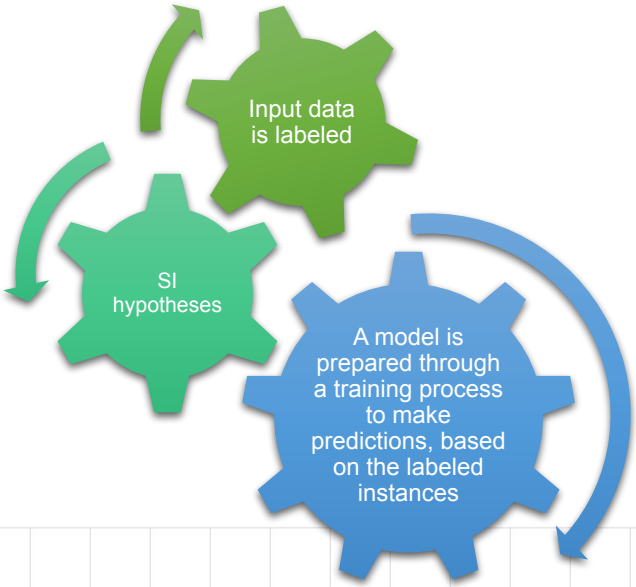
Learn structure and find pattern in the data to generate information



Supervised Learning

Task driven

Learn how training examples maps to labels to make predictions or classify new, unseen data



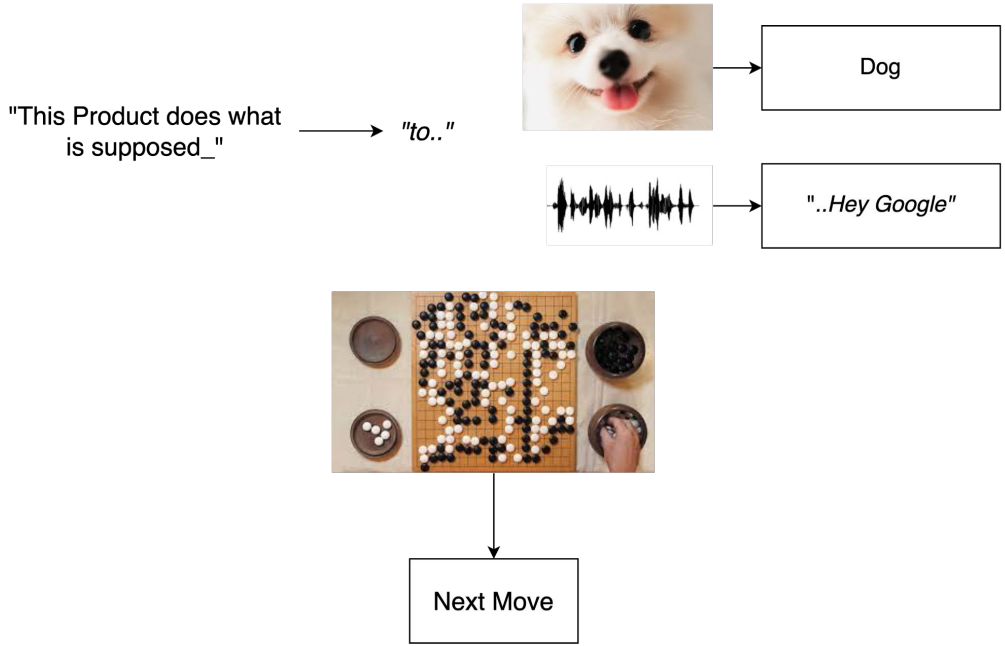
Reinforcement learning

Learn from mistakes

Learn how to act in an environment to obtain rewards.

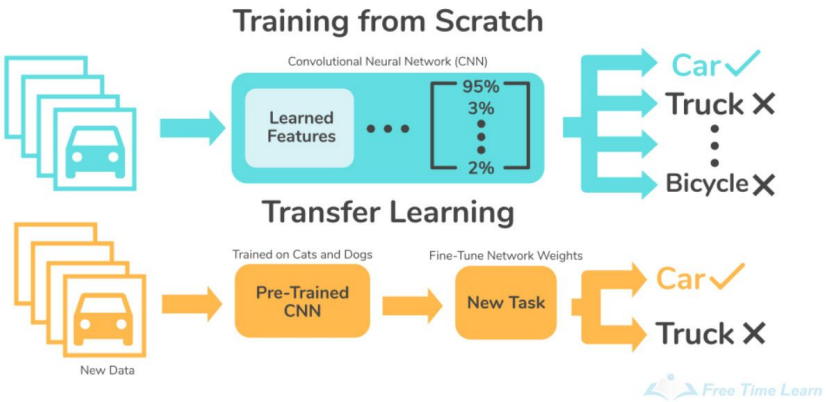
Converging on just..

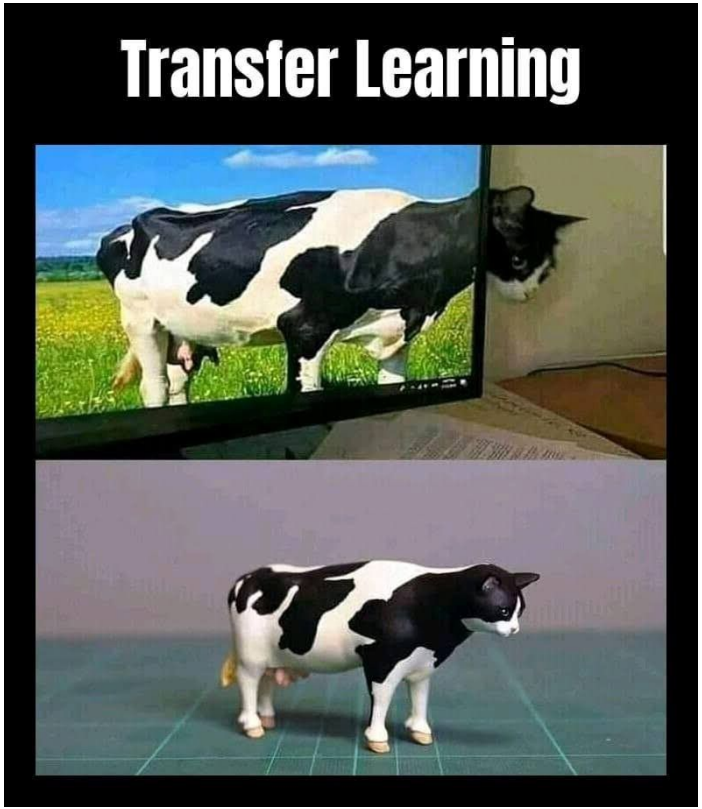
Supervised or self-supervised learning



Transfer Learning vs Fine Tuning

Transfer learning and fine-tuning are two closely related techniques used in machine learning and deep learning to leverage pre-trained models for a task that is similar, but not identical, to the one the model was originally trained for. Despite their similarities, there are distinct differences between the two:





Transfer Learning

Definition: Taking a pre-trained model (trained on a large dataset for a task) and applying it to a different but related task. This can mean using the model as is for feature extraction or as a starting point for further training.

Flexibility: Transfer learning can be very flexible in terms of how much of the pre-trained model you use. You might use the entire model, just the convolutional base, or even a part of the architecture as a feature extractor.

Application: It is particularly useful when you have a limited amount of data for the new task. You can use the layers of the pre-trained model to extract features and train a new, smaller model on top of these features for your specific task.

Training Speed and Data Requirements: It reduces the need for a large dataset for the new task and can significantly decrease the training time, since the model has already learned useful features from the original dataset it was trained on.

Fine Tuning

Definition: Fine-tuning takes transfer learning a step further. After initializing your model with the pre-trained weights, you continue training it on your new, specific task, allowing the weights to be adjusted from their pre-trained values. This can lead to better performance than transfer learning alone, as the model adjusts more closely to the specifics of the new task.

Flexibility: Fine-tuning requires a more careful approach than transfer learning, as adjusting too many parameters too quickly can lead to overfitting, especially if the new dataset is small.

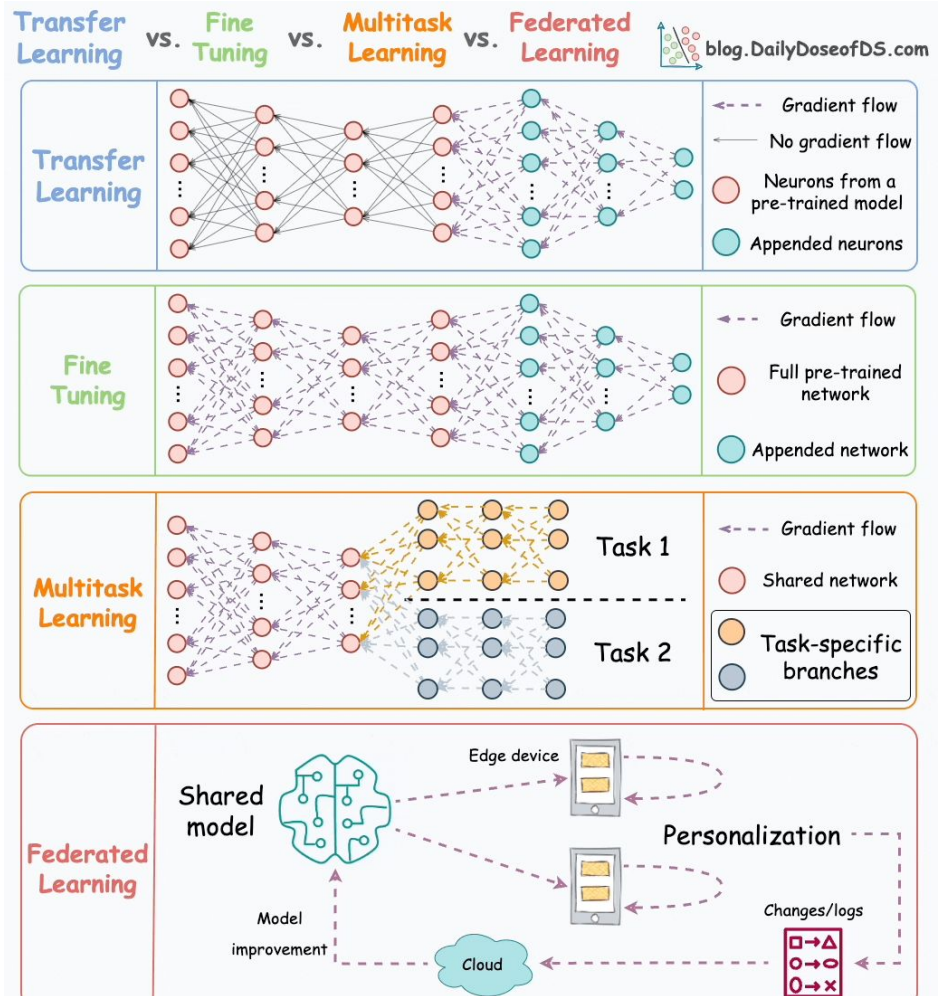
Application: Fine-tuning is often used after transfer learning. Initially, you might freeze the weights of the pre-trained layers and only train the top-level layers specific to your task. Then, you might unfreeze all or a portion of the pre-trained layers and continue training, allowing the pre-trained weights to adjust to your specific task.

Training Speed and Data Requirements: Fine-tuning typically requires more data than transfer learning without fine-tuning, because you are adjusting more weights within the model. It also usually takes longer, since you are effectively continuing the training process, although this can lead to improved performance on the task.

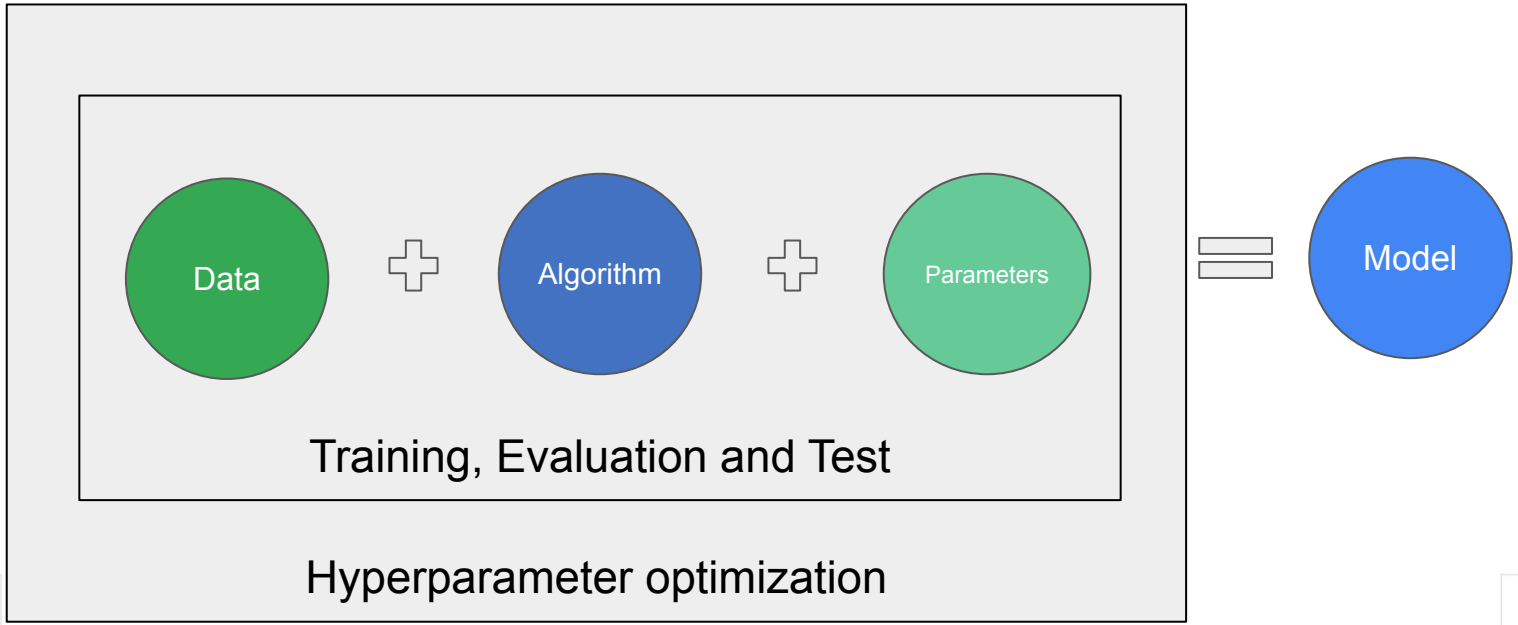
In summary

In summary, transfer learning is about leveraging pre-trained models for a related task, often without altering the original model's weights significantly. Fine-tuning, on the other hand, involves making more extensive adjustments to the model's weights to tailor it more closely to the new task, which can provide improved performance at the cost of requiring more data and potentially longer training times.

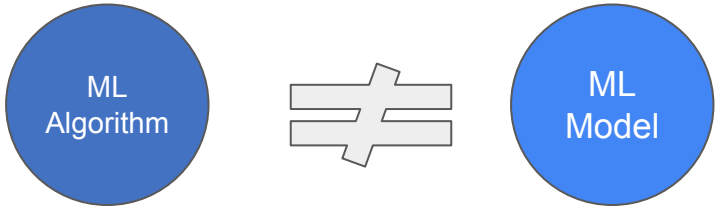
https://www.blog.dailydoseofds.com/p/transfer-learning-vs-fine-tuning


Transfer Learning vs. Fine Tuning vs. Multitask Learning vs. Federated Learning
blog.DailyDoseofDS.com
Transfer Learning
Gradient flow
No gradient flow
Neurons from a pre-trained model
Appended neurons
Fine Tuning
Gradient flow
Full pre-trained network
Appended network
Multitask Learning
Task 1
Task 2
Gradient flow
Shared network
Task-specific branches
Federated Learning
Shared model
Edge device
Personalization
Changes/logs
Model improvement
Cloud

The ML equation



The ML equation



Procedure to find the best parameters

Algorithm + best parameters

ML Pipeline

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

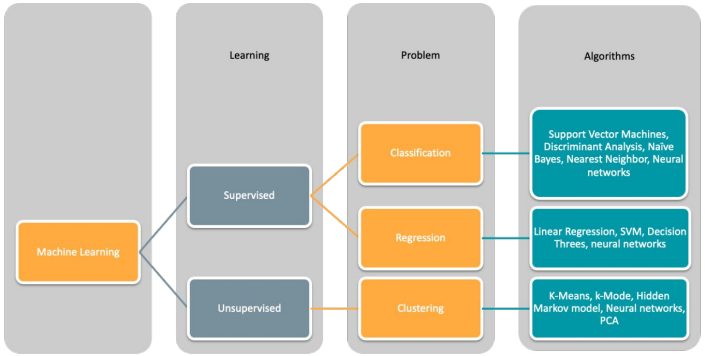
Structured data

Non-Structured data

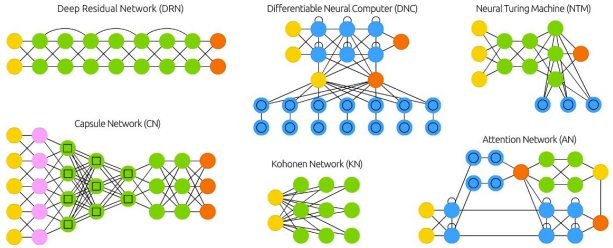
Fit data

Training

Algorithm/Architecture Selection



Traditional ML algorithms



https://www.asimovinstitute.org/neural-network-zoo/

Deep Learning

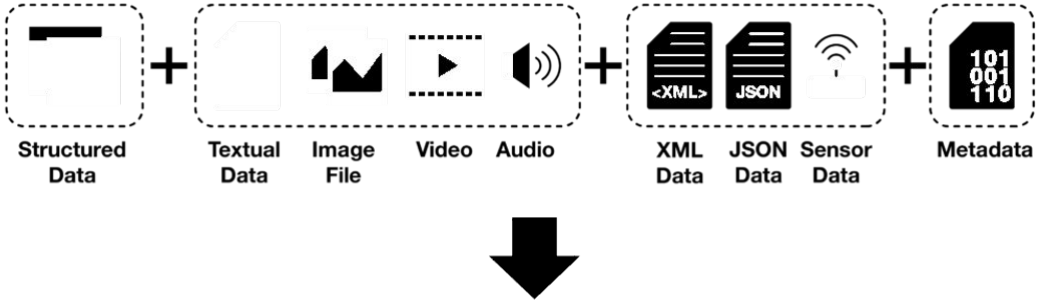
Evaluation and test

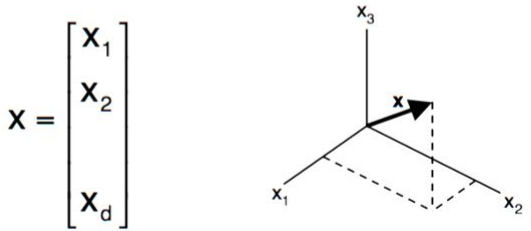
Model

Experts

Google Developers

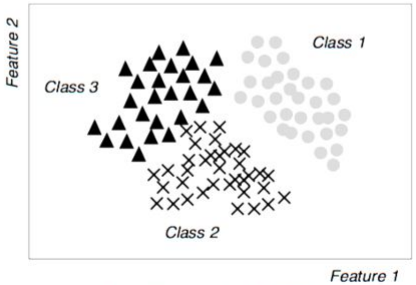
How our ML model interpret our data?





Feature vector

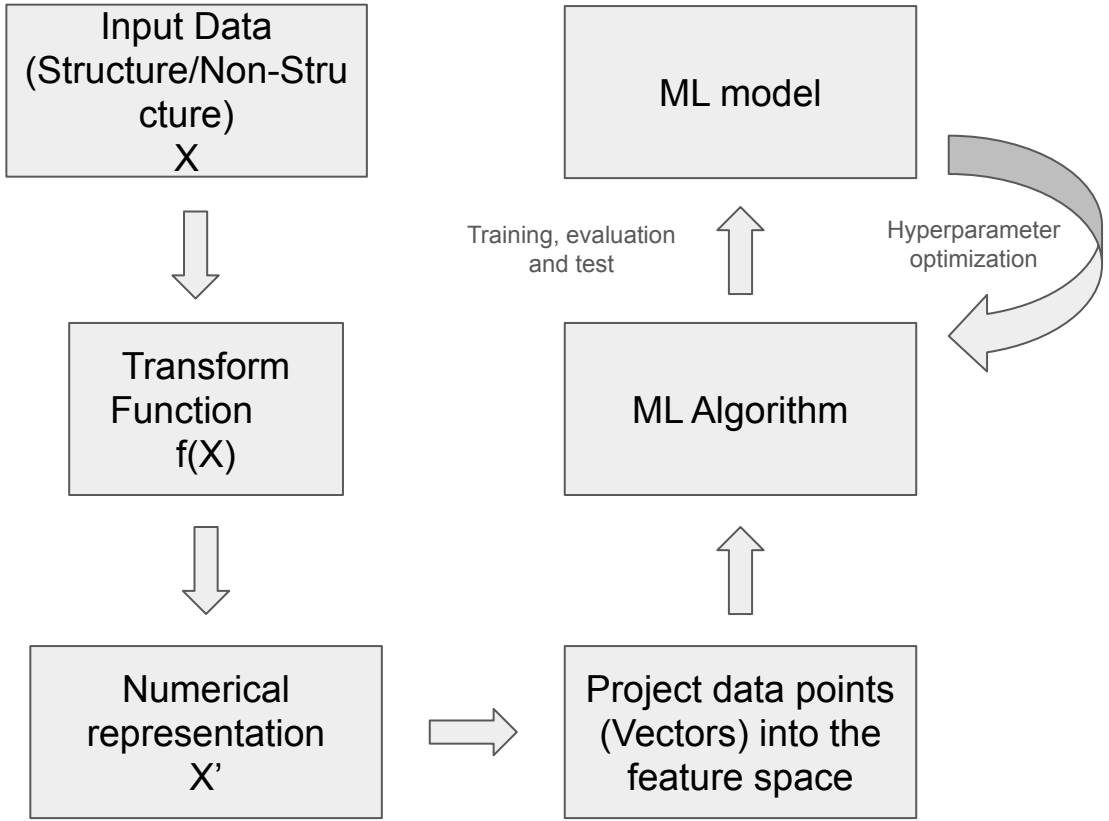
Feature space (3D)



Scatter plot (2D)

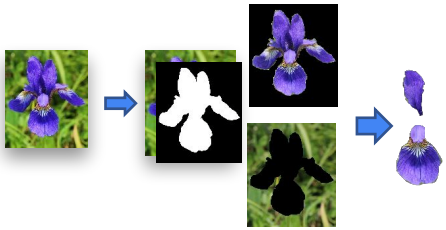
Embeddings

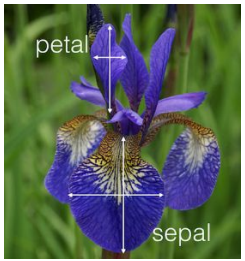
Model's performance highly depend of how much information is **encoded** in the **numerical representation/embedding representation** of your data.



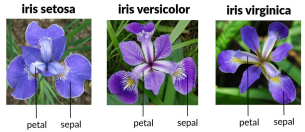
Features Engineering Approach (Features are known)

Traditional ML relies on your Feature engineering skills to transform your data into numbers





	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

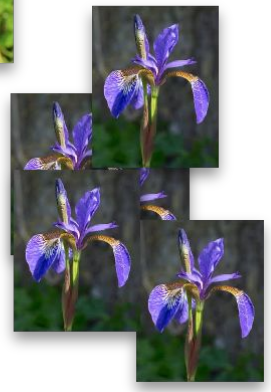


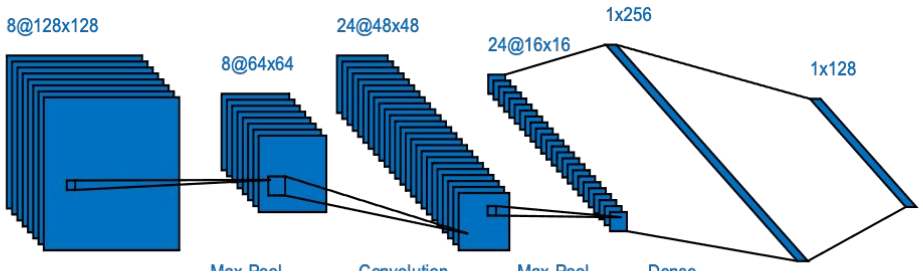
Hand Crafted feature extraction using image processing

Automating Feature Extraction through Deep Learning: A Breakthrough in Machine Learning Advances

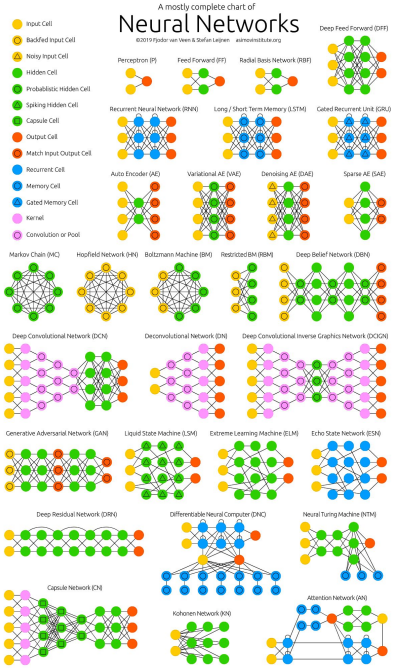
Deep learning(Features are unknown)

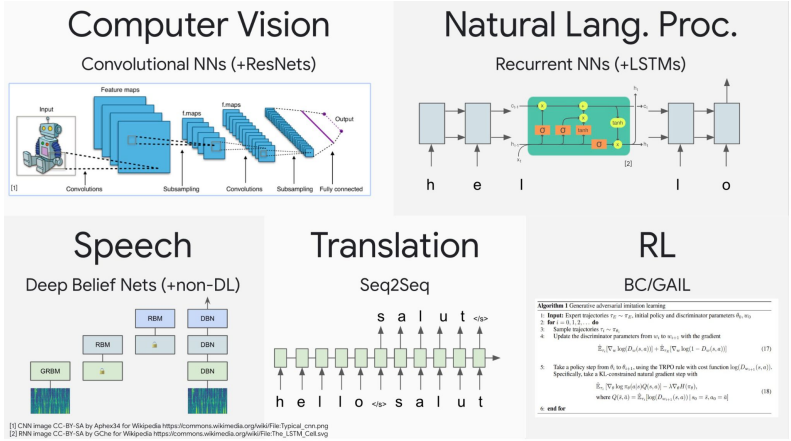
Deep Learning Algorithms automate the feature engineering by introducing the feature extraction layers





Before ~2020: each task had its own NN architecture





https://t.co/aYfnVKPDjT

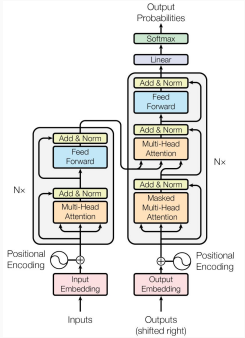
https://www.asimovinstitute.org/neural-network-zoo/

Now: All transformers

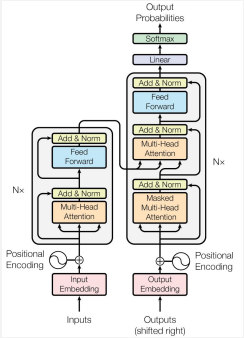
Attention is all you need(2017)

The transformer architecture, introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017, has revolutionized the field of natural language processing (NLP) and beyond

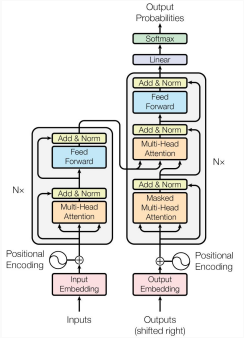
Computer Vision



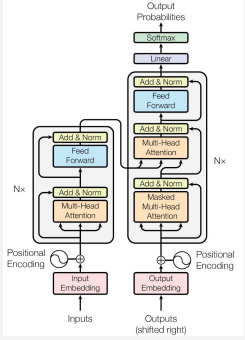
Natural Lang. Proc.



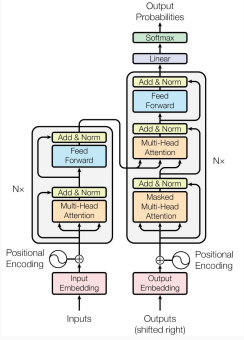
Reinf. Learning



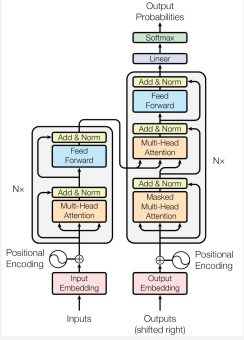
Speech



Translation



Graphs/Science



Transformer image source: "Attention Is All You Need" paper

What does it works so well?

Andrej Karpathy 
@karpathy

The Transformer is a magnificient neural network architecture because it is a general-purpose differentiable computer. It is simultaneously:
1) expressive (in the forward pass)
2) optimizable (via backpropagation+gradient descent)
3) efficient (high parallelism compute graph)

11:54 AM · Oct 19, 2022

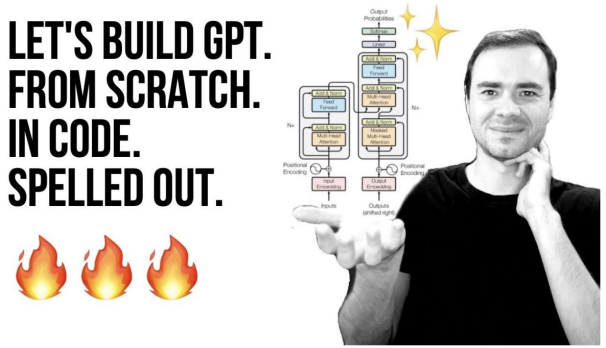
490 Retweets **39** Quotes **3,670** Likes **1,023** Bookmarks

Should you be able to code a Transformer?

Definitely not necessary!

BUT: it's not difficult, it is fun, and is probably worth doing.

Andrej Karpathy's GPT-2 implementation is <400 lines of code, including Attention and MLP blocks



Transformer Architecture

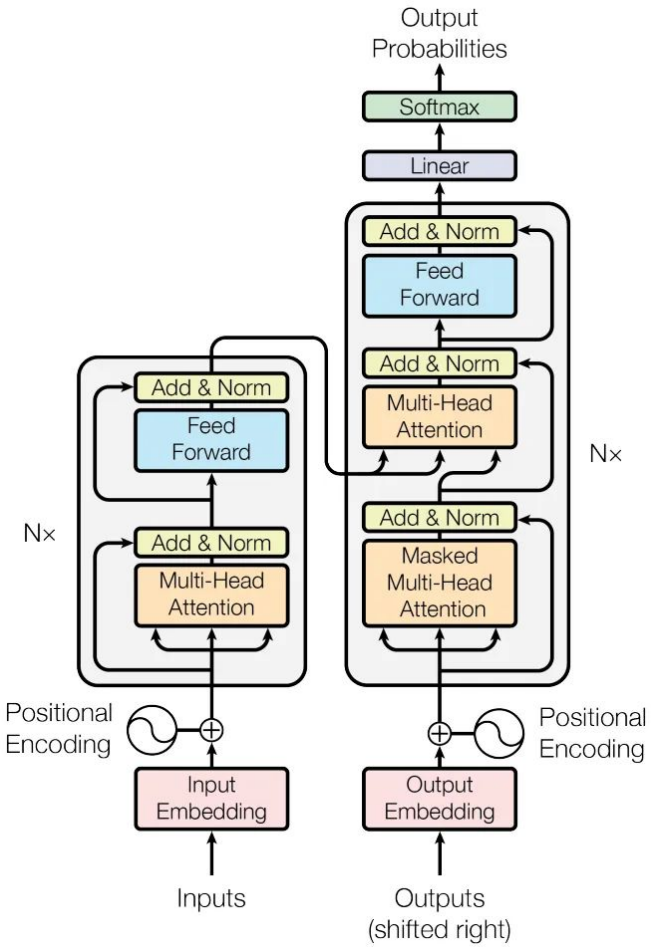
The transformer architecture is comprised of two major components: **feed-forward networks and self-attention.**

what is self-attention?: It takes n elements (or tokens) as input, transforms them, and returns n tokens as output.

It is a sequence-to-sequence module that, for each input token, does the following:

1. Compares that token to every other token in the sequence
2. Computes an attention score for each of these pairs
3. Sets the current token equal to the weighted average of all input tokens, where weights are given by the attention scores

Such a procedure allow the model to identifying the most relevant tokens. In other words, it asks the question: “Which tokens are worth paying attention to?” (hence, the name self-attention)



https://jalammar.github.io/illustrated-transformer/

Transformer Architecture

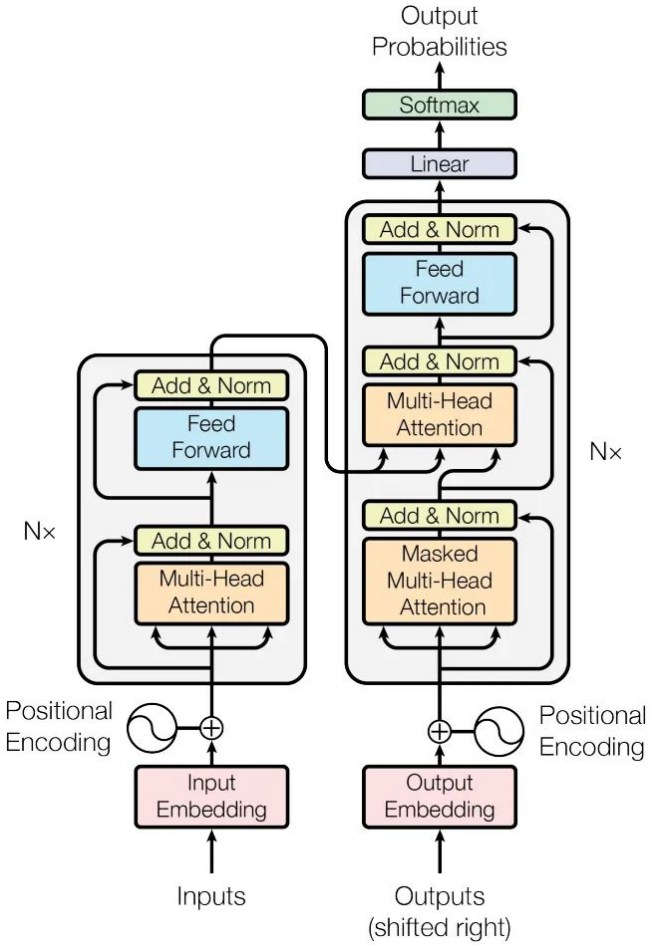
The transformer architecture is comprised of two major components: feed-forward networks and self-attention.

multi-headed self-attention. The variant of attention used in most transformers is slightly different than the description provided above. N

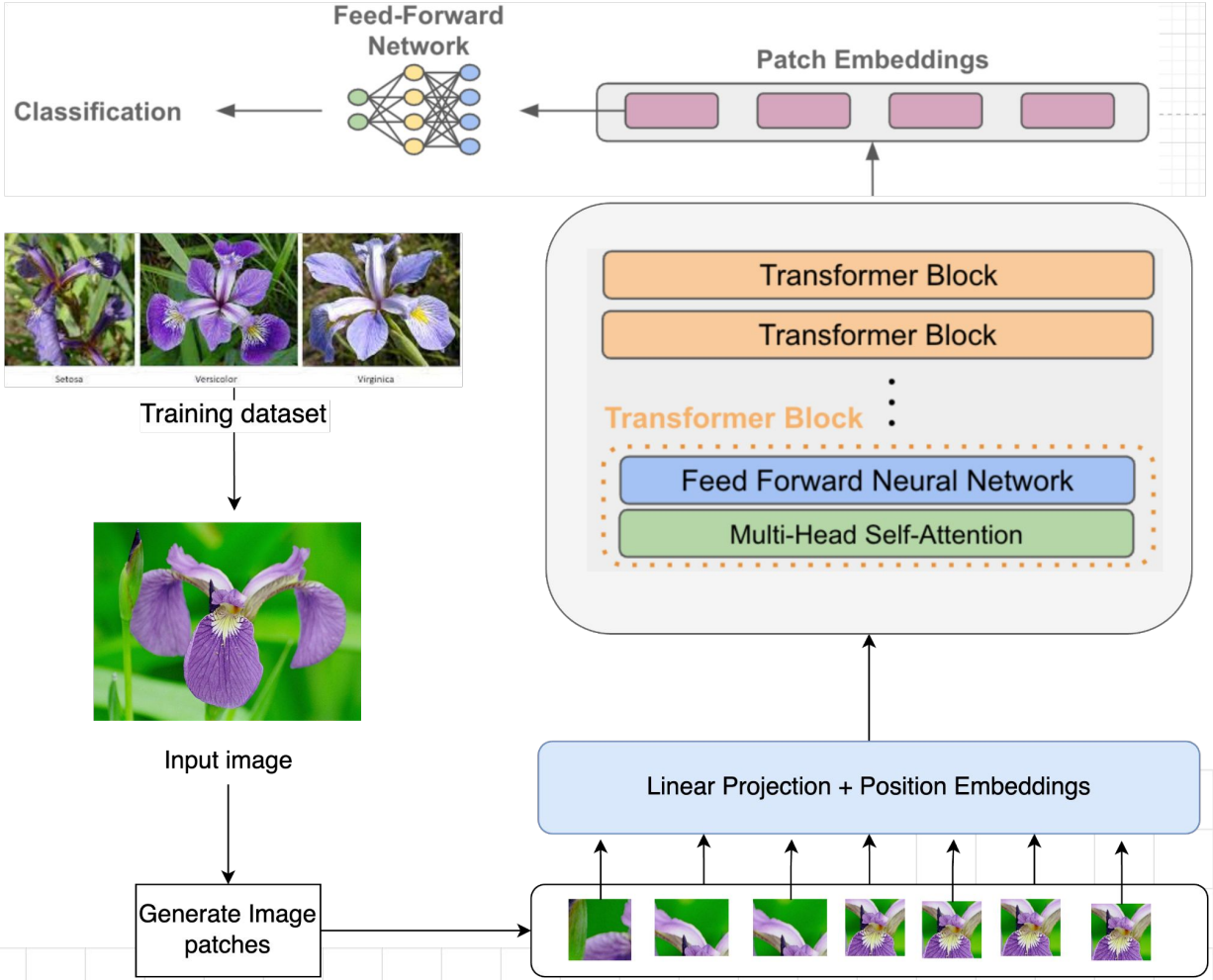
amely, transformers oftentimes leverage a “multi-headed” version of self attention. Although this may sound complicated, it’s not … at all.

Multi-headed self-attention just uses multiple different self-attention modules (e.g., eight of them) in parallel. Then, the output of these self-attention models is concatenated or averaged to fuse their output back together.

https://jalammar.github.io/illustrated-transformer/



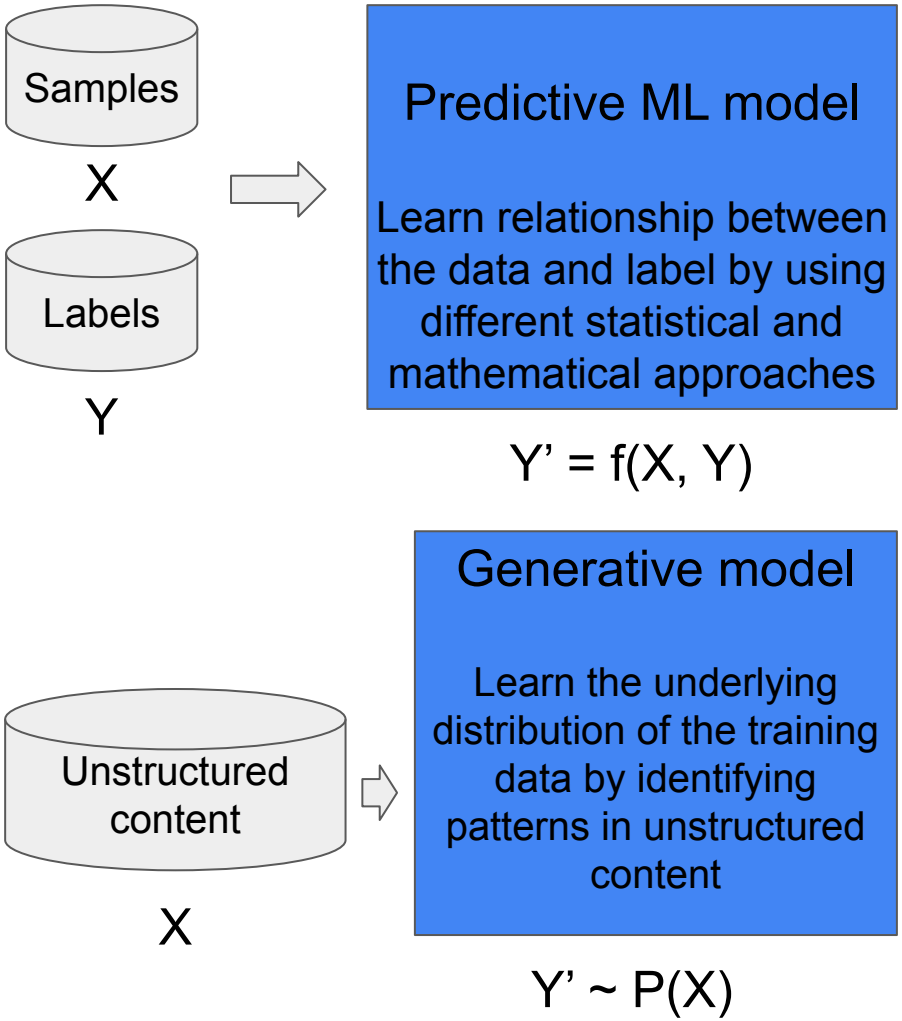
1. Convert an image into a sequence of flattened image patches
2. Pass the image patch sequence through the transformer model
3. Take the first element of the transformer's output sequence and pass it through a classification module



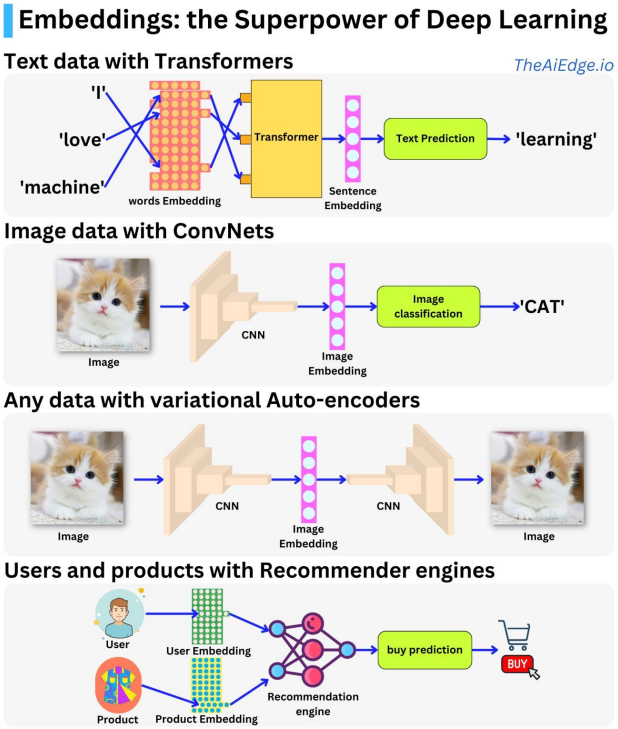
Experts

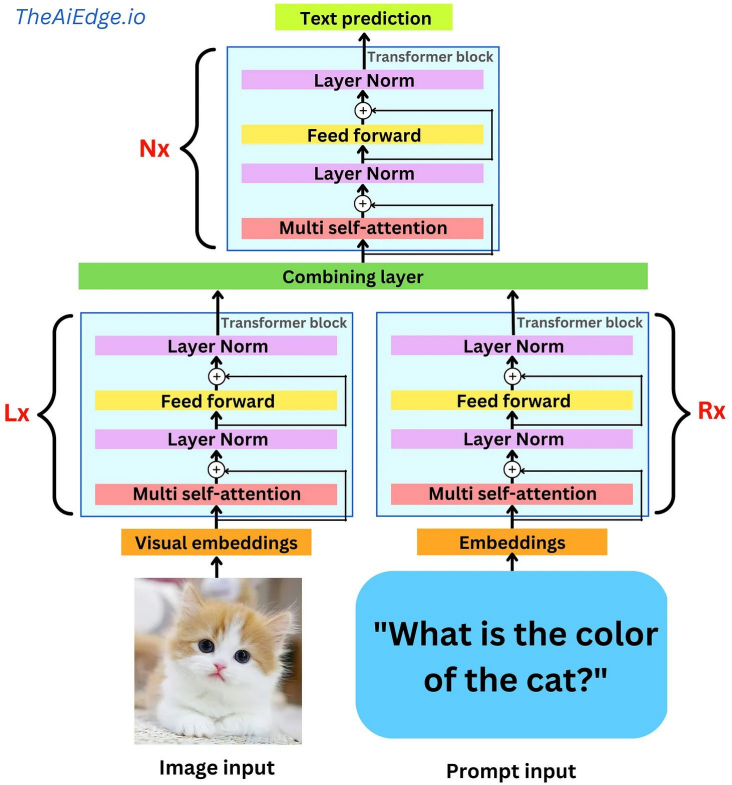
But, we can do more than just predict the value of a single variable (Distribution Sampling)

The new Gold: ML data generation techniques and use cases


Samples
X
Labels
Y
Predictive ML model
Learn relationship between the data and label by using different statistical and mathematical approaches
Y' = f(X, Y)
Trained Model (Probabilistic / Deterministic)
Y' is a:
Discrete(classification) or a continuous number (Regression)
It could be also a probability distribution
Based on what the model learns from the provided label and data pair, the model attempts to predict the value of the output variable Y
At the end we have a probabilistic or deterministic model that can do predictions on unseen inputs based on what it learn
Unstructured content
X
Generative model
Learn the underlying distribution of the training data by identifying patterns in unstructured content
Y' ~ P(X)
Trained Model (Probabilistic)
Y' is a:
Natural Language
Image
Audio
After learning the statistical properties of the training data distribution, the model generates a random sample from that distribution. Y` is expected to have the same statistical properties and structure as Y.

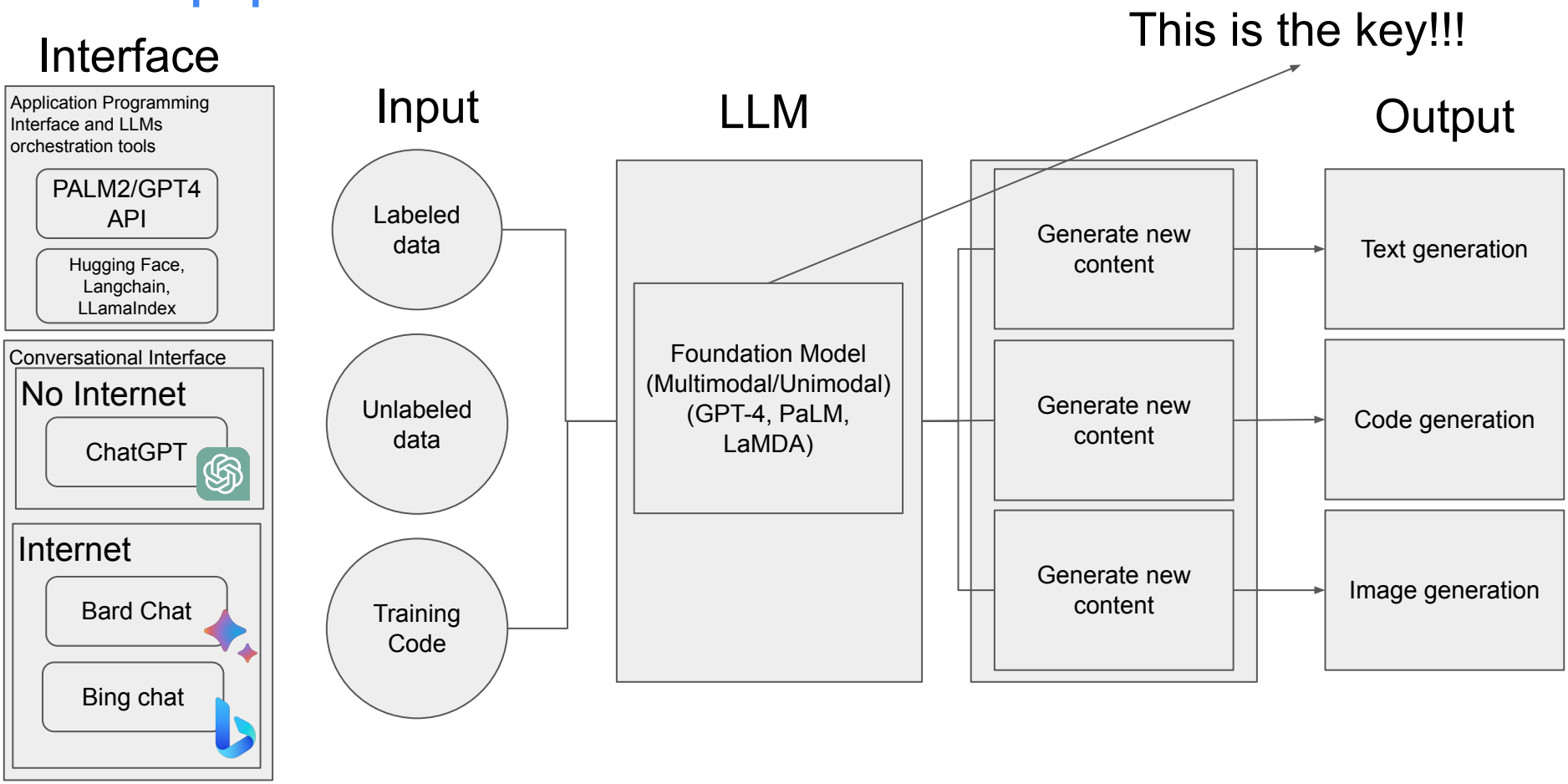
Modern Deep Learning



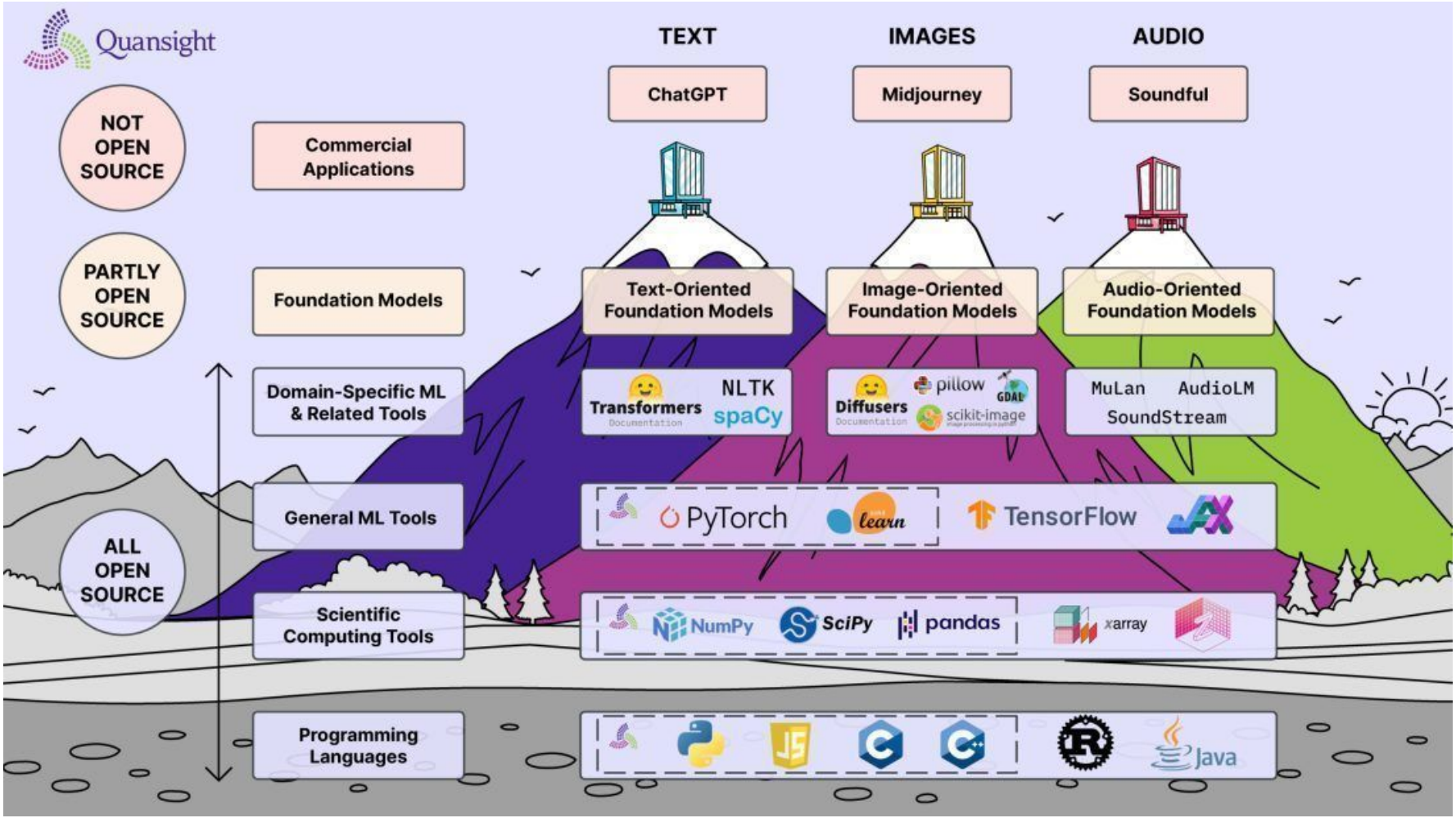


https://newsletter.theaiedge.io/

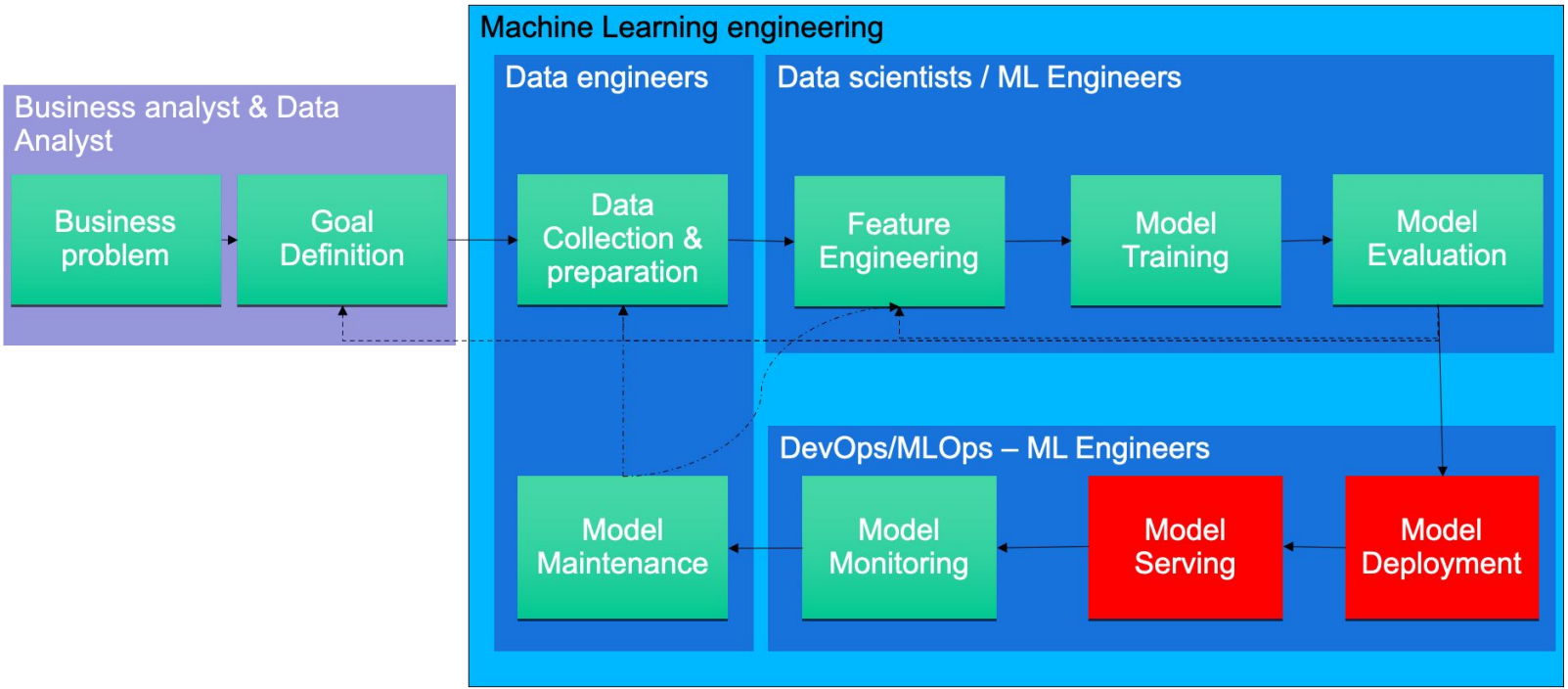
LLM pipeline




Google Cloud Platform
Azure
amazon web services™
IBM Watson™
Stanford Alpaca
Meta
VICUNA
Falcon meets LLama
TensorFlow
PyTorch
LangChain
Hugging Face Agents


Quansight
TEXT
IMAGES
AUDIO
ChatGPT
Midjourney
Soundful
NOT OPEN SOURCE
PARTLY OPEN SOURCE
ALL OPEN SOURCE
Commercial Applications
Foundation Models
Domain-Specific ML & Related Tools
General ML Tools
Scientific Computing Tools
Programming Languages
Text-Oriented Foundation Models
Image-Oriented Foundation Models
Audio-Oriented Foundation Models
Transformers
Documentation
NLTK
spaCy
Diffusers
Documentation
pillow
GDAL
scikit-image
MuLan
AudioLM
SoundStream
PyTorch
learn
TensorFlow
NumPy
SciPy
pandas
xarray
Java

ML Learning Pipeline



Setting up Machine Learning Projects

Based on the statistics, according to a 2019 Report, 85% of AI projects fail to deliver On their intended promise to business

Why do so many projects fail?

ML is still in research - you shouldn't aim for 100% success rate

But, many are doomed to fail

Technically infeasible or poor scoped

Never make the leap to production

Unclear success criteria

Poor team management

Lifecycle of a ML project

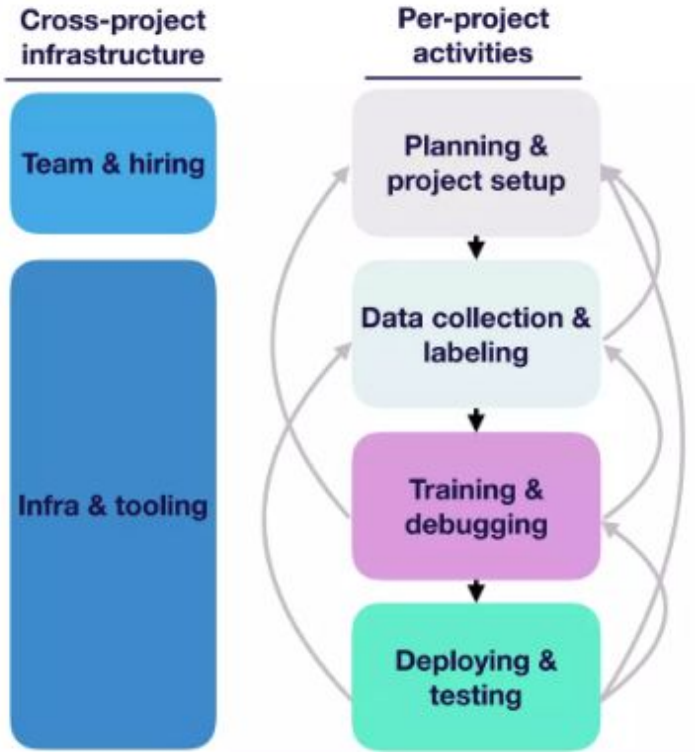
https://github.com/ageron/handson-ml/blob/master/ml-project-checklist.md

Phase 1 is **Project Planning and Project Setup**: At this phase, we want to decide the problem to work on, determine the requirements and goals, as well as figure out how to allocate resources properly.

Phase 2 is **Data Collection and Data Labeling**: At this phase, we want to collect training data (images, text, tabular, etc.) and potentially annotate them with ground truth, depending on the specific sources where they come from.

Phase 3 is **Model Training and Model Debugging**: At this phase, we want to implement baseline models quickly, find and reproduce state-of-the-art methods for the problem domain, debug our implementation, and improve the model performance for specific tasks.

Phase 4 is **Model Deployment and Model Testing**: At this phase, we want to pilot the model in a constrained environment, write tests to prevent regressions, and roll the model into production.



What also do you need to know?

Understand state of the art in your domain

Understand what is possible

Know what to try next

We will introduce most promising areas

Be aware that not the more complex solution will be the best, it will impact significantly your resources management plan.

ML is not alway the solution

Applied machine learning projects (ML Component)

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

What is the problem?

Informal description
Describe the problem as though you were describing it to a friend or colleague. This can provide a great starting point for highlighting areas that you might need to expand upon. It also provides the basis for a one sentence description you can use to share your understanding of the problem.
For example: I need a program that will tell me which tweets will get retweets.

Formalism
Tom Mitchell defines a useful formalism for describing a machine learning problem:
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Applied machine learning projects

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

Analyze the data

The objective of the data analysis step is to increase the understanding of the problem by better understanding the problems data. This involves providing multiple different ways to describe the data as an opportunity to review and capture observations and assumptions that can be tested in later experiments.
There are two different approaches I use to describe a given dataset:

1. Summarize Data: Describe the data and the data distributions.
2. Visualize Data: Create various graphical summaries of the data.

Applied machine learning projects

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

Prepare the data

The more disciplined you are in your handling of data, the more consistent and better results you are likely to achieve. The process for getting data ready for a machine learning algorithm can be summarized in three steps:

1. Select Data
2. Preprocess Data
3. Transform Data

Applied machine learning projects

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

Evaluate Algorithms

n this part you will step through a process to rapidly test algorithms and discover whether or not there is structure in your problem for the algorithms to learn and which algorithms are effective.
There are two considerations when evaluating algorithms:
1. Define your Test Harness
2. Spot Checking Algorithms

Applied machine learning projects

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

Improve results

When tuning algorithms you must have a high confidence in the results given by your test harness. This means that you should be using techniques that reduce the variance of the performance measure you are using to assess algorithm runs. I suggest cross validation with a reasonably high number of folds (the exact number of which depends on your dataset).
The three strategies you will learn about in this part are:

1. Algorithm Tuning
2. Ensembles
3. Extreme Feature Engineering

Applied machine learning projects

Problem Definition: Understand and clearly describe the problem that is being solved.
Analyze Data: Understand the information available that will be used to develop a model.
Prepare Data: Discover and expose the structure in the dataset.
Evaluate Algorithms: Develop a test harness and baseline accuracy from which to improve.
Improve Results: Leverage results to develop more accurate models.
Present Results: Describe the problem and solution so that it can be understood by third parties.

Presents results

Depending on the type of problem you are trying to solve, the presentation of results will be very different. There are two main facets to making use of the results of your machine learning endeavor:

1. Report the results
2. Operationalize the system

Applied machine learning projects (Software Solution)

Agile methodologies, traditionally used in software development to adapt to changing requirements and ensure quick delivery, can also be effectively applied to machine learning (ML) projects.

Implementing Agile in ML projects helps address challenges like rapidly changing data, evolving project requirements, and the iterative nature of model development and validation. Here are some Agile methodologies that can be adapted for ML projects:

Adaptations for ML Projects

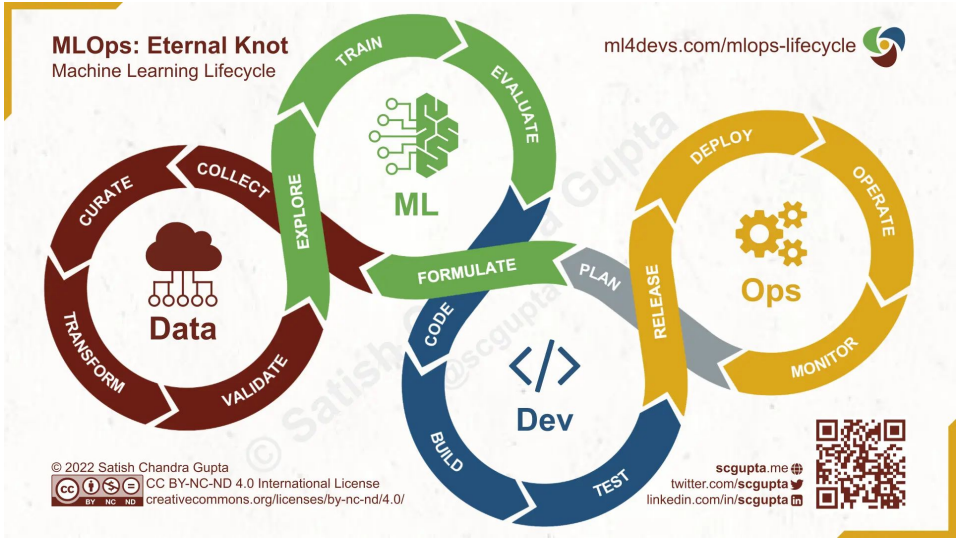
While these methodologies offer frameworks for managing work, ML projects may require specific adaptations:

Iterative Experimentation: Embrace the iterative nature of ML, where initial models often serve as baselines for further experimentation and refinement.

Flexible Planning: Allow for adjustments in project scope and direction based on intermediate results and discoveries.

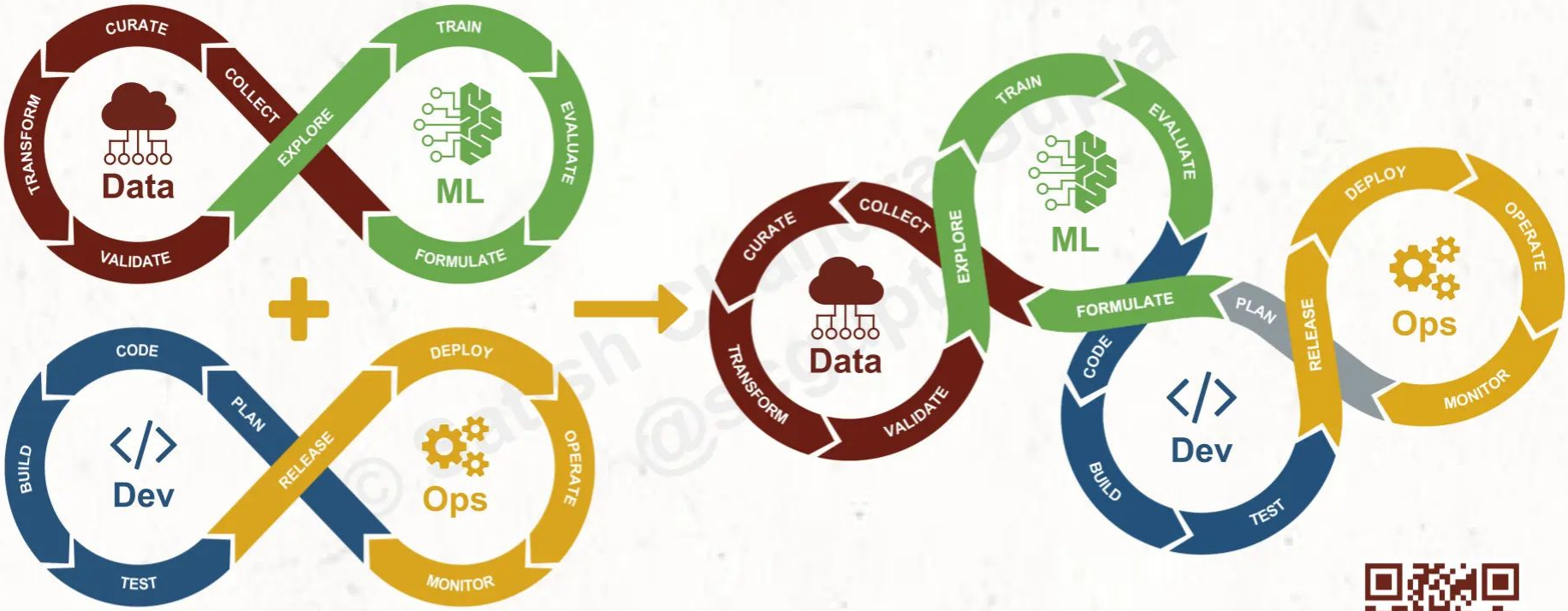
Model Versioning and Experiment Tracking: Implement tools and practices for tracking different model versions, experiment parameters, and results to ensure reproducibility and facilitate decision-making.

Collaboration between Data Scientists and Domain Experts: Foster close collaboration to ensure that models are developed with a deep understanding of the domain and are aligned with business needs.



MLOps = DataML + DevOps

ml4devs.com/mlops-lifecycle



scgupta.me
twitter.com/**scgupta**
linkedin.com/in/**scgupta**

Software Methodologies

Waterfall	Agile	Scrum	XP(Extreme Programming)
• Most traditional and sequential choices. • Linear sequential approach • Each stage must be completed before moving on to the next • Best suited for projects with well-defined and stable requirements	• An iterative and incremental approach • Emphasizes teamwork, adaptability, and customer satisfaction • Used in rapidly changing environments and for projects with unclear requirements • Using the Agile approach, teams develop in short sprints or iterations, each of which includes a defined duration and list of deliverables, but in no particular order.	• Another way to implement the Agile approach, Scrum borrows from Agile's foundational beliefs and philosophy that teams and developers should collaborate heavily and daily. • Uses sprints, roles (e.g. product owner, scrum master, development team), and ceremonies (e.g. daily standup, sprint review, sprint retrospective) • Agile framework for managing and completing complex projects • Best suited for projects with rapidly changing requirements or in complex environments	• Agile methodology for software development • Emphasizes communication, simplicity, feedback, and courage • Practices include pair programming, test-driven development and continuous integration and deployment

1. Scrum

Scrum is a popular Agile framework that organizes work into small, manageable pieces delivered in short cycles called sprints, typically lasting 2-4 weeks. For ML projects, Scrum can facilitate rapid experimentation and iteration. Teams can define sprints for different phases of ML development, such as data preparation, model training, evaluation, and deployment. Daily stand-ups can help track progress and address blockers quickly.

2. Kanban

Kanban emphasizes continuous delivery without overburdening the team, using a visual board to track work items through various stages of completion. In ML projects, Kanban can be used to manage the flow of tasks like data annotation, feature engineering, model experimentation, and performance tuning. Its flexibility is particularly useful for projects where priorities shift frequently based on experimental results or business needs.

3. Extreme Programming (XP)

XP focuses on technical excellence and high customer involvement, with practices like pair programming, test-driven development (TDD), and frequent releases. While some XP practices may not directly translate to ML projects (e.g., TDD is challenging due to the probabilistic nature of ML outcomes), the emphasis on quality and collaboration can be beneficial. For instance, pair programming can be adapted for collaborative model development and code reviews, ensuring high-quality code and model architecture.

4. Lean Software Development

Lean principles focus on maximizing customer value while minimizing waste. Applied to ML, this means rapidly delivering models that provide value, eliminating unnecessary complexities in data processing or model architecture, and iterating based on feedback. Lean encourages a build-measure-learn loop, ideal for experimenting with different models, features, and hyperparameters to find the most effective solutions.

5. Feature-Driven Development (FDD)

FDD combines best practices from other methodologies, focusing on developing and delivering client-valued features in short iterations. In the context of ML, features could be interpreted as model functionalities or capabilities. FDD can be adapted to prioritize ML tasks that directly contribute to the project's goals, ensuring that development efforts align closely with business requirements.

Google I/O Extended
North America

August 31st, 2023 10:00 AM PT

Large Language Models and the Future of Programming

Peter Norvig
Director of Research, Google


kubernetes
THE DOCUMENTARY
by Honeypot
PART 1