The Artificial Intelligence Journey

Version 4.0
June-2025

By Dr. Shlomi Boutnaru

Created using Craiyon, AI Image Generator

Table of Contents

Introduction

Artificial Intelligence has become an integral part of our lives. Think about chatbots, image recognition, NLP (natural language processing), voice assistants, GenAI and more. Thus, it is something that many folks want to understand more from a technical perspective.

Overall, I wanted to create something that will improve the overall knowledge regarding Artificial Intelligence using writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (https://twitter.com/boutnaru). Also, you can read my other writeups on medium - https://medium.com/@boutnaru. Lastly, You can find my free eBooks at https://TheLearningJourneyEbooks.com.

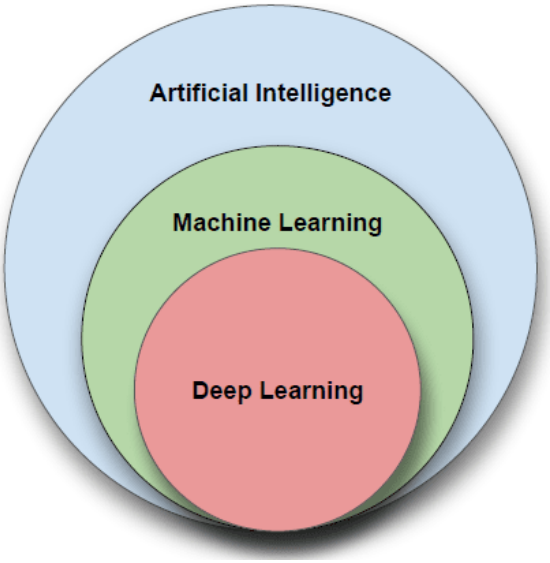
Lets GO!!!!!!

AI (Artificial Intelligence)

AI (Artificial Intelligence) is defined as the ability of machines/systems to enhance/replicate human intellect, think about reasoning and learning from experience. AI is based on the theory of probability, linear algebra and algorithms. We can break AI into two main fields: Machine Learning (ML) and Deep Learning (DL) - as shown in the diagram below[1].

In his paper "What is artificial intelligence?" from 2014 John McCarthy gave the following definition for "Artificial Intelligence": "It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable"[2]. By the way, John McCarthy is an American computer scientist which received in 1971 a "Turing Award" for his contribution to the field of AI[3].

Thus, we can say that the goal of AI is to preserve, synthesize and infer information by computer systems in order to solve problems, represent knowledge, process natural languages and more. Think about tasks such as: computer vision, speech recognition, language translation, summarizing text and more. It is believed that AI was founded as an academic discipline in 1956[4].



[1] https://www.red-gate.com/simple-talk/development/data-science-development/introduction-to-artificial-intelligence/
[2] https://www-formal.stanford.edu/jmc/whatisai.pdf
[3] https://amturing.acm.org/award_winners/mccarthy_1118322.cfm
[4] https://en.wikipedia.org/wiki/Artificial_intelligence

STT (Speech to Text)

STT (Speech to Text) is a technology used to convert spoken language into written text - as demonstrated in the diagram below[5]. By the way, it is also called “voice to text”. Among the challenges that STT technologies face we can find the following: background noise (like loud acoustic), privacy, security, integration with other tools, accents and dialects. STT can help improve productivity, improve communication, enhance collaboration across teams even in case of language barriers and more[6].

Overall, examples of use-cases relevant for STT are (but not limited to): transcription services, call center automation, voice noting and dictation, real-time captioning, translation, voice keyword search, speech analytics and accessibility. There are different open-source solutions for STT like (but not limited to): “Wav2letter, Kaldi”, “SpeechBrain”, “Whisper ASR” and “DeepSpeech”. Also, there are API based alternatives such as (but not limited to): “Google STT”, “Microsoft Azure STT” and “Amazon Transcribe”[7].

Lastly, the key phases in STT are: pre-processing (like noise reduction and echo cancellation), feature extraction (such as frequency and amplitude), acoustic modeling (like mapping extracted features to phonemes), language modeling decoding and post-processing[8]. Today, STT is based on different technologies like NLP[9], LLMs[10] and more.

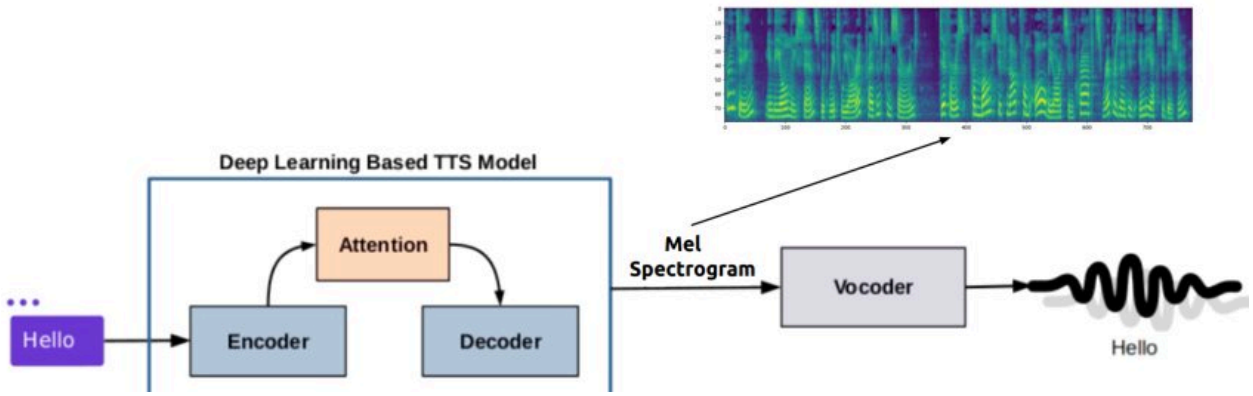
[5] https://hub.ftech.ai/en/blog/speech-to-text-advanced-solutions-for-digital-communication-editing-and-experiences/
[6] https://aiola.ai/glossary/speech-to-text-stt/
[7] https://www.gladia.io/blog/introduction-to-speech-to-text-ai
[8] https://www.gladia.io/blog/introduction-to-speech-to-text-ai
[9] https://medium.com/@boutnaru/the-artificial-intelligence-journey-nlp-natural-language-processing-41ae7d1d4428
[10] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6

TTS (Text-to-Speech)

TTS (Text-to-Speech) is a technology focused on converting text into spoken audio. Examples of such use-cases (but not limited to) are reading books\websites\PDFs\etc. Thus, it allows us to access written content in auditory format. By the way, it is also referred to as speech synthesis, a transformative technology that leverages AI to transform written text into lifelike spoken words[11].

Overall, TTS technologies have been evolving over the last decade. It can help in different use-cases such as (but not limited to): generate speech in dozens of languages, create media at a fraction of the cost and more[12]. There are two distinct ways for performing real-time TTS: "End-to-End TTS" (all processing is done in one phase) and "Two Stages TTS" (the text input is first converted to a spectrogram which is used to generate the audio) - as shown in the diagrams below[13].

Lastly, we can think about it as the opposite of STT[14]. Examples of open-source TTS models are (but not limited to): "Chatterbox TTS", "Dia-1.6B", "Kokoro-82M", "XTTS-v2" and "SpeechT5"[15].



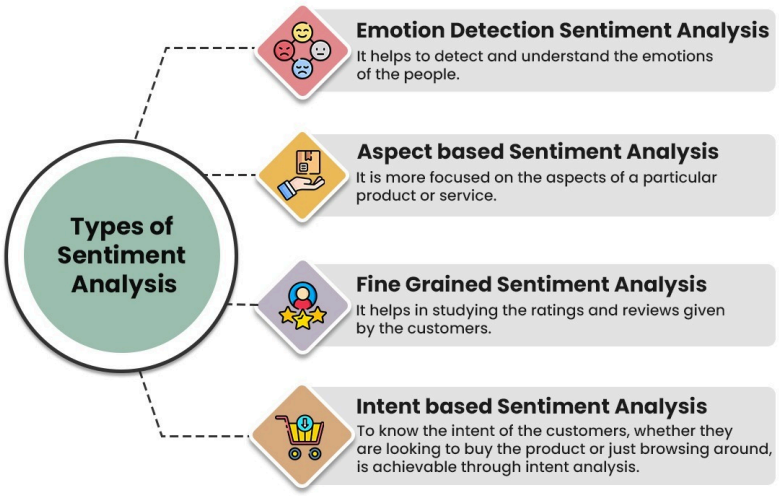
[11] https://www.naturalreaders.com/online/
[12] https://aws.amazon.com/polly/what-is-text-to-speech/
[13] https://www.gnani.ai/resources/blogs/an-essential-guide-to-real-time-text-to-speech-tts-systems/
[14] https://medium.com/@boutnaru/the-artificial-intelligence-journey-stt-speech-to-text-6546aaefce79
[15] https://huggingface.co/models?pipeline_tag=text-to-speech&sort=trending

Sentiment Analysis (Opinion Mining)

"Sentiment Analysis" (aka opinion mining) is the process of analyzing a large volume of text in order to decide if it expresses positive\negative\neutral sentiment. Sentiment analysis uses NLP[16] technologies for interpreting text like humans. There are different approaches for implementing sentiment analysis such as (but not limited to): rule based (like searching for positive\negative words), machine learning based (like linear regression, naive bayes, deep learning and more) and hybrid approach between the two[17].

Overall, there are different types of sentiment analysis as described in the diagram below. First, "Fine-Grained" (graded) which groups text into different emotions and the level of emotion being expressed (expressed in scale of 0-100). Second, "Aspect-Based" (ASBA) is focused on analyzing a singular aspect of a product\service\customer experience. Third, "Emotional Detection" which tries to understand the psychological state of specific text. Fourth, "Intent Based" is used for understating what a user wants to do[18].

Lastly, among the use-case in which sentiment analysis is used we can find: brand monitoring, market research, social media monitoring, tracking campaign performance, improving customer service and more[19]. It is important to understand that there are different challenges in sentiment analysis like (but not limited to): slang and informal language, data imbalance issues, data privacy concerns, real-time analysis demands, contextual understanding shifts, machine translation hurdles and irony\sarcasm interpretation[20].



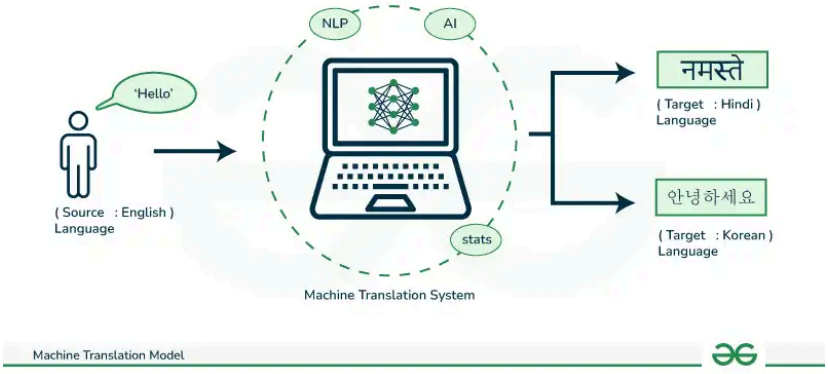
[16] https://medium.com/@boutnaru/the-artificial-intelligence-journey-nlp-natural-language-processing-41ae7d1d4428
[17] https://www.ibm.com/think/topics/sentiment-analysis
[18] https://itechindia.co/blog/what-is-sentiment-analysis-and-which-businesses-need-sentiment-analysis/
[19] https://aws.amazon.com/what-is/sentiment-analysis/
[20] https://convin.ai/blog/sentiment-analysis-example-best-practices

Text Translation

Translation is the process of converting a text from one language to another while preserving its overall meaning - as shown in the diagram below[21]. While doing so we need to balance between making the text natural in the target language and staying true to the original meaning[22]. On one hand, traditional machine translation tools are based on phrase matching\statistical rules for converting data between languages. On the other hand, LLMs[23] can generate translations by predicting what words should come next based on context[24].

Overall, text translation can be used for different use-cases such as (but not limited to): translating web\app\communication materials, contract translation and management, social media content translating, customer support and automated post-editing[25]. LLMs can outperform traditional translation tools due to different reasons like (but not limited to): more contextual understanding (they can grasp a broader context), cultural nuance and adaptability[26]. However, we also need to understand LLMs can have disadvantages like hallucinations or made-up content.

Lastly, examples of AI models for performing text translations are: Sarvam-Translate[27], PLaMo Translation Model[28], T5 base[29] and MADLAD-400-3B-MT[30].



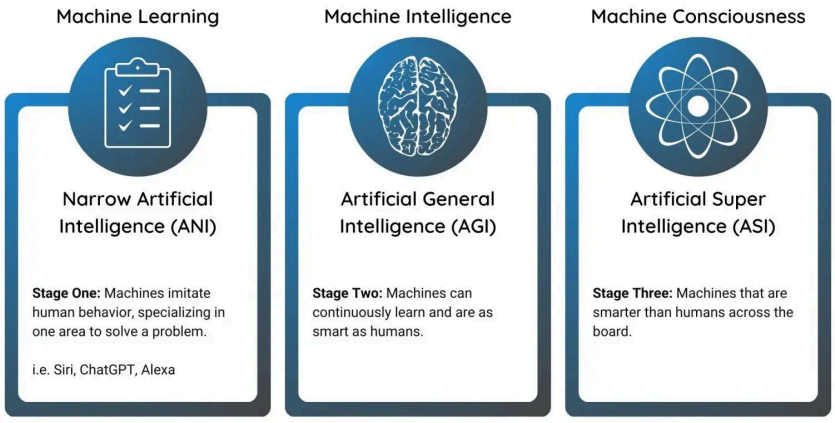
[21] https://botpenguin.com/blogs/discovering-the-major-applications-of-mistral-llm
[22] https://phrase.com/blog/posts/translation/#what-is-translation
[23] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[24] https://lokalise.com/blog/can-llm-translate-text-accurately/
[25] https://www.smartling.com/blog/llm-translation
[26] https://lokalise.com/blog/what-is-the-best-llm-for-translation/
[27] https://www.sarvam.ai/blogs/sarvam-translate
[28] https://tech.preferred.jp/ja/blog/plamo-translate/
[29] https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html
[30] https://arxiv.org/abs/2309.04662

Stages of Artificial Intelligence

AI (Artificial Intelligence) is a computer system that can perform complex tasks that would otherwise require human minds (visual perception\speech recognition\decision-making\etc). We can classify AI to three different types\level\stages: ANI (Artificial Narrow Intelligence), AGI (Artificial General Intelligence) and ASI (Artificial Super Intelligence)[31].

Overall, ANI (also called "Weak AI") is used for specific tasks while AGI (also called "Strong AI") has a human-like intelligence and can do many things at once. In the case of ASI, it is smarter than the human mind and can perform any task better[32] - as shown in the diagram below.

Lastly, it does not matter what stage\type\level of AI we are using, we must ensure it functions in a safe and ethical manner[33]. We can summarize that these categories differ in terms of intelligence\capabilities and thus also in the potential to influence the world[34]. As for 2025 ANI is what we call today AI, while AGI and ASI are still currently theoretical[35].



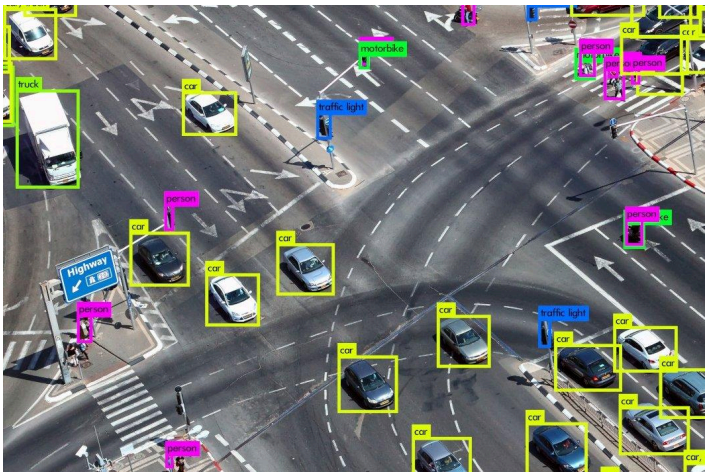
[31] https://www.ediweekly.com/the-three-different-types-of-artificial-intelligence-ani-agi-and-asi/
[32] https://viso.ai/deep-learning/artificial-intelligence-types/
[33] https://youevolve.net/ani-agi-and-asi-what-do-they-mean/
[34] https://arbisoft.com/blogs/understanding-the-levels-of-ai-comparing-ani-agi-and-asi
[35] https://www.coursera.org/articles/what-is-artificial-narrow-intelligence

ANI (Artificial Narrow Intelligence)

ANI (Artificial Narrow Intelligence) is sometimes also called “Weak AI”. This type of AI is focused on performing specific tasks based on given instructions. Such tasks are: language translation, speech recognition, driving cars autonomously, NLP (Natural Language Processing), facial recognition, chat bots, people counting, object detection and more - as shown in the image below[36].

Overall, examples of known ANI tools\technologies are: Siri, Google Translate, ChatGPT, Dall-E and AlphaFold[37]. ANI has different limitations like: lack of general intelligence, ethical concerns, limited learning capabilities, dependency on data quality and vulnerability to malfunctions[38].

Lastly, ANI has also benefits\advantages such as (but not limited to): relieves humans from mundane tasks, performs single tasks better than humans and provides faster decision making[39]. As for 2025 ANI is what we call today AI, while AGI (Artificial Generic Intelligence) and ASI (Artificial Super intelligence) are still currently theoretical[40].



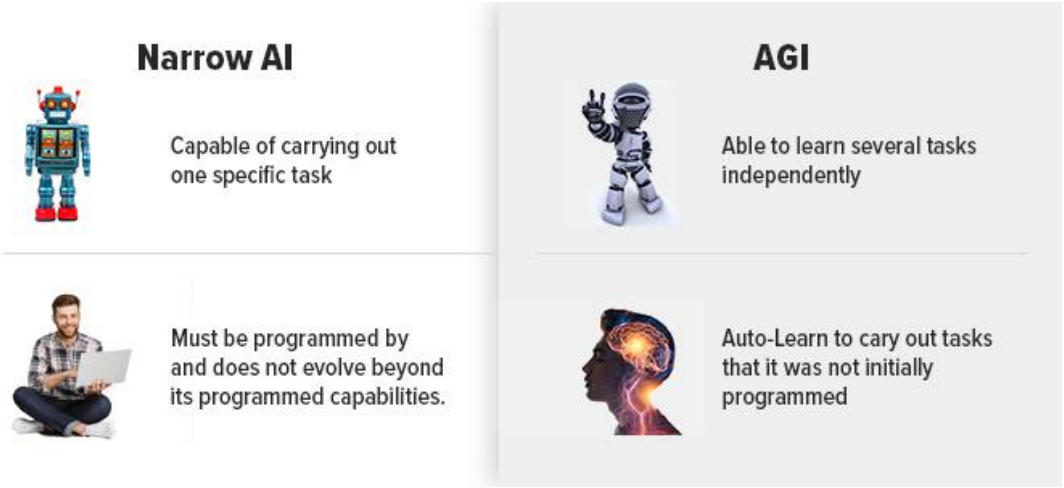
[36] https://viso.ai/deep-learning/artificial-intelligence-types/
[37] https://venturebeat.com/ai/what-is-artificial-narrow-intelligence-ani/
[38] https://www.appliedaicourse.com/blog/artificial-narrow-intelligence/
[39] https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-narrow-ai/
[40] https://www.coursera.org/articles/what-is-artificial-narrow-intelligence

AGI (Artificial General Intelligence)

AGI (Artificial General Intelligence) is a field of theoretical "Artificial Intelligence"[41] research. AGI's goal is to to create software with human-like intelligence with the ability to self-teach. Thus, we want software to be able to perform tasks that it is not necessarily trained and\or developed for[42].

Overall, AGI is sometimes also called "Strong AI" and\or "Deep AI"[43]. IBM demonstrated seven critical skills that current AI struggles with and AGI would need to master: navigation, problem-solving, creativity, fine motor skills, audio perception, visual perception and social\emotional engagement[44].

Lastly, as opposed to ANI (which is used for one specific task) AGI is able to learn several tasks and can auto-lean without the need to be programmed for every task - as shown below[45]. As for 2025 ANI is what we call today AI, while AGI is still currently theoretical[46].



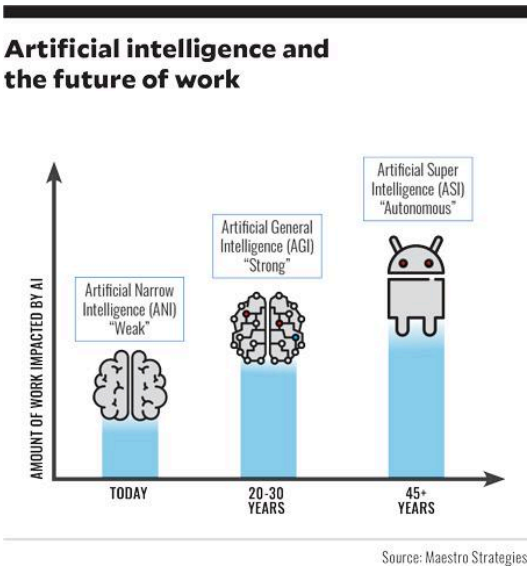
[41] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[42] https://aws.amazon.com/what-is/artificial-general-intelligence/
[43] https://www.spiceworks.com/tech/artificial-intelligence/articles/narrow-general-super-ai-difference/
[44] https://www.ibm.com/think/topics/artificial-general-intelligence-examples
[45] https://insights.daffodilsw.com/blog/all-about-artificial-general-intelligence-the-next-frontier-in-ai
[46] https://www.coursera.org/articles/what-is-artificial-narrow-intelligence

ASI (Artificial Super Intelligence)

ASI (Artificial Super Intelligence) is a hypothetical software-based artificial intelligence[47]. It is focused on providing intellectual scope beyond human intelligence. ASI is also called "Strong AI"[48].

Overall, ASI represents the highest level\stage of AI development and should surpass human intelligence in all aspects. By the way, while ASI far exceeds AI's current capabilities, many experts believe that its creation is inevitable and of course only possible[49].

Lastly, in 2020 a management consulting company (Maestro Strategies) anticipated that AGI[50] is going to be achieved in 20-30 years and ASI in about 45+ years[51].



[47] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[48] https://www.ibm.com/think/topics/artificial-superintelligence
[49] https://builtin.com/artificial-intelligence/asi-artificial-super-intelligence
[50] https://medium.com/@boutnaru/the-artificial-intelligence-journey-agi-artificial-general-intelligence-9b70184dc9c5
[51] https://www.hfma.org/technology/artificial-intelligence/artificial-intelligence-5-realities-for-financial-leaders/

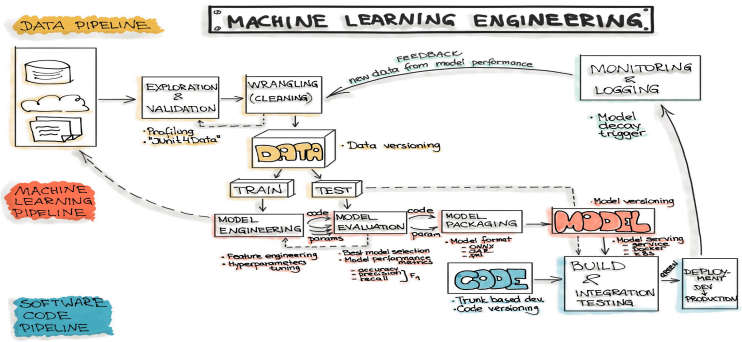
ML (Machine Learning)

Machine Learning is a subset/branch of artificial intelligence[52] which leverages the use of statistical methods in order to train algorithms for making classifications and/or peridications. This provides the ability to uncover key insights in data mining projects. This can be done using different frameworks like PyTorch and TensorFlow[53].

Due to the fact, we have not talked yet about “Deep Learning” (I will do that on future writeups) so for now we are going to focus on classical/”non deep” learning. Thus, machine learning is more dependent on human intervention like: selection of features, data cleaning and more. Think about the process of feature engineering which can include adding/mutating/combining data within the data set by experts for improving the results of the machine learning models[54].

Overall, we can categorize machine learning to four main types: “Supervised Learning”, “Unsupervised Learning”, “Semi-Supervised Learning” and “Reinforcement Learning”. “Supervised Learning” is when we have a pre-labeled/classified dataset (like by users) which allows a machine learning model to measure its performance. “Unsupervised Learning” is when we are using raw datasets which are not labeled, in this case we try to find patterns/relations in the data without the user's help.

Moreover, “Semi-Supervised Learning” we have a dataset which has both labeled and unlabeled data which allows the models to learn how to label unlabeled data. “Reinforcement Learning” uses AI agents that try to find an optimal way to perform a specific task, when they take an action that goes towards the goal they get a reward[55]. More on those types in future writeups. Lastly, there are different phases in the machine learning pipeline like: data collection, data clearing, training a model, testing a model deploying it and more - as shown in the diagram below[56]. By the way, well known machine learning algorithms that you might have heard of are: decision trees, neural networks, linear regression, logistic regression SVM and more.



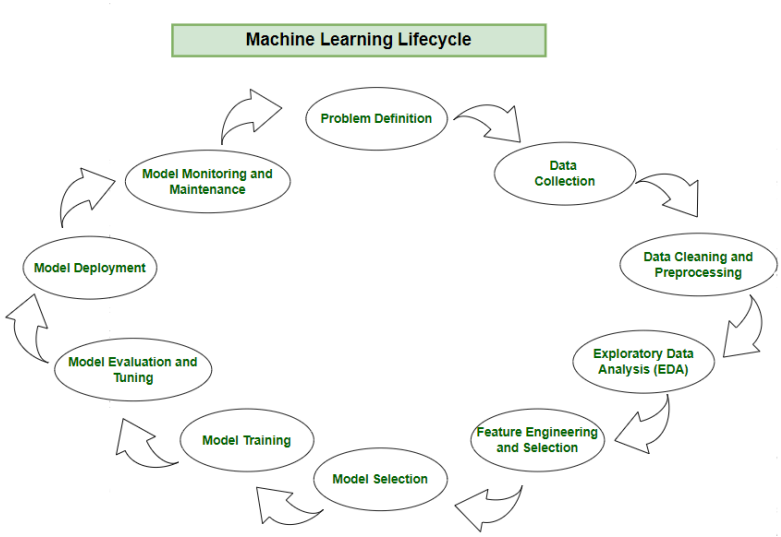
[52] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[53] https://www.ibm.com/topics/machine-learning
[54] https://www.snowflake.com/guides/feature-extraction-machine-learning
[55] https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/
[56] https://ml-ops.org/content/end-to-end-ml-workflow

Machine Learning Lifecycle

The “Machine Learning Lifecycle” is a flow/process which guides us how to develop and deploy machine learning models. The reason for having such a process is to help us solve complex problems. The lifecycle is composed of different stages each focused on specific areas of the overall problem[57] - as shown in the diagram below.

Overall, the we can use the following states: problem definition, data collection, data cleaning and preprocessing, exploratory data analysis, feature engineering and selection, model selection, training, evaluation and tuning, deployment and monitoring & maintenance[58] - more on each phase in future writeups.

Lastly, the number of stages/phases is not mandatory and not set in stone. Thus, different people can merge/separate different stages/phases like: planning, data preparation, model engineering, model evaluation, model deployment and monitoring & maintenance[59]. We can of course select anyone of those (or others).



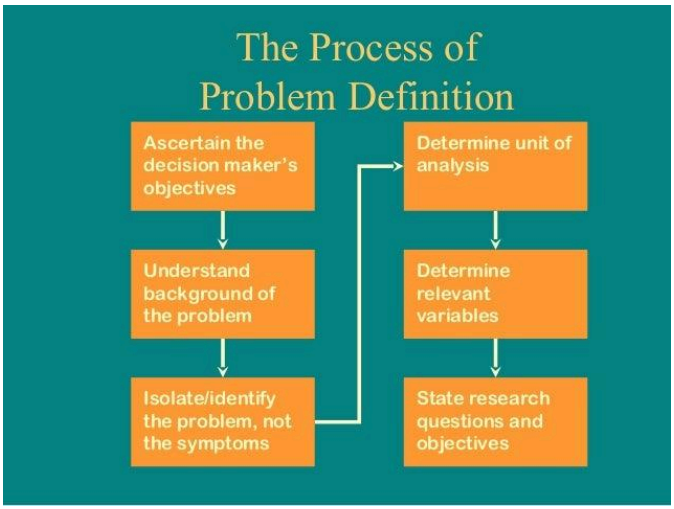
[57] https://www.geeksforgeeks.org/machine-learning-lifecycle/
[58] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[59] https://www.datacamp.com/blog/machine-learning-lifecycle-explained

Problem Definition

As with every problem we want to solve we first need to define it clearly. It is important in order to understand the problem so we can select the correct model to use. Thus, doing so clarifies the goal, determinants, data needs and influences model selection[60]. For example do we need to use supervised learning[61], unsupervised learning[62] or anything else.

Overall, this is the first phase of the machine learning cycle. The understanding\learning from this phase helps us understand what type of data we need to collect in the next phase aka "Data Collection" phase[63].

Lastly, the process of problem definition can include the following steps: identifying the decision maker's objective, understanding the background of the problem, focusing on the problem and not the symptoms, determining the unit of analysis, identifying relevant variables and stating research questions and objectives[64] - as shown in the diagram below.



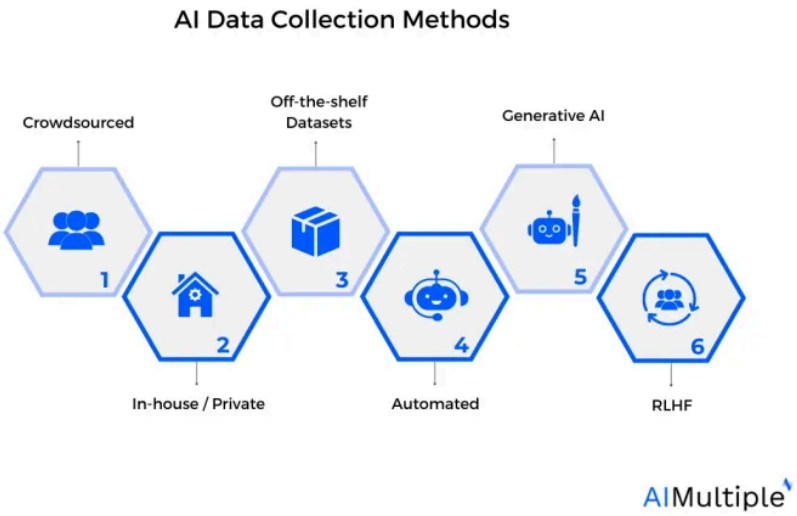
[60] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[61] https://medium.com/@boutnaru/the-artificial-intelligence-journey-supervised-learning-4a5aaf298275
[62] https://medium.com/@boutnaru/the-artificial-intelligence-journey-unsupervised-learning-a6a813b19f5b
[63] https://medium.com/@boutnaru/the-artificial-intelligence-journey-machine-learning-lifecycle-f74b70c4d136
[64] https://www.slideshare.net/alishare/problem-definition-and-research-proposalbrm

Data Collection

After understanding and defining the problem[65] we want to solve problems with machine learning. It is time to collect the data we are going to use for training. It is important to understand that the quantity and quality of the data hugely impacts the performance of the machine learning model. Hence, data collection is a methodical process for acquiring informational elements that are organized as a dataset[66].

Overall, there are different data sources that we can leverage for collecting data such as (not limited to): web scraping, public datasets (platforms like Kaggle or UCI) and internal databases of the company (like transaction logs, customer's data, company records and more). When collecting those (and others) we need to ensure the relevance of the data collected to the problem we want to solve[67].

Lastly, we can use different collection methods like (but not limited to): crowdsourcing (assigning data collection task to the public), in-house data collection (AI\ML developer collecting their own data), off-the-shelf datasets, automated data collection (for example web scraping, web crawling and using APIs), leveraging GenAI tools for generating data and RLHF (reinforcement learning from human feedback) - as shown in the diagram below[68].



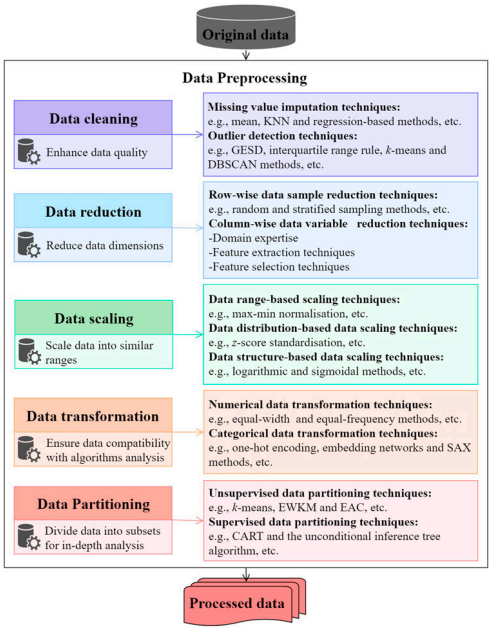
[65] https://medium.com/@boutnaru/the-artificial-intelligence-journey-problem-definition-301a387e2082
[66] https://www.altexsoft.com/blog/data-collection-machine-learning/
[67] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[68] https://research.aimultiple.com/data-collection-methods/

Data Cleaning and Preprocessing

After collecting data (we are going to use for training and\or validation of our machine learning model) there is a need to perform some cleaning and preprocessing operations to them[69]. This includes operations like handling missing values, formatting data and fixing different errors[70].

Overall, examples of such techniques are: dropping duplicate values, dropping outliers, encoding categorical features, splitting dataset into training set and a test set and scaling features (like normalization) - as shown below[71]. We could also need to handle an imbalanced dataset, in that case we can undersample the majority class and oversample the minority class[72].

Lastly, we can summarize that "Data Cleaning" identifies and rectifies errors\inconsistencies\missing values within a dataset. "Data Preprocessing" on the other hand transforms and optimizes the data for a specific analysis. Both can be performed greatly with the Pandas library using Python code[73].



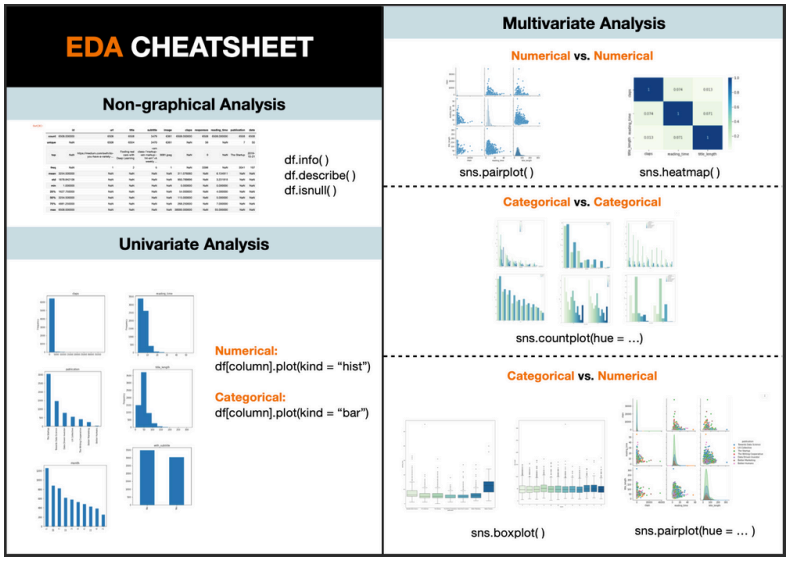
[69] https://medium.com/@boutnaru/the-artificial-intelligence-journey-problem-definition-301a387e2082
[70] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[71] https://www.lupon.gov.ph/data-preprocessing-machine-learning-visualization-using-jupyter-notebook-cc-Ze3yzrS1
[72] https://www.kdnuggets.com/2023/08/7-steps-mastering-data-cleaning-preprocessing-techniques.html
[73] https://www.freecodecamp.org/news/data-cleaning-and-preprocessing-with-pandasbdvhj/

Exploratory Data Analysis (EDA)

After cleaning the data and preprocessing it is time to perform an exploratory data analysis[74]. EDA (Exploratory Data Analysis) is leveraged for identifying patterns in the collected data. Hence, EDA uses different techniques such as summarization and visualization for discovering trends, relationships (like feature correlation) and potential inghits[75].

Overall, there are four major types of EDA. First, "Univariate Non-Graphical" which analyzes only one variable. Second, "Univariate Graphical" uses graphical methods for analyzing one variable only like stem and leaf plots, histograms and box plots. Third, "Multivariate Non Graphical" in which we analyze more than one variable that can surface relationships. Fourth, "Multivariate Graphical" that uses graphical tools for analyzing two or more variables such as: scatter plots, multivariate charts, run charts, bubulle charts and heatmaps[76].

Lastly, EDA allows stakeholders to confirm they are asking the correct questions and that the collected data is also relevant for them. Also, some of the most commonly used programming languages for EDA are "R" and "Python" - as shown in the Python's EDA cheat sheet below[77].



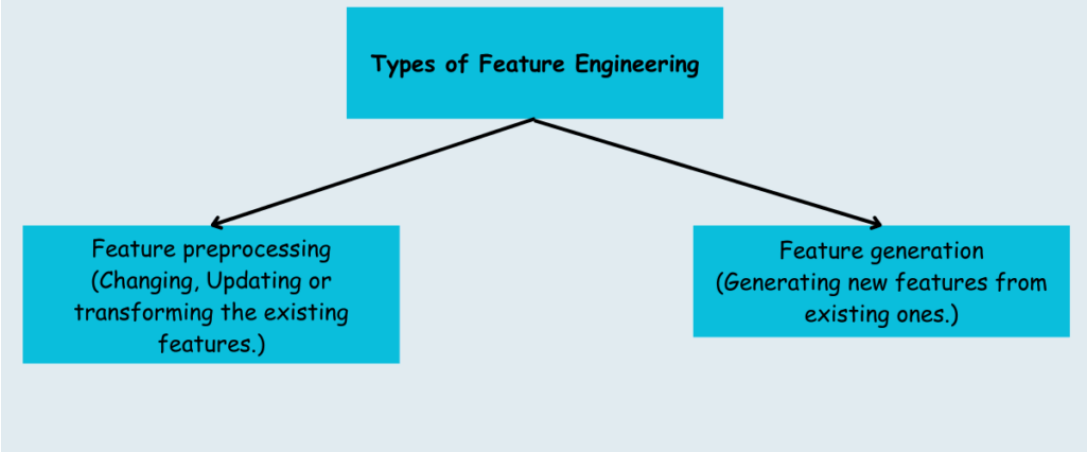
[74] https://medium.com/@boutnaru/the-artificial-intelligence-journey-problem-definition-301a387e2082
[75] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[76] https://www.ibm.com/think/topics/exploratory-data-analysis
[77] https://towardsdatascience.com/semi-automated-exploratory-data-analysis-eda-in-python-7f96042c9809/

Feature Engineering and Selection

Just after exploring the data[78] it is time for feature engineering and selection[79]. Features are the variables\attributes from the dataset we are going to train the model so it can perform classification\prediction tasks. This phase is based on two sub-steps. "Feature Engineering" that can create new features by splitting\combining\transforming variables. "Features Selection" which chooses the features that are most impactful on the model performance, hence part of that can be done only after assessing the performance of the model itself[80].

Overall, there are different techniques for feature engineering like (but not limited to): imputation (numerical\categorical), handling outliers (removal, replacing values, capping and discretization), scaling (normalization of values between 0 to 1 or z-score), one-hot encoding (elements of a finite site is represented by the index in that se) and log transform[81] - as shown below[82].

Lastly, "Feature Selection" is concerned with choosing the most relevant features to use for our model in order to enhance the model accuracy\insights\performance. There are supervised feature selection methods like: filter methods (information gain, mutual information, chi-square test and more), wrapper methods (forward selection, backward selection, RFE and more) and embedded methods (gradient boosting, random forest importance and more). Also, there are unsupervised feature selection methods such as: PCA (Principal Component Analysis), ICA (Independent Component Analysis) and autoencoders[83] - more on those in future writeups.



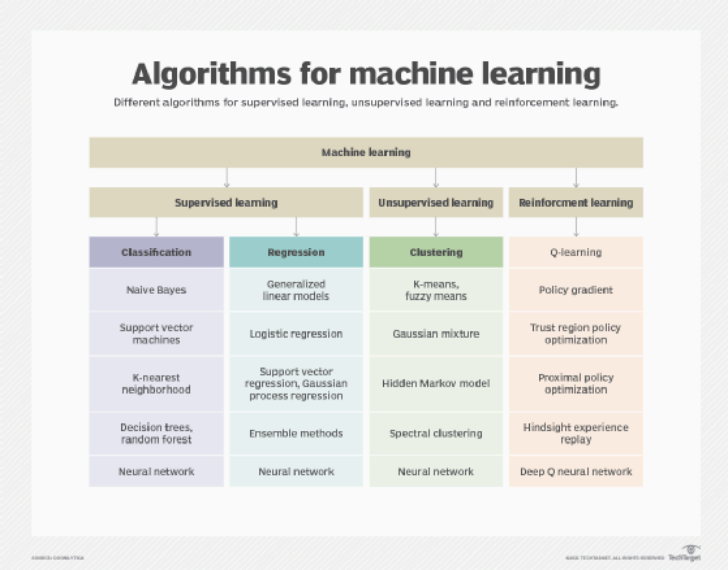
[78] https://medium.com/@boutnaru/the-artificial-intelligence-journey-exploratory-data-analysis-eda-103c223ed84e
[79] https://medium.com/@boutnaru/the-artificial-intelligence-journey-machine-learning-lifecycle-f74b70c4d136
[80] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[81] https://builtin.com/articles/feature-engineering
[82] https://www.askpython.com/python/examples/feature-engineering-in-machine-learning
[83] https://www.ibm.com/think/topics/feature-selection

Model Selection

The next phase after feature engineering and selection[84] is model selection[85]. The goal of this phase is to choose the most appropriate machine learning model that can solve the problem we are faced with[86]. Before selecting a model we need to ask different questions such as: Do we have labeled data or not? How much data do we have? Are we predicting a category? Are we predicting a quantity\number? Are we predicting a structure?[87].

Overall, this phase is needed due to the fact there are different types of machine learning models. Among those types are: "Supervised Learning"[88], "Unsupervised Learning"[89] and "Reinforcement Learning"[90]. Each of those has its advantages and disadvantages and thus use cases in which it is more relevant than others.

Lastly, examples for some use-cases are: leveraging clustering models (like K-means) for customer segmentation, regression models (like logistic\linear regression) for predicting house prices and reinforcement learning (like Q-learning and SARSA) models for robotics[91] - as shown below[92].



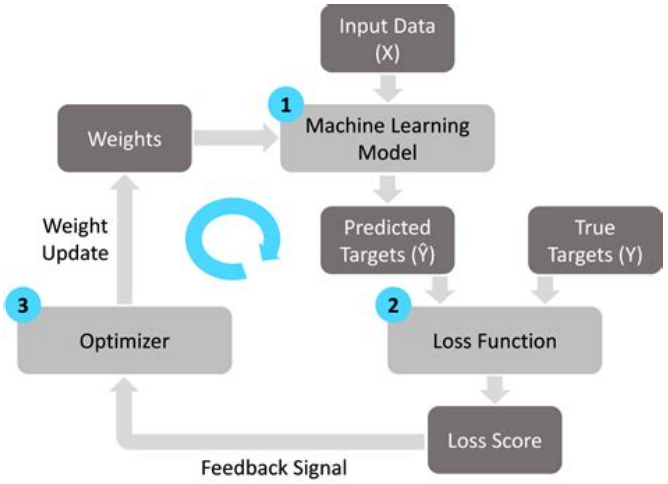
[84] https://medium.com/@boutnaru/the-artificial-intelligence-journey-feature-engineering-and-selection-3ceca7546335
[85] https://medium.com/@boutnaru/the-artificial-intelligence-journey-machine-learning-lifecycle-f74b70c4d136
[86] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[87] https://scikit-learn.org/stable/machine_learning_map.html
[88] https://medium.com/@boutnaru/the-artificial-intelligence-journey-supervised-learning-4a5aaf298275
[89] https://medium.com/@boutnaru/the-artificial-intelligence-journey-unsupervised-learning-a6a813b19f5b
[90] https://medium.com/@boutnaru/the-artificial-intelligence-journey-reinforcement-learning-87f8c0cc8099
[91] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[92] https://www.techtarget.com/searchenterpriseai/feature/How-to-build-a-machine-learning-model-in-7-steps

Model Training

After selecting a machine learning model[93] we need to train it. In this phase the model learns from the provided data in order to perform its tasks like classification and value predictions. In the current phase the input data is splitted into a "Training Set" and a "Testing Set". The training set is used for learning the patterns in the data. The testing set is used for evaluating how well the model learned[94].

Overall, we can think about the current phase as "teaching" the machine learning model to optimize performance based on the given dataset. It involves adjusting the parameters of the model (like weights and biases) as part of the mathematical functions which the algorithm is composed of[95] - as shown in the flow diagram below[96].

Lastly, the goal of the model is to minimize the loss function[97] which is used for quantifying the error of model outputs. Moreover, we need to avoid as much as possible both "Overfitting" and "Underfitting" that can hurt the model's performance[98]. By the way, there are also hyperparameters which are not "learnable" and might be needed to be tuned[99].



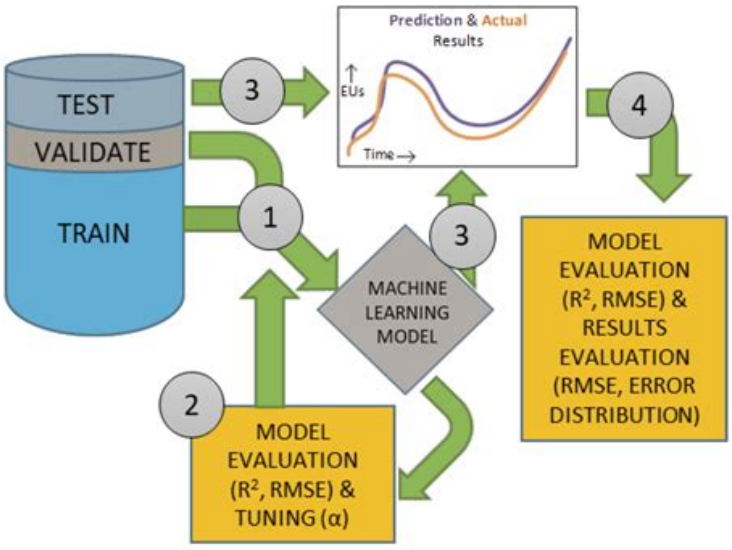
[93] https://medium.com/@boutnaru/the-artificial-intelligence-journey-model-selection-62049b109a2c
[94] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[95] https://www.ibm.com/think/topics/model-training
[96] https://agmanic.com/machine-learning-basics-how-are-models-trained/
[97] https://medium.com/@boutnaru/the-artificial-intelligence-journey-loss-function-3534db6c0d69
[98] https://medium.com/@boutnaru/artificial-intelligence-overfitting-vs-underfitting-66d83d38b1a0
[99] https://www.ibm.com/think/topics/hyperparameter-tuning

Model Evaluation and Tuning

After training a machine learning model[100] we have to evaluate it and further tune it if needed. In this phase we use different metrics in order to estimate the performance of the model like: accuracy, precision, recall, F1 and more. This is mostly done with cross-validation in which the input data is splitted for performing the performance test in a reliable way - as shown in the flow diagram below[101]. In the tuning part we can adjust parameters like and learning rate (and others) and check if they affect the performance of the model[102].

Overall, the model evaluation can be performed in one of two ways: “Online” or “Offline”. online evaluation is tested in production as part of the model monitoring. In case of offline evaluation the model is tested just after the training phase or as part of a retraining flow. It is important to understand that different categories of models (like supervised learning and unsupervised learning) can have different performance metrics[103].

Lastly, model tuning is the act of configuring the model’s implementation-specific parameters that act as control how to guide its learning. It is important to understand that model tuning is relevant for hyper-parameters that are not learned from the data itself. Examples are: tree depth (in decision tree based models) and activation functions\number of layers\etc in case of neural networks[104].



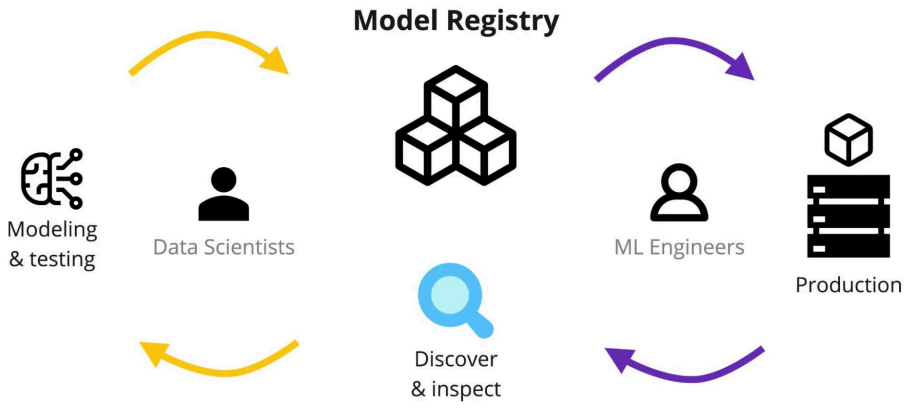
[100] https://medium.com/@boutnaru/the-artificial-intelligence-journey-model-training-9164530c7179
[101] https://medium.com/@avosehrebekah/machine-learning-model-evaluation-6ee875112e32
[102] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[103] https://www.iguazio.com/glossary/model-evaluation/
[104] https://www.iguazio.com/glossary/model-tuning/

Model Deployment

After we evaluated and tuned[105] the trained model and we are satisfied with the results it's time to deploy the model to production. It is important to understand that we might deploy the model in order to perform online evaluation before fishing the previous phase[106]. By the way, we can use a machine learning model registry (similar to container registry) which can act as a repository for model management and documentation[107] - as shown in the diagram below[108].

Overall, deploying a model is basically integrating into a real-work environment. Also, there are different deployment options available like: based on cloud platforms, API based and/or embedded systems. It is important to provide scalability that allows the model to handle a large number of requests without causing significant delays and error[109].

Lastly, there are different ways in which we can deploy a machine learning model like (but not limited to: “One-Off”, “Batch” and “Realtime”. In case of one-off the model is trained ad-hoc when needed and deployed to production. In case of a batch we can an up-to-date version of the model by sub-sampling the data each update. In case of real time we have to use an online machine learning model that we pre-trained on the most relevant data[110].



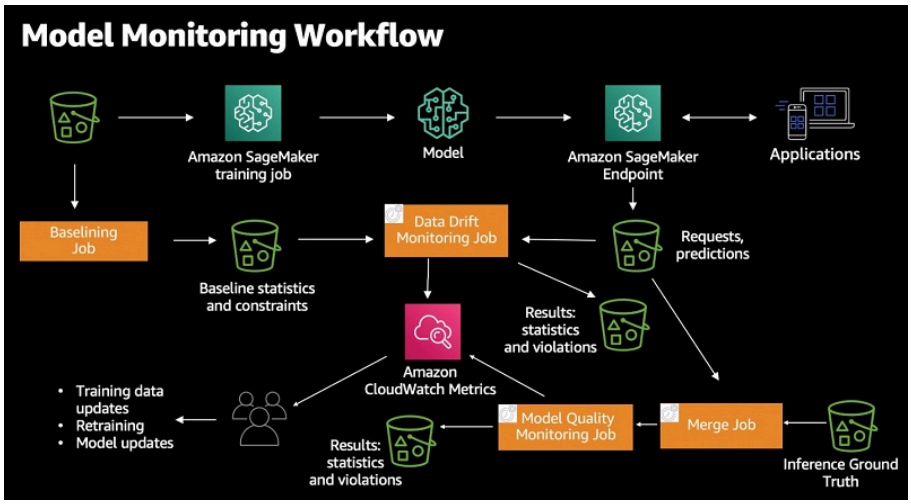
[105] https://medium.com/@boutnaru/the-artificial-intelligence-journey-model-evaluation-and-tuning-222020a2fb28
[106] https://medium.com/@boutnaru/the-artificial-intelligence-journey-machine-learning-lifecycle-f74b70c4d136
[107] https://neptune.ai/blog/ml-model-registry
[108] https://jfrog.com/learn/mlops/model-registry/
[109] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[110] https://builtin.com/machine-learning/model-deployment

Model Monitoring and Maintenance

After the machine learning model is deployed in production[111] we can get to the last phase of monitoring it and managing it[112].

Overall, monitoring means tracking the predictions in order to detect any decrease in performance/accuracy/drifts. In regards to maintenance of a machine learning model we can update it based on new data and/or new parameters/settings[113] - an example of such workflow is shown in the diagram below[114].

Lastly, among the reasons\goals for performing model monitoring we can find the following: issue detection, issue alerting, root cause analysis, performance visibility, model's behaviour analysis and more[115]. Also, by providing model maintenance we ensure it stays relevant while user's interactions and patterns might change[116].



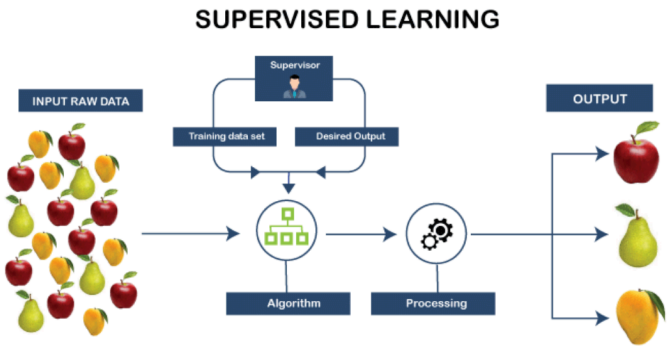
[111] https://medium.com/@boutnaru/the-artificial-intelligence-journey-model-deployment-4f2bfcf2b1ef
[112] https://medium.com/@boutnaru/the-artificial-intelligence-journey-machine-learning-lifecycle-f74b70c4d136
[113] https://www.appliedaicourse.com/blog/machine-learning-life-cycle/
[114] https://aws.amazon.com/blogs/machine-learning/monitoring-in-production-ml-models-at-large-scale-using-amazon-sagemaker-model-monitor/
[115] https://www.evidentlyai.com/ml-in-production/model-monitoring
[116] https://dialzara.com/blog/ai-model-monitoring-vs-maintenance-key-differences/

Supervised Learning

“Supervised Learning” is one of the four types of machine learning[117]. In supervised learning we use labeled data for training AI[118] models in order to identify the underlying patterns and relationships between input features and outputs. By doing so we can create a model that predicts correct outputs on new real-world data[119].

Overall, we can use supervised learning for both classification and regression[120]. Classification is when the model tries to predict the correct label of the input data. Thus, the target variable is discrete[121] - as shown in the diagram below[122]. Regression is when the goal of the model is to predict continuous numerical value based on one or more independent features. This is done by finding relationships between the input variables[123].

Lastly, deep learning can be both supervised or unsupervised[124]. Examples of supervised learning algorithms are: "Naive Bayes"[125], “Linear Regression”[126], “Support Vector Machine”[127], “Random Forest”[128] and “K-nearest Neighbor”[129].



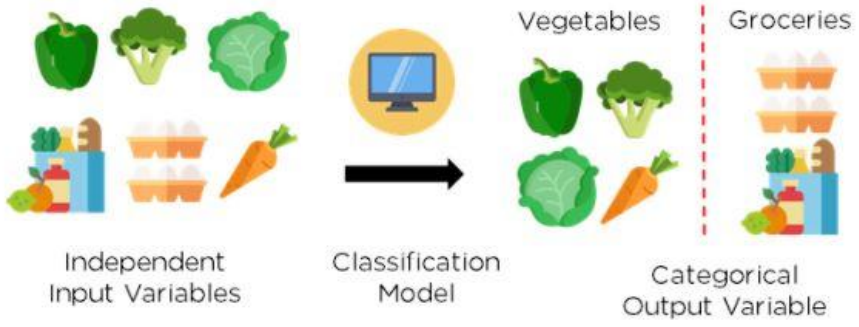
[117] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[118] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[119] https://www.ibm.com/think/topics/supervised-learning
[120] https://scikit-learn.org/stable/supervised_learning.html
[121] https://www.datacamp.com/blog/classification-machine-learning
[122] https://mungfali.com/explore/Supervised-Learning-Logo
[123] https://www.geeksforgeeks.org/regression-in-machine-learning/
[124] https://levity.ai/blog/difference-machine-learning-deep-learning
[125] https://scikit-learn.org/stable/modules/naive_bayes.html
[126] https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression
[127] https://www.geeksforgeeks.org/support-vector-machine-algorithm/
[128] https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
[129] https://www.w3schools.com/python/python_ml_knn.asp

Classification

Classification is a supervised[130] machine learning technique. It is used in order to predict the correct label of an input data[131]. Thus, we can say that it is the process of recognition and grouping of objects into preset categories ("sub-populations"). It is based on the fact the model is trained on a pre-categorized dataset. There are four different types of classification tasks: "Binary Classification", "Multi-Class Classification","Multi-Label Classification" and "Imbalanced Classification". An example of the classification flow is shown in the diagram below[132].

Overall, while regression provides continuous output, classification provides categorical output[133]. Classification models are basically the task of a mapping function (f) from input variables (x) to discrete output variables (y). Also, we can divide classification models based on the way in which the learning is performed: "Lazy Learning" vs "Eager Learning"[134].

Lastly, there are different use cases for using classification machine learning models. Examples are: email spam detection and churn prediction (binary classification), plant species\face classification (multi-class classification) and medical diagnostic and fraud detection (imbalanced classification). As with other machine learning models we can evaluate the performance of the models using different metrics such as: F1,F2, accuracy and AUC[135].



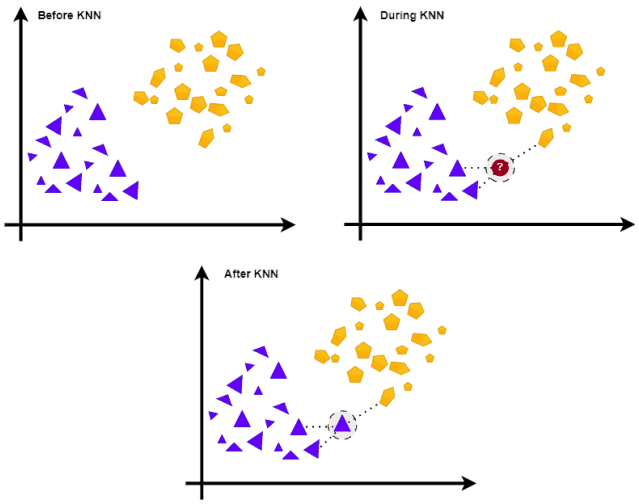
[130] https://medium.com/@boutnaru/the-artificial-intelligence-journey-supervised-learning-4a5aaf298275
[131] https://www.datacamp.com/blog/classification-machine-learning
[132] https://www.simplilearn.com/tutorials/machine-learning-tutorial/classification-in-machine-learning
[133] https://learn.microsoft.com/en-us/training/modules/understand-classification-machine-learning/
[134] https://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/
[135] https://h2o.ai/wiki/classification/

Lazy Learning

“Lazy Learning” is a machine learning[136] technique. In case of lazy learning the training data is not analyzed until the model needs to make a prediction. Thus, algorithms based on lazy learning “memorize” the training data rather than construct a general model[137].

Overall, we can use lazy learning with classification models[138]. When we have a new object we want to classify the model and search the training data for the most similar objects. By the way, lazy learners are based on similarity metrics for finding the most similar examples to the input given[139] - as shown below.

Lastly, among the machine learning algorithms which are based on lazy learning we can find: kNN (k-nearest neighbors), RBF (Radial Basis Function), LVQ (Learning Vector Quantization) and CBR (Case-Based Reasoning)[140].



[136] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[137] https://www.datacamp.com/blog/what-is-lazy-learning
[138] https://medium.com/@boutnaru/the-artificial-intelligence-journey-classification-ea539e713fd2
[139] https://www.baeldung.com/cs/lazy-vs-eager-learning
[140] https://www.sciencedirect.com/topics/computer-science/k-nearest-neighbor-classification

Eager learning

"Eager Learning" is a machine learning[141] technique. As opposed to "Lazy Learning"[142] the model parameters are adjusted during learning while minimizing the cost function - as also shown in the table below[143]. After the training phase the model can provide predictions regarding new input[144].

Overall, eager learning is an example of "offline learning" (post-training queries\requests does not affect the system and the response given). Probably the biggest advantage of eager learning is the ability to approximate the target function globally during training. Due to that, the main disadvantage is being unable to provide good local approximations in the target function[145].

Lastly, examples are: decision trees, SVM (Support Vector Machine), Naive Bayes, ANN (Artificial Neural Networks) and more[146].

Feature	Lazy Learning	Eager Learning
Training Approach	Memorizes entire training data during training.	Creates a generalized representation during training.
Prediction Process	Searches for similar instances during prediction and applies their labels.	Directly applies the learned representation for prediction.
Adaptability to New Data	Quickly adapts to new data without retraining.	Less adaptable to new data; retraining might be necessary.
Prediction Speed	It can be slower, especially with large datasets.	Faster predictions due to pre-trained model.
Offline Predictions	Requires access to training data during prediction.	Can make predictions offline or without training data.
Handling Complex Relationships	Effective at handling complex and non-linear relationships.	Better suited for well-defined patterns and relationships.
Model Representation	No fixed model representation; relies on memorized data.	Requires a fixed model representation from training.

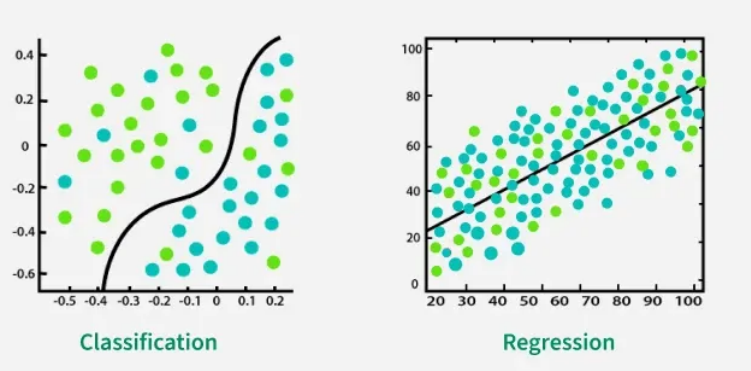
[141] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[142] https://medium.com/@boutnaru/the-artificial-intelligence-journey-lazy-learning-fc22be4f0dfc
[143] https://www.pickl.ai/blog/eager-learning-and-lazy-learning-in-machine-learning-a-comprehensive-comparison/
[144] https://www.baeldung.com/cs/lazy-vs-eager-learning
[145] https://en.wikipedia.org/wiki/Eager_learning
[146] https://devcamp.com/trails/introduction-machine-learning-data-science/campsites/machine-learning-terms/guides/lazy-vs-eager-learning

Regression

"Regression" is a supervised machine learning technique[147]. It is used for learning the relationship between variables by estimating how one variable affects others. In this case we train the model both on input and output features. Thus, we can predict a continuous outcome based on one (or more) predictor variables[148].

Overall, regression is an integral part of any forecasting model. Among the common use cases of regression in machine learning are: forecasting stock prices/sales/house prices, creating time series visualizations, predicting customers/users trends and more. There are different types of regression analysis techniques like: simple linear regression, multiple linear regression and logistic linear regression[149].

Lastly, as opposed to classification[150] which is focused on predicting a category/discrete values, regression is mainly focused on predicting a continuous value[151] - as shown below. Although regression is often continuous, there are also regression methods where the values can be discrete[152].



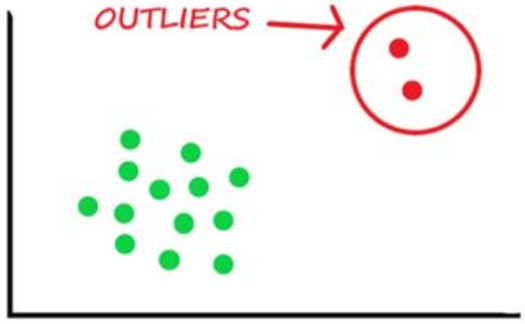
[147] https://medium.com/@boutnaru/the-artificial-intelligence-journey-supervised-learning-4a5aaf298275
[148] https://builtin.com/data-science/regression-machine-learning
[149] https://www.seldon.io/machine-learning-regression-explained
[150] https://medium.com/@boutnaru/the-artificial-intelligence-journey-classification-ea539e713fd2
[151] https://www.geeksforgeeks.org/ml-classification-vs-regression/
[152] https://towardsdatascience.com/ml-basics-part-1-regression-a-gateway-method-to-machine-learning-36d54d233907/

Outlier Detection

The goal of “Outlier Detection” is to identify data points which significantly differ from all other data points - as shown in the diagram below[153]. Those outliers can represent: errors and unusual\rare events which can lead to important discoveries in various fields. There are different types of outliers: “Global\Point Outliers” (single data point which is way off the others), “Collective Outliers” (group of data points together stand out from all others) and “Contextual Outliers” (data points that might not usually be outliers, however they are in a some cases)[154].

Overall, there are different approaches for detecting outliers such as (but not limited to): statistical models (like Grubb's test, IQR (Interquartile Range), MAD (Median Absolute Deviation) and Z-Score), proximity based models (like clustering algorithms, nearest neighbor methods and isolation forest), high dimensional outlier detection (like HiCS, feature bagging and subspace outlier detection) and machine learning models (like neural networks, random forest, SVM and logistic regression)[155].

Lastly, outlier detection can be leveraged for many use-cases such as (but not limited to): fraud detection, medical diagnostics, industrial process control and information security[156]. Also, outlier detection is sometimes referenced as anomaly detection although there are some differences between the two[157].



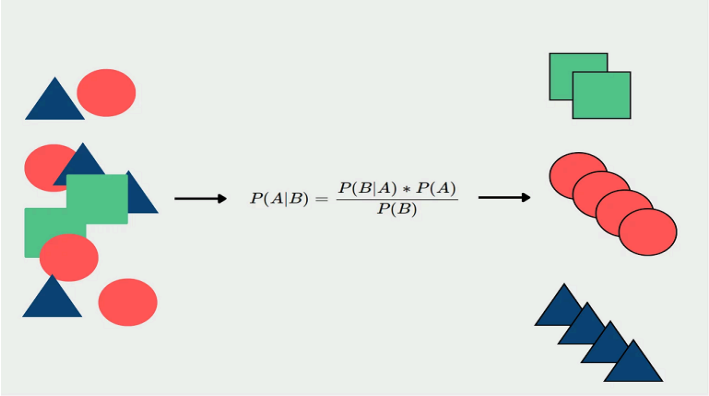
[153] https://vitalflux.com/outlier-detection-techniques-in-python/
[154] https://www.anodot.com/blog/quick-guide-different-types-outliers/
[155] https://www.ever.ai/blog/outlier-detection-algorithm-an-introduction/
[156] https://www.numberanalytics.com/blog/practical-applications-of-outlier-detection-in-real-world-data-sets
[157] https://www.geeksforgeeks.org/machine-learning/difference-between-outlier-and-anomaly/

Naive Bayes

"Naive Bayes" is a supervised machine learning algorithm[158] used for classification tasks[159]. It is based on the "Bayes' Theorem" for determining conditional probability (the likelihood of an outcome occurring based on a previous outcome in similar circumstances) - as shown in the diagram below[160]. By the way, it is named after "Thomas Bayes, a mathematician from the 18th-century[161].

Overall, the naive bayes algorithm is based on the following assumptions: features are independent, continuous features are normally distributed, discrete features have multinomial distributions, features are equally important and there is no missing data. It is important to know that assumptions are not generally correct in real-world situations. However, the algorithm still works well[162].

Lasly, we can improve the performance of "Naive Bayes" using the following methods: feature engineering, ensemble methods, parameter tuning, dealing with unbalanced data and more[163]. Also, they are different variations for classifiers such as: "Gaussian Naive Bayes", "Multinomial Naive Bayes", "Complement Naive Bayes", "Bernoulli Naive Bayes" and "Categorical Naive Bayes"[164].



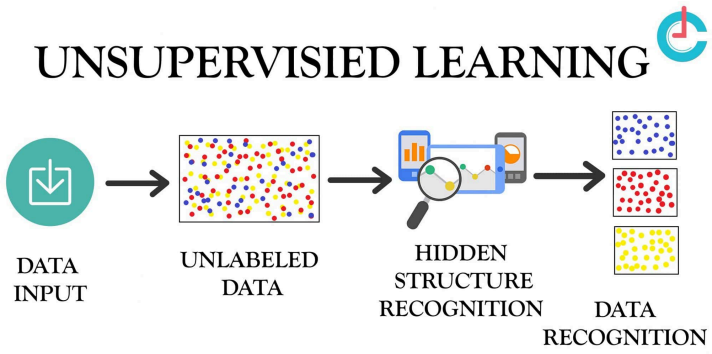
[158] https://medium.com/@boutnaru/the-artificial-intelligence-journey-supervised-learning-4a5aaf298275
[159] https://medium.com/@boutnaru/the-artificial-intelligence-journey-classification-ea539e713fd2
[160] https://databasecamp.de/ki/naive-bayes
[161] https://www.investopedia.com/terms/b/bayes-theorem.asp
[162] https://www.geeksforgeeks.org/naive-bayes-classifiers/
[163] https://databasecamp.de/en/ml/naive-bayes-algorithm
[164] https://scikit-learn.org/stable/modules/naive_bayes.html

Unsupervised Learning

"Unsupervised Learning" is one of the four types of machine learning[165]. Unsupervised learning allows learning from data without any human supervision. They are given as input unlabeled data in order to discover patterns and insights - as shown below[166]. AI models based on unsupervised learning are used in different use cases such as: genetic research, NLP (Natural Language Processing), fraud detection, recommendation engines, anomaly detection and customer segmentation[167].

Overall, there are three types of algorithms usually used in case of unsupervised learning: clustering (groups unlabeled data into clusters based on their similarities), dimensionality reduction (reducing the number of features in a dataset while preserving as much information as possible) and association rule learning[168] - more on those in future writeups.

Lastly, we have different examples of unsupervised learning algorithms. Examples of clustering algorithms are: "K-means", "Fuzzy K-means" and GMMs (Gaussian Mixture Models). In the field of association rules we can use algorithms like: "Apriori" and "FP-Growth"[169]. PCA (Principal Component Analysis) is an example of dimensionality reduction algorithm[170].



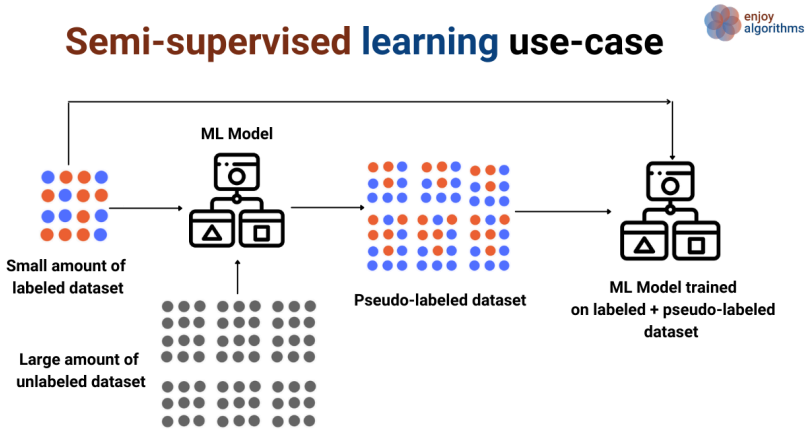
[165] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[166] https://mungfali.com/explore/Unsupervised-Learning-Block-Diagram
[167] https://cloud.google.com/discover/what-is-unsupervised-learning
[168] https://www.geeksforgeeks.org/unsupervised-learning/
[169] https://www.altexsoft.com/blog/unsupervised-machine-learning/
[170] https://medium.com/@boutnaru/the-artificial-intelligence-journey-pca-principal-component-analysis-80b4d12b2cd2

Semi-Supervised Learning

"Semi-Supervised Learning" is one of the four types of machine learning[171]. In semi-supervised learning part of our training data is not labeled. Semi-supervised learning algorithms can perform well even if we have a small amount of labeled data together with a large amount of unlabeled data points[172] - as shown in the diagram below[173].

Overall, there are different pros and cons in regards to semi-supervised learning. Among the pros are: improved model performance , effectiveness for unstructured data (like video and audio) and it is less expensive due to the fact it reduces the need for manual labeling. However, there are also cons such as: sensitivity to the data quality of the labeled data, limited transparency in regards with the reasons which lead to the predictions and less suited to complex/diverse data sets[174].

Lastly, there are several approaches for semi-supervised learning: self-training, co-training, multi-view training and SSL using graph models - more on those in future writeups. There are different applications in which semi-supervised learning is leveraged like: document classification, image classification, speech recognition, face recognition, OCR (Optical Character Recognition) and handwritten text recognition[175].



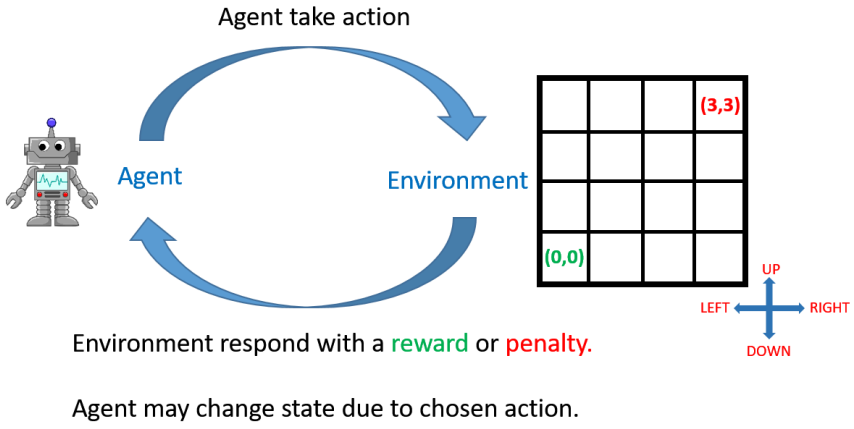
[171] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[172] https://scikit-learn.org/stable/modules/semi_supervised.html
[173] https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning/
[174] https://www.oracle.com/ba/artificial-intelligence/machine-learning/semi-supervised-learning/
[175] https://maddevs.io/blog/semi-supervised-learning-explained/

Reinforcement Learning

"Reinforcement Learning" (RL) is one of the four types of machine learning[176]. As part of reinforcement learning we mimic the "trial and error" learning experience human use for learning. Thus, actions which work towards the goal are reinforced while actions that detract from the goal are ignored. This means that reinforcement learning uses a "reward and punishment" technique as data is processed[177].

Overall, in reinforcement learning we have an "agent" that learns to make decisions by interacting with an environment (the problem space). The agent has a "state" that is the situation of the agent in the environment. Due to that each action affects the state. Based on the action of the agent a feedback is given to the agent (reward\penalty). Also, the agent has a policy which is a strategy it has to decide on actions at each state[178] - as shown below[179].

Lastly, there are two methods by which an agent can collect data for learning: "Online" and "Offline". In the first (online) one the agent collects data directly from interacting with its surrounding environment. In the second (offline), agent does not have direct access to an environment, so it can learn using logged data from the relevant environment. By the way, we can use different approaches for RL like: dynamic programming, monte carlo, and temporal difference learning[180].



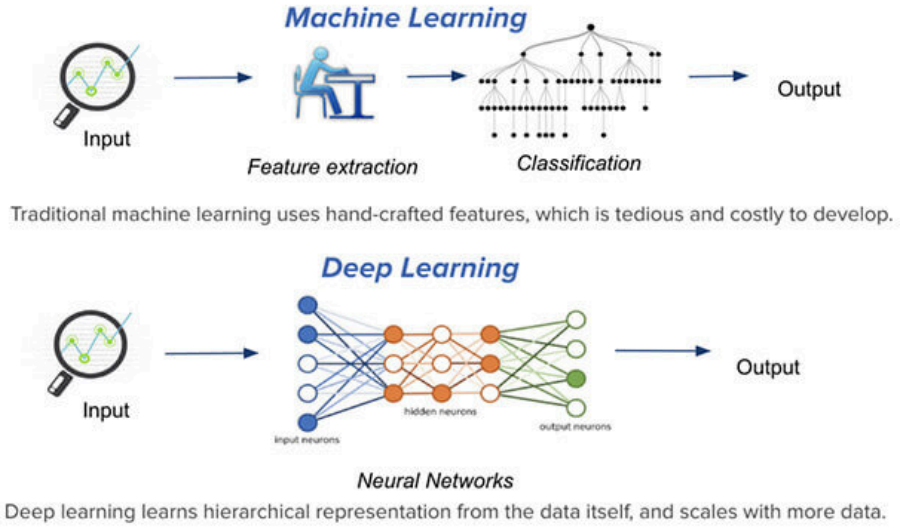
[176] https://medium.com/@boutnaru/introduction-to-machine-learning-9b488f02efb9
[177] https://aws.amazon.com/what-is/reinforcement-learning/
[178] https://saturncloud.io/glossary/reinforcement-learning-environments/
[179] https://morioh.com/a/eeaebebc8872/a-simple-reinforcement-learning-environment-from-scratch
[180] https://www.ibm.com/think/topics/reinforcement-learning

Deep Learning

As stated in the "Introduction to Artificial Intelligence"[181], "Deep Learning" is a subset of "Machine Learning" - in the diagram below we can see a description on the difference between the two[182]. Moreover, "Deep Learning" leverages neural networks in order to try and achieve learning in the way in which it tries to simulate the behavior of the human brain (learning from large amounts of data). Usually it is done using neural networks with at least three layers[183]. By the way, more on neural networks in a future writeup.

Also, "Deep Learning" models allow us to learn complex patterns which are impossible/difficult to identify using traditional machine learning models. This is due to the fact "Deep Learning" models are able to learn hierarchical representations of the data, while also performing automatic feature extraction[184].

Thus, let us think about a "Deep Learning" model that is trained for identifying cars in an image. In this case the model might identify edges->shapes->objects->car. Thus, we can summarize that "Deep Learning" enables computational models that are based on multiple layers in order to learn representations of data with levels of abstractions. This is done by using backpropagation algorithms for tweaking the internal state of the model on every layer[185]. Lastly, "Deep Learning" is used in different fields such as fraud detection, medical diagnostics, speech recognition, computer vision, NLP (natural language processing), recommendation engines and more[186]



[181] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[182] https://www.merkle.com/blog/dispelling-myths-deep-learning-vs-machine-learning
[183] https://www.ibm.com/topics/deep-learning
[184] https://aws.amazon.com/what-is/deep-learning/
[185] https://www.nature.com/articles/nature14539
[186] https://aws.amazon.com/deep-learning/

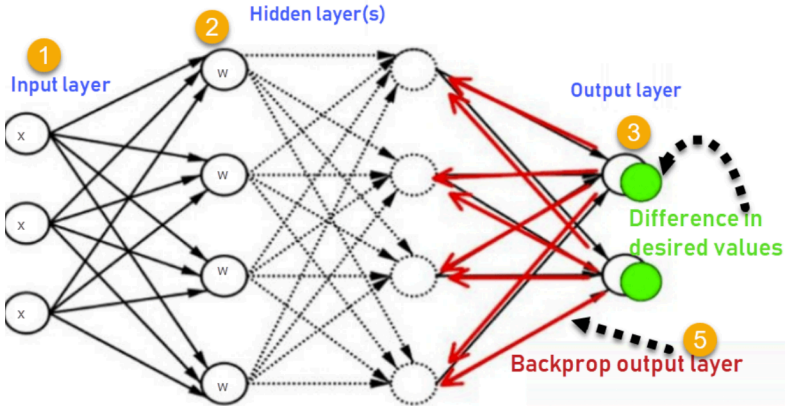
Neural Networks

A neural network is a learning system composed of interconnected nodes (aka neurons) that learn to perform tasks by processing data. Those networks are made up of layers of neurons. Each neuron in a layer [I] receives input from the neurons in layer [I-1]. Those neurons outputs a signal to the neurons in layer [I+1][187].

Moreover, the signals that are passed between neurons are weighted. Those weights are calibrated as the neural network learns more and more. An output of a neuron can be defined as y=x*W+b (x is the input, W is the weight and b is the bias). Bias is also a learnable parameter which is which stats the difference between the function's output and the intended data[188]. We can see a diagram of this description in the diagram below[189].

Overall, the learning process in a neural network is called backpropagation. By leveraging it the errors in the output (which are measured against the labels in the training data) of the neural network are propagated back through the network. Then weights of the neurons are calibrated accordingly based on the data. This process is repeated until the neural network is able to perform the task with a desired level of accuracy[190].

Lastly, we can say that the structure of a neural network depends on the number of hidden layers (determines the complexity of the model), the number of neurons in each layer (determines the amount of information the model can process) and the connections between those neurons (determines how the model learns). They are different types of neural networks like CNNs, RNNs, Deep neural networks[191]



[187] https://www.investopedia.com/terms/n/neuralnetwork.asp
[188] https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network
[189] https://www.guru99.com/backpropogation-neural-network.html
[190] https://builtin.com/machine-learning/backpropagation-neural-network
[191] https://viso.ai/deep-learning/deep-neural-network-three-popular-types/

Loss Function

Basically, loss function is a term used in math optimization/decision theory. It is sometimes called cost function, fitness function or even error function. The goal of this function is mapping an event/value(s) of a variable(s) to a real number, which represents the "cost" of the event. Thus, the loss function helps us in evaluating how well our algorithm is modeling the data set[192].

Moreover, by using a loss function we can define a learning problem as an optimization problem - as shown in the diagram below[193]. If we choose the correct loss function the learning phase can omit a good outcome[194].

Lastly, there are different loss functions which are commonly used as part of deep learning algorithms - following are a couple of examples. With regression example loss functions are: MSE (Mean square error), MAE (Mean Absolute Error) and Hubber loss. With classification examples loss functions are: binary cross entropy and categorical cross-entropy. Autoencoders can use KL divergence. Object detection can leverage and word embedding can use triplet loss. GAN can use a loss function discriminator loss or minmax GAN loss[195] .

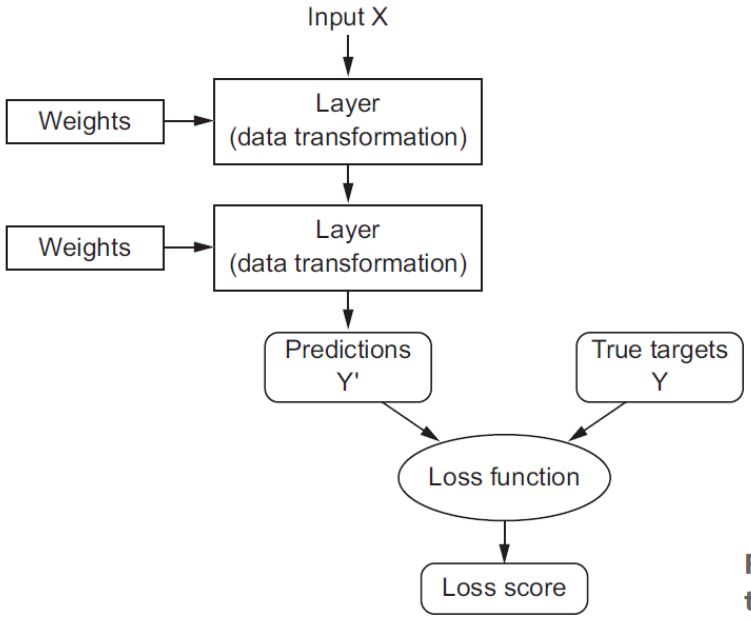


Figure 1.8 A loss function measures the quality of the network's output.

[192] https://en.wikipedia.org/wiki/Loss_function
[193] https://medium.com/howtoai/playing-with-loss-functions-in-deep-learning-26faf29c85f
[194] https://c3.ai/glossary/data-science/loss-function/
[195] https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/

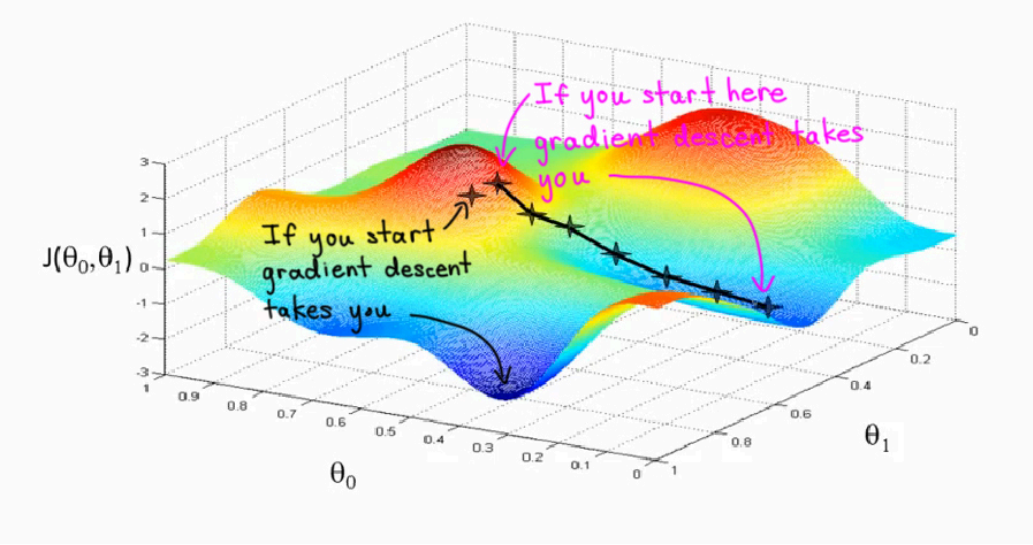
Gradient Descent (GD)

Gradient descent (GD) is an optimization algorithm that is focused on locating the local minimum/maximum of a specific function. It is done in an iterative manner which starts from a random point and then takes steps in the direction of the negative gradient of the function - as shown in the diagram below[196]. Moreover, gradient descent does not work for every function. There are two requirements for a function in order for GD to work: the function needs to be differentiable and convex[197].

Differentiable means that for every point in the function domain there is a derivative. Not all functions are like that, think about $f(x)=1/x$[198]. Convex is relevant for continuous functions, it means that for any two distinct points on the function's graph the line that connects them is above the graph between those points[199].

Overall, we can think of a gradient as a vector that points in the direction of the steepest ascent of the function. If we take steps in the opposite direction of the vector we are going to move in the direction of the nearest minimum point of the function[200].

Thus, we can use gradient descent in order to minimize the errors between the actual results and the expected results while training machine learning models. There are multiple variants of gradient descent like: Batch gradient descent, stochastic gradient descent, and mini-batch gradient descent[201] - More on them and others in future writeups. By the way, gradient descent was discovered by Augustin-Louis Cauchy in the mid 18th century[202].



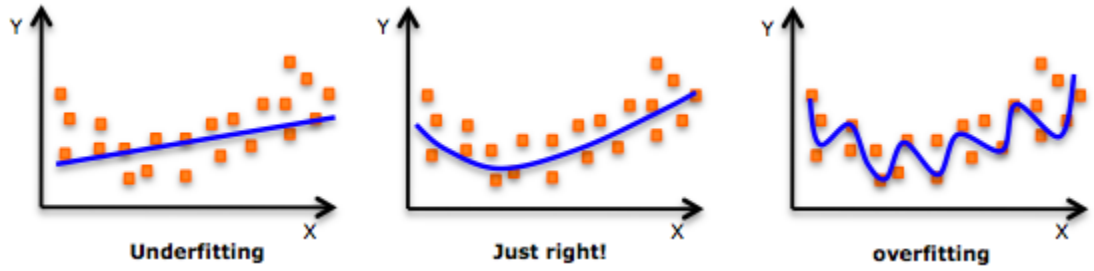
[196] https://regenerativetoday.com/machine-learning-gradient-descent-concept/
[197] https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
[198] https://math.mit.edu/~djk/calculus_beginners/chapter09/section03.html
[199] https://mathworld.wolfram.com/ConvexFunction.html
[200] https://www.uio.no/studier/emner/matnat/ifi/IN3050/v23/groups/gradient-descentascent.pdf
[201] https://www.javatpoint.com/gradient-descent-in-machine-learning
[202] https://www.deeplearning.ai/the-batch/gradient-descent-its-all-downhill/

Overfitting vs Underfitting

Overfitting is a case in which a machine learning model learns the training data too well and due to that it is unable to generalize on new data (not contained in the training data). So we can say that the statistical model performs great on the training data but it can't perform accurately against unseen data. We need to remember that generalization of a machine learning model to new data is what makes it relevant for classifying data/predicting results[203]. There are two main reasons for that: the training data is not representative of the real world and/or the model is too complex.

Underfitting is the case in which a machine learning model can't capture the relationship between the input and output variables. It generates a high error rate on real data (it can also happen with the training data). It can happen in case the size of the training dataset used is not enough, the model is too simple, the training data is not cleaned and more[204].

Lastly, as we can see both overfitting and underfitting are statistical problems we want to avoid when creating machine learning models. We can see an illustration of the different problems in the diagram below[205]. In the diagram the data points are shown as orange squares, while the model is represented as the blue line. In the next writeups we are going to talk about how to avoid overfitting and underfitting.



[203] https://www.ibm.com/topics/overfitting
[204] https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
[205] https://www.analyticsvidhya.com/blog/2015/02/avoid-over-fitting-regularization/

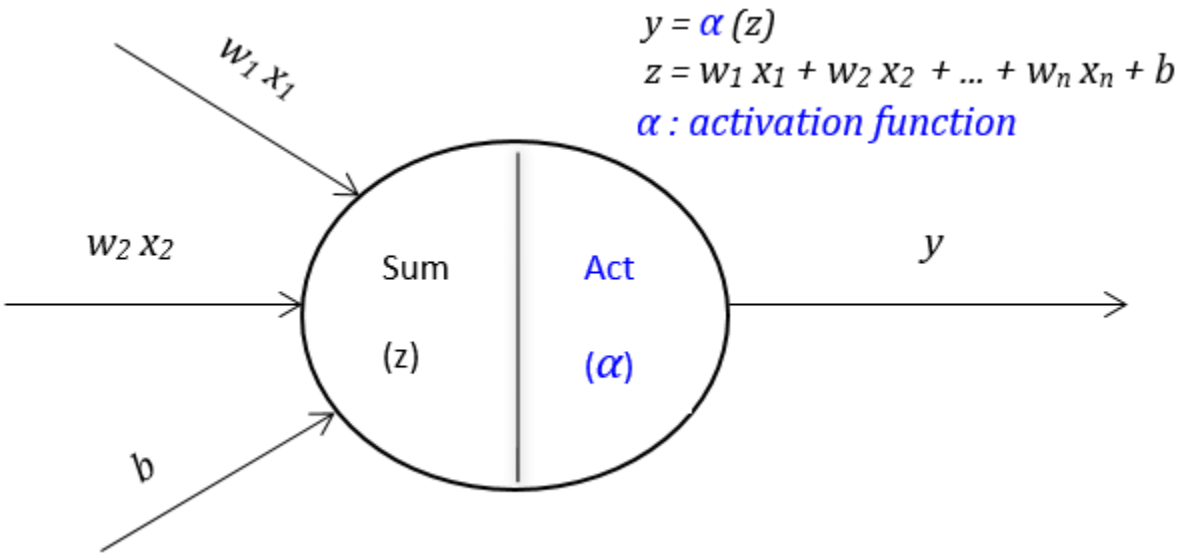
Activation Functions

An activation function is used by neural networks in order to compute the weighted sum of the input and biases - as shown in the diagram below. In some books those functions are called "Transfer Functions"[206]. We call them "Activation Functions" because they cause a specific neuron to be activated or not.

Moreover, activation functions can be linear or nonlinear. If we think about it, classical machine learning algorithms (like regression and SVM) were created based on linear models. However, not all real-life problems have a nature of non-linearity. Due to that we can get non-optimal results in case we use just linear based models[207].

Based on that understanding and the fact an activation function can be linear/non-linear we can use them to cope with the non-linearity of real-life problems. We just apply a non-linear function on the output of a specific layer before it is being propagated to the input of the next layer.

Lastly, there are different types of non-linear active functions that we can use like: Sigmoid, TanH, ReLU, Parametric ReLU and ELU .



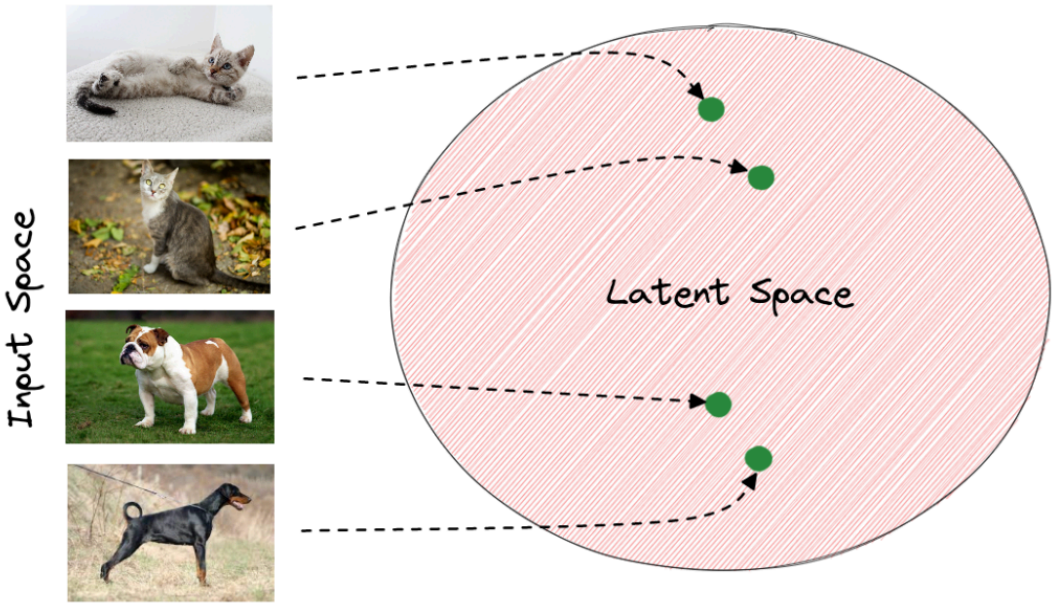
[206] https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/
[207] https://www.exxactcorp.com/blog/Deep-Learning/activation-functions-and-optimizers-for-deep-learning-models

Latent Space

A latent space provides a lower-dimensional representation designed to capture the underlying structure and relationships within that data. It is used in different areas such as neuroscience, data analysis and deep learning. By the way, it is also known as embedding space or as a latent feature space. We can say that by using a latent space the input data is organized and mapped in a way where similar things are positioned closer together - as shown in the diagram below[208].

Thus, using a latent space we can perform dimension reduction and identify similarity and relationship between different inputs. We can construct a latent space using different linear/nonlinear dimensionality reduction techniques, such as Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and autoencoders[209].

Lastly, latent space is used in clustering[210], dimensionality reduction[211], GAN[212], feature extraction[213] , image recognition[214], anomaly detection[215] and more. .



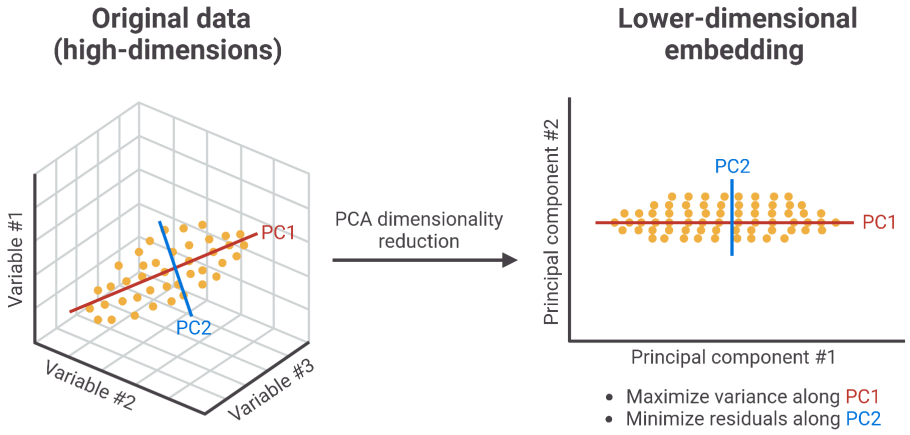
[208] https://www.baeldung.com/cs/dl-latent-space
[209] https://saturncloud.io/glossary/latent-space/
[210] https://ojs.aaai.org/index.php/AAAI/article/view/4385
[211] https://cdn.aaai.org/AAAI/2008/AAAI08-108.pdf
[212] https://proceedings.mlr.press/v119/voynov20a.html
[213] https://www.mdpi.com/2076-3425/12/10/1348
[214] https://ieeexplore.ieee.org/abstract/document/8545506/
[215] https://link.springer.com/article/10.1007/s10994-022-06153-4

PCA (Principal Component Analysis)

The goal of PCA (Principal Component Analysis) is to perform dimensionality reduction - as shown in the diagram below[216]. By using PCA we can simplify a large data set into a smaller set while still maintaining significant patterns and trends[217].

Thus, PCA is usually used as a data preprocessing phase with machine learning algorithms. By doing so we reduce the model complexity, because the addition of each new feature negatively impacts model performance ("curse of dimensionality)". The idea is to project a high-dimensional dataset into a smaller feature space[218].

Lastly, as opposed to LDA (Linear Discriminant Analysis) we can use PCA both for supervised learning tasks and unsupervised learning[219]. PCA is based on different concepts from linear algebra such as: eigenvalues, eigenvectors and covariance matrix.



[216] https://www.biorender.com/template/principal-component-analysis-pca-transformation
[217] https://builtin.com/data-science/step-step-explanation-principal-component-analysis
[218] https://www.ibm.com/topics/principal-component-analysis
[219] https://www.ibm.com/topics/linear-discriminant-analysis

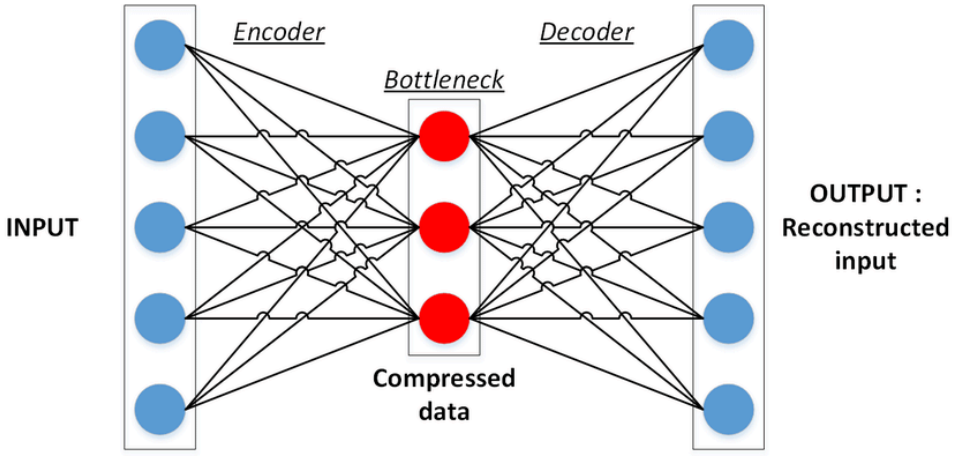
Autoencoders

In general, autoencoders are a specific type of a neural network[220] which can be used for dimension reduction by learning an efficient coding of unlabeled data. We can say it is trained to compress the input data (to a lower dimension) and then reconstructing the data in order to match it to the original input as closely are possible[221].

Thus, the architecture of an autoencoder is composed of three layers: encoder, code (aka the bottleneck) and decoder - as shown in the diagram below[222]. The encoder layer compresses the input data into a latent-space representation. The code layer represents the compressed input fed to the decoder layer. The decoder layer reconstructs the input data from the latent space representation[223].

Moreover, the goal is to train autoencoders to minimize the reconstruction error. This is done by using a loss/fitness/error function[224] an example is MSE (mean squared error) between the input data and the reconstructed data. The weights of the neural network are adjusted during the training phase to minimize the loss function[225] mostly using backpropagation.

Lastly, there are different types of autoencoders such as: sparse autoencoders, contractive autoencoders, undercomplete autoencoders, denoising autoencoders and variational autoencoder.



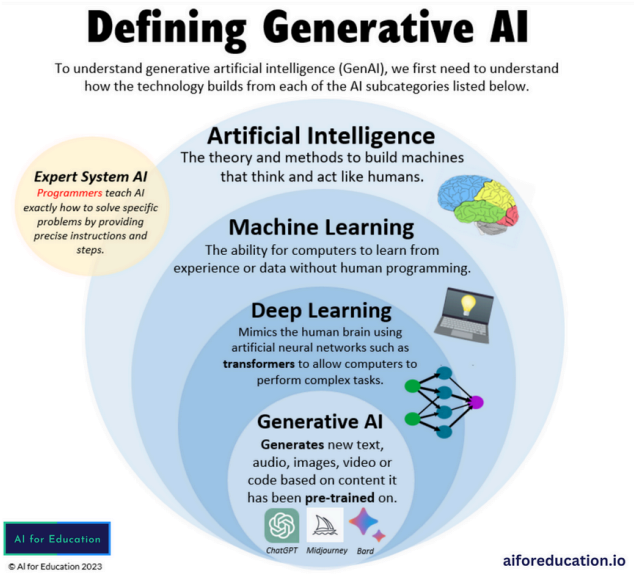
[220] https://medium.com/@boutnaru/introduction-to-neural-networks-f65bf17afd2c
[221] https://deepai.org/machine-learning-glossary-and-terms/autoencoder
[222] https://stackabuse.com/autoencoders-for-image-reconstruction-in-python-and-keras/
[223] https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-are-autoencoders-in-deep-learning
[224] https://medium.com/@boutnaru/the-artificial-intelligence-journey-loss-function-3534db6c0d69
[225] https://www.v7labs.com/blog/autoencoders-guide

GenAI (Generative Artificial Intelligence)

GenAI (Generative Artificial Intelligence) in the field of artificial intelligence[226]. It uses "Deep Learning"[227] models which receive as input raw data (think about all wikipedia as an example and much more) and "learn" how to generate statistically probable outputs when prompted[228] - as also explained in the diagram below[229]. Well known examples of GenAI applications are: "ChatGPT", "DALL-E" and "Google Gemini" which was previously called "Bard"[230].

Overall, there are multiple types of generative AI models like diffusion models. Examples of those are VAEs (Variational Autoencoders) and GANs (Generative Adversarial Networks). Also, there are different architectures for GenAI models, probably the most used one is a "Transformer Network"[231].

Lastly, generative AI is able to create new content like text, images, video and speech based on the patterns of data learned by the models powering it. Another type of models that can be used for generative AI: LLMs (Large Language Models), SLMs (Small Language Models), RAG (Retrieval-Augmented Generation) - they can also be combined with those mentioned above. Also, by using "AI Agents" we can autonomously perform different tasks by using generative AI[232].



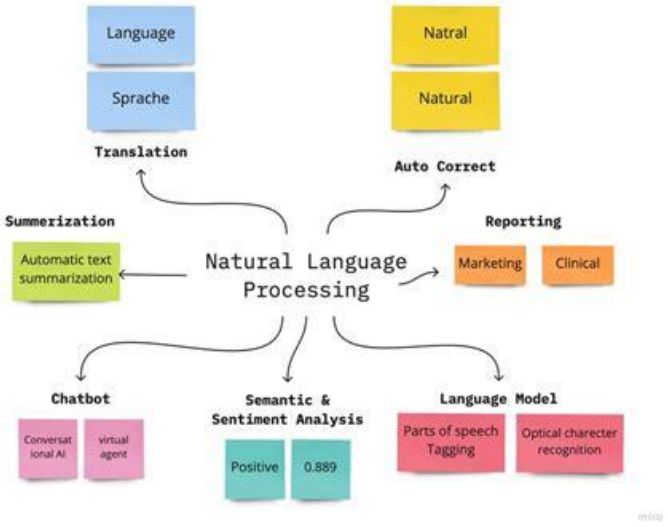
[226] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[227] https://medium.com/@boutnaru/introduction-to-deep-learning-433680bfafef
[228] https://research.ibm.com/blog/what-is-generative-AI
[229] https://www.aiforeducation.io/ai-resources/generative-ai-explainer
[230] https://www.coursera.org/articles/generative-ai-applications
[231] https://www.nvidia.com/en-eu/glossary/generative-ai/
[232] https://generativeai.net/

NLP (Natural Language Processing)

NLP (Natural Language Processing) is a realm of AI[233] which is used for helping computer systems analyze/understand/respond to humans using speech and written text. NLP is a part of "computational linguistics" which merges between AI, computer science and linguistics. Its goal is studying the computational aspects of human language[234].

Overall, there are different approaches for NLP. "Rules Based NLP" the earlier form, we can think about that as if-then decision trees based on programming rules an example is the original version of "Moviefone". "Statistical NLP" which maps language elements (words and/or grammatical rules) to a vector representation. Then that language can be modeled by using statistical methods (like regression or Markov models). "Deep Learning NLP" leverages volumes of unstructured data (voice and text) in order to be more accurate. Examples of such models are: "seq2seq" (Sequence-to-Sequence), "Transformer Models" and "Autoregressive Models"[235].

Lastly, we could use NLP for different use case such as machine translation, spam detection, sentiment analysis, text simplifications and more - as also described in the diagram below[236]. Real world examples for those are: "Google Translate", "Grammarly", "Alexa" and "OK Google"[237].



[233] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[234] https://www.elastic.co/what-is/natural-language-processing
[235] https://www.ibm.com/topics/natural-language-processing
[236] https://www.researchgate.net/figure/Natural-Language-Processing_fig1_364842359
[237] https://k21academy.com/datascience-blog/machine-learning/natural-language-processing/

LLM (Large Language Model)

LLM (Large Language Model) is an AI[238] application which can be leveraged (among other things) to recognize and generate text. Those models are trained on huge data sets (so they have enough examples to recognize\interpret human language) using a type of neural network called a "transformer model"[239].

Overall, the goal of an LLM is to understand the relationships between characters, words and sentences. We can use LLMs to generate/translate text and perform different NLP[240] tasks. Thus, we can leverage LLMs from many use cases like: chatbots, document summarization, AI-powered contact center and more[241]. We can compare between various LLMs based on attributes like size, code license, use and more - as shown in table below[242].

Lastly, from all the above we can say the LLMs are a subcategory of AI. They are focused on understanding, predicting, and generating human-like text. Because of that, LLMs are a fundamental piece of GenAI[243]. Alos, large language models come with different challenges such as: privacy concerns, technical complexity, business dependency and continuity, ethical concerns regarding bias and fairness. misinterpretation of data and more[244].

Comparing different LLMS

Model	Size	Use	Training code available	Inference code available	Finetuning code available	Code license	Weights license	Instruction-tuned / foundation model	Backbone
Bloomz	176B	Restricted applications	❌	✅	❌	Responsible AI (OpenRail)	Responsible AI (OpenRail)	Instruction-tuned	Bloom
Chat GPT (gpt-3.5-turbo)	175B	Paid API	❌	❌	❌	Public Web API	Public Web API	Instruction-tuned	GPT3
Dolly-V2	1B	Commercial	❌	❌	❌	Apache License 2.0	Apache License 2.0	Instruction-tuned	Pythia
Lit-LLaMA	7B	Non-commercial research	✅	✅	✅	Apache License 2.0	Non-commercial research	Foundation model	–
Lit-LLaMA + Alpaca	7B	Non-commercial research	✅	✅	✅	Apache License 2.0	Non-commercial research	Instruction-tuned	LLaMA

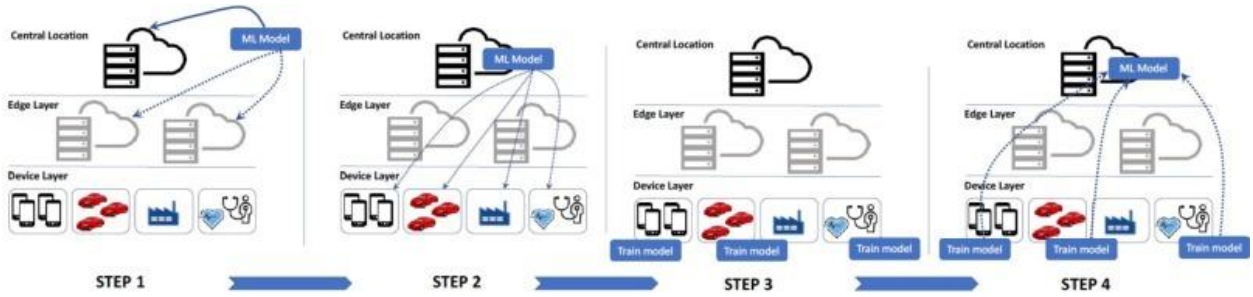
[238] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[239] https://www.cloudflare.com/learning/ai/what-is-large-language-model/
[240] https://medium.com/@boutnaru/the-artificial-intelligence-journey-nlp-natural-language-processing-41ae7d1d4428
[241] https://cloud.google.com/ai/llms?hl=en
[242] https://muhtasham.github.io/blog/posts/llm-bootcamp/
[243] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[244] https://www.sap.com/resources/what-is-large-language-model

Federated Learning

The goal of "Federated Learning" (FL) is to train AI models without anyone seeing\touching our data. This term was keyed by Google in 2016 (when the use and misuse of personal data was gaining global attention). By using federated learning multiple entities share their data remotely for training a single deep learning[245].

Overall, federated learning (sometimes called collaborative learning) is a decentralized approach for training AI models. By using federated learning we can achieve: privacy, data security and access to heterogeneous data[246]. We can summarize the FL flow in the following four phases. First, the central server selects a base AI model. Second, the model is sent to the involved edge entities (clients). Third, each edge entity trains a model using its local data (infers updates or gradients). Fourth, the updates are sent to the aggregator server, which updates the global model (by the way the flow can be reiterated as needed) - as shown in the diagram below[247].

Lastly, there are different approaches for FL: centralized federated learning, decentralized federated learning and heterogeneous federated learning[248]. The aggregator approach is essential to make the process of collaborating the obtained results efficient. There are different algorithms that can be used by an aggregator like: FedSGD (Federated Stochastic Gradient Descent), FedAvg (Federated Averaging) and FedDyn (Federated Learning with Dynamic Regularization)[249].



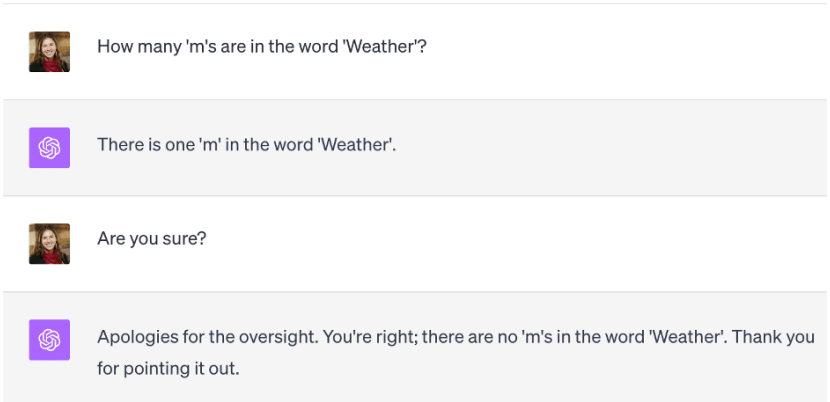
[245] https://research.ibm.com/blog/what-is-federated-learning
[246] https://www.v7labs.com/blog/federated-learning-guide
[247] https://www.sciencedirect.com/topics/computer-science/federated-learning
[248] https://openreview.net/pdf?id=Cnn60wwTe1
[249] https://arxiv.org/html/2405.16682v1

LLM Hallucinations

"LLM Hallucinations" are cases in which an LLM[250] emits outputs which seem coherent and grammatically correct but factually incorrect or nonsensical. Thus, "Hallucinations" are basically generations of false\misleading information by an LLM. It can be created due to limitations in the model, training data, biases and more. An example of such a case is shown in the screenshot below[251].

Overall, we can classify LLM hallucinations into different types like: "Intrusive Hallucinations" (introducing irrelevant\incorrect information into an otherwise coherent\relevant response), "Fabricative Hallucinations" (creating entirely fictional narratives\events\facts with no basis in reality), "Misinformative Hallucinations" (presenting incorrect data\assertions as factual) and "Contextual Hallucinations" (generating contextually inappropriate responses\disconnected from the preceding dialogue\query). We could use different techniques for identifying hallucinations such as: cross reference facts, integration with user feedback and continually learning/updating the model[252].

Lastly, in order to prevent hallucinations we can leverage different techniques. We can limit the possible outcomes a model can predicate. Also, we should train our AI model using only specific sources relevant to the taks we want it to perform. Creating a template for the AI model to follow and telling it what we want (and what we don't) can also help[253].



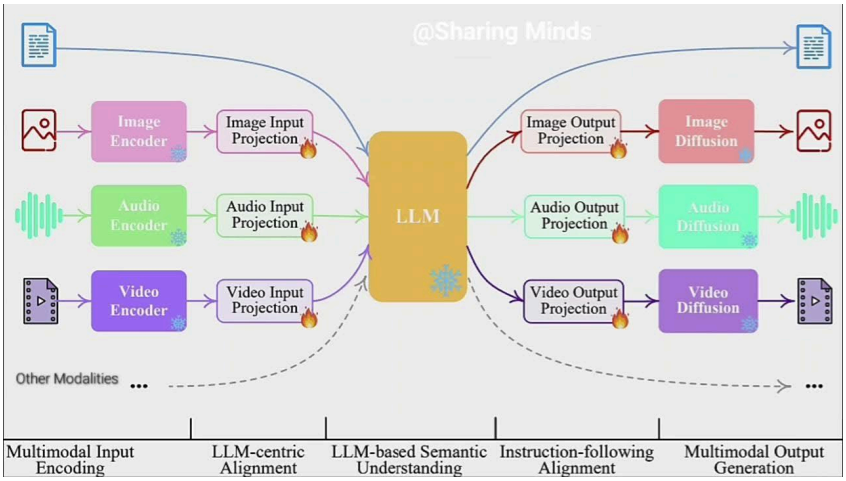
[250] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[251] https://www.iguazio.com/glossary/llm-hallucination/
[252] https://www.gigaspaces.com/data-terms/llm-hallucinations
[253] https://cloud.google.com/discover/what-are-ai-hallucinations

LMM (Large Multimodal Model)

LMM (Large Multimodal Model) is an AI[254] application which can be leveraged to understand/create content across multiple data types like: text, images, audio and video. Those models are also sometimes called MLLM (Multimodal Large Language Models). They are trained on vast amounts of data from different fields for learning the patterns\relationships\structures within each data type. By using LMMs we don't need a separate model for each data type[255] - as shown in the diagram below[256] .

Overall, we can think about LLMs as an integration of extra modalities into LLMs[257]. By the way, a system can be called multi-model if its inputs are multimodal or its outputs are multimodal or if both its inputs\outputs are multimodal. It does need to include a language model component in order to be called LMM[258].

Lastly, examples of LMM models are: "GPT 4o", "Perplexity Sonar 3.1", "Claude 3 Opus", "Copilot Pro" and "Gemini Advanced". There are also open source LLMs like: "Janus-Pro" by DeepSeek, CLIP (Contrastive Language–Image Pretraining) by OpenAI and "Flamingo" by DeepMind[259].



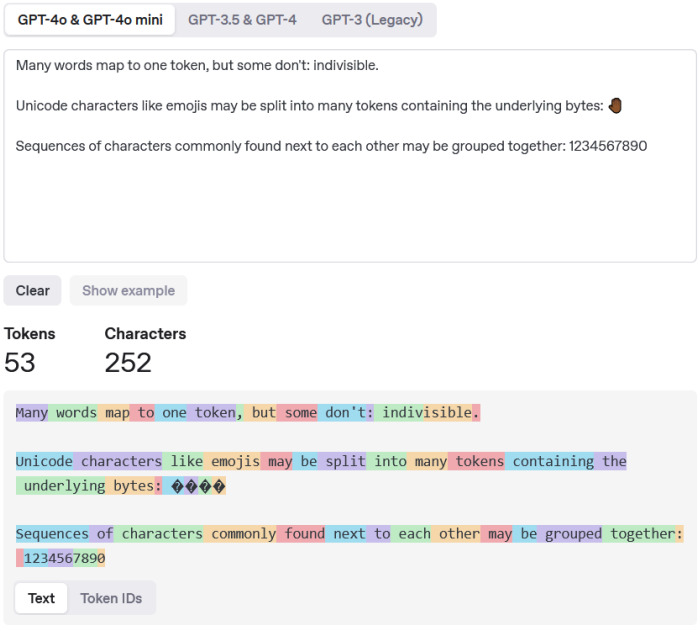
[254] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[255] https://www.theainavigator.com/blog/what-is-an-lmm-large-multimodal-model
[256] https://www.youtube.com/watch?v=Z0lj53pgLDI
[257] https://www.holisticai.com/glossary/lmm
[258] https://pmc.ncbi.nlm.nih.gov/articles/PMC10915764/
[259] https://research.aimultiple.com/large-multimodal-models/

Tokenization

LMMs (Large Multimodal Models) are trained on huge corpuses of text[260]. Text is composed of words that are cataloged as a token (the basic unit\"atom" of LLMs). In case of long words their tokenization process can include splitting them to smaller strings[261]. Thus, we can define "tokenization" as the process of translating strings and converting them into sequences of tokens and vice versa[262].

Overall, AI models (such as LLMs) are trained based on tokens. The tokenization flow determines the model's vocabulary. Using it the model can process input and predict the output[263]. There are different tokenization types like: word tokenization, character tokenization and subword tokenization[264].

Lastly, for a reference implementation of a tokenizer we can check out tiktoken from OpenAI[265]. We can also use it online from OpenAI to understand how a piece of text might be tokenized by a language model - as shown in the screenshot below[266]. Those tokens are of course assigned tokens IDs.



[260] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[261] https://www.vellum.ai/llm-parameters/top-k
[262] https://christophergs.com/blog/understanding-llm-tokenization
[263] https://airbyte.com/data-engineering-resources/llm-tokenization
[264] https://www.datacamp.com/blog/what-is-tokenization
[265] https://github.com/openai/tiktoken
[266] https://platform.openai.com/tokenizer

Base LLM (Base Large Language Model)

A “Base LLM” (Base Large Language Model) is an LLM[267] in the field of “Generative AI”[268]. They represent the fundamental model which had been obtained due the training on large volumes of text from the Internet. Examples of such models are: “GPT-3”, “GPT-4” and “BERT”[269].

Overall, “Base LLMs” are great for understanding and predicting language patterns but they don’t work well in following instructions provided by prompts (as opposed to “Instruction-tuned LLMs” - more on those in future writeups). Due to those reasons “Base LLMs” are mostly relevant for applications in the following areas: general knowledge responses, translations, summarization and more[270]. By the way, “Base LLMs” are sometimes called pre-trained LLMs[271].

Lastly, we can summarize that “Base LLMs” are designed to predict the next word based on their training data[272] - as shown below[273]. For better understanding let us think about the next example. If we prompt “What is the capital of France?” we could get a response like “What is France's largest city?” or “What is France's population?” from the “Base LLM”[274].

Eric: Hi Tom! How are
Eric: Hi Tom! How are you?
Eric: Hi Tom! How are you? Tom:
Eric: Hi Tom! How are you? Tom: Sorry,
Eric: Hi Tom! How are you? Tom: Sorry, who
Eric: Hi Tom! How are you? Tom: Sorry, who are
Eric: Hi Tom! How are you? Tom: Sorry, who are you
Eric: Hi Tom! How are you? Tom: Sorry, who are you again?

Several iterations of an LLM predicting the next word of a sentence

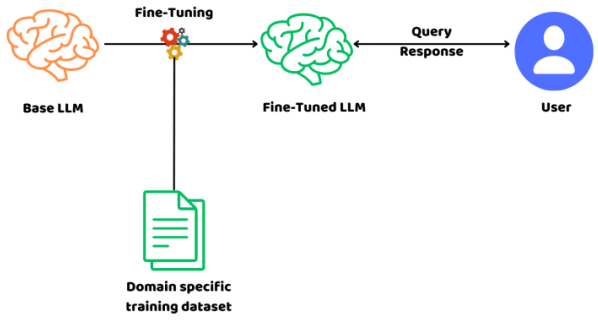
[267] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[268] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[269] https://oliviermills.com/articles/understanding-ai-models-base-language-learning-models-vs-instruction-tuned-language-learning-models
[270] https://toloka.ai/blog/base-llm-vs-instruction-tuned-llm/
[271] https://whylabs.ai/learning-center/introduction-to-llms/training-and-fine-tuning-large-language-models
[272] https://roadmap.sh/prompt-engineering/basic-llm/llm-types
[273] https://www.gravity-testing.com/blog/mastering-llm-agent-the-testers-essential-guide/
[274] https://learnius.com/llms/2+LLMs+and+Transformers/base+LLM

Instruction Tuned LLM

"Instruction Tuned LLM" is a LLM[275] built on top of "Base LLMs"[276]. However, instead of just trying re predictive (autocomplete) text they try to follow the given instructions using the data that they have been trained on. Thus, we can start with a "Base LLM" and expend the training using a large dataset covering sample "Instructions" and how the model should react as a result of those instructions. Then we fine-tune the process - as shown in the diagram below[277]. The fine-tuned model can leverage "RLHF" ("Reinforcement Learning with Human Feedback"). By using RLHF the model learns from human feedback and improves its performance[278].

Overall, instruction tuned LLMs provide different benefits like: improved instruction following, better handling complex requests, task specialization, responsive to tune and style, better understanding of user intent and more. Examples of applications are: business and productivity tools, customer support and virtual assistance and creative writing and content ideas[279].

Lastly, there are different open source human created instruction datasets like: Flan, OpenAssistant and Dolly. There are also LLM generated datasets (due to the prohibitive amount of cost and labor required to manually generate instructions) such as: Self-Instruct, Evol-Instruct and OpenOrca[280].



Fine-Tuning Process

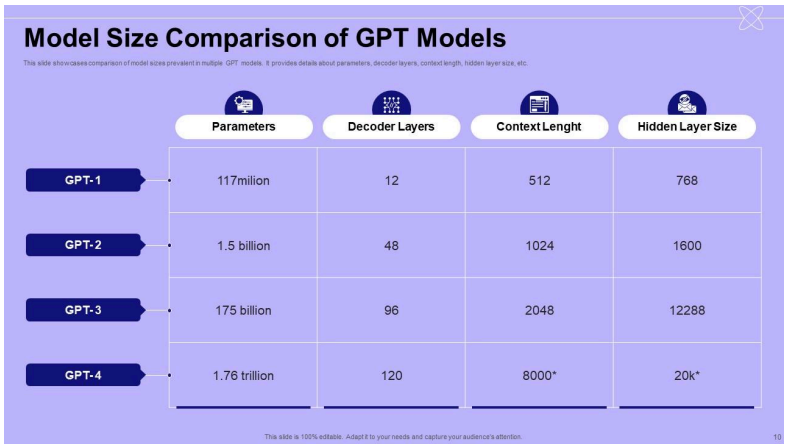
[275] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[276] https://medium.com/@boutnaru/the-artificial-intelligence-journey-base-llm-base-large-language-model-726423106b14
[277] https://crucialbits.com/blog/llm-customizations-prompt-engineering-rag-fine-tuning/
[278] https://roadmap.sh/prompt-engineering/basic-llm/llm-types
[279] https://toloka.ai/blog/base-llm-vs-instruction-tuned-llm/
[280] https://www.ibm.com/think/topics/instruction-tuning

GPT (Generative Pre-Trained Transformer)

GPT (Generative Pre-Trained Transformer) is a specific type of a LLM[281]. It is used in the field of NLP aka "Natural Language Processing"[282]. It is based on the transformer deep learning architecture and pre-trained on large amounts of text which allows it to generate new content - as we can learn from its name[283].

Overall, GPT was first introduced in 2018 by OpenAI. From then OpenAI have released different foundation models (trained on broad data that can be adapted to a wide range of tasks) from the "GPT-n" series such as: GPT-1 (117 million parameters), GPT-2 (1.5 billion parameters), GPT-3 (175 billion parameters), GPT-4 (estimated 1.7 trillion parameters), GPT-4o and GPT-4.5. A more detailed comparison between models is shown below[284]. Based on GPT (and instruction based training) OpenAI launched in 2022 ChatGPT[285].

Lastly, it is important to understand that as of today the term "GPT" is described also for models which are developed not only by OpenAI. Examples are the GPT models created by EleutherAI: GPT-Neo (125 million\1.3 billion\2.7 billion parameters), GPT-J (6 billion parameters), GPT-NeoX (20 billion parameters) and more[286].



[281] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[282] https://medium.com/@boutnaru/the-artificial-intelligence-journey-nlp-natural-language-processing-41ae7d1d4428
[283] https://huggingface.co/docs/transformers/en/model_doc/openai-gpt
[284] https://www.slideteam.net/introduction-to-gpt-4-powerpoint-ppt-template-bundles-chatgpt-mm.html
[285] https://en.wikipedia.org/wiki/Generative_pre-trained_transformer
[286] https://en.wikipedia.org/wiki/EleutherAI#GPT_models

SLM (Small Language Model)

SLM (Small Language Model) is an AI[287] model which can be used (among others) to process/understand/generate natural language content. Their main goal is speed and realtime performance. They are more compact than LLMs[288]. Also, they can run on smartphones, tablets and even smartwatches[289].

Probably the most significant difference between LLMs and SLMs in the size of the model. For example "GPT-4" which is a LLM has about 1.76 trillion parameters while an SLM like "Mistral 7B" can have only 7 billion model parameters. Moreover, an SLM is usually trained on data from a specific domain as opposed to LLMs - a table comparing "LLM vs LSM" is shown below[290]. Because the model is smaller SLMs can run on local computers and generate data within acceptable time[291].

Lastly, a great example for using SLM usage is "Microsoft Recall"[292]. By the way, examples of popular SLMs are: "DistilBERT", "Gemma" and "Phi"[293].

LLM vs SLM

Aspect	LLM (Large Language Models)	SLM (Small Language Models)
Advantages 	• Deep language understanding • Versatility • Contextual relevance	• Efficient resource utilization • Suitable for smaller datasets • Faster training and inference
Drawbacks 	• Computationally intensive • Data requirements • Potential for bias	• Limited language understanding • Contextual limitations • Task-specific training

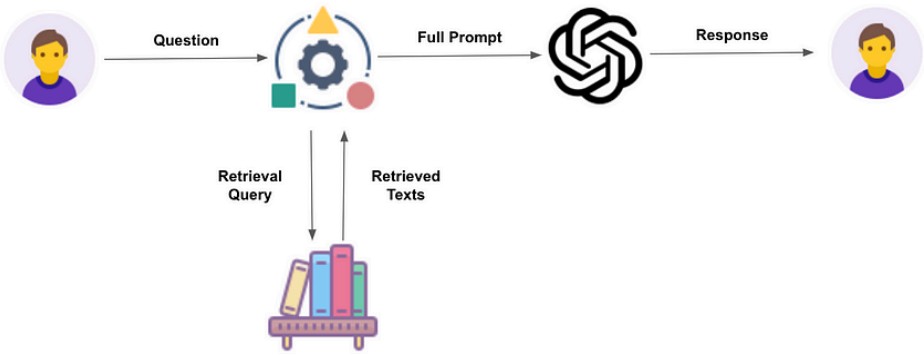
[287] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[288] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[289] https://www.datacamp.com/blog/top-small-language-models
[290] https://www.thesunflowerlab.com/llms-vs-slms/
[291] https://www.splunk.com/en_us/blog/learn/language-models-slm-vs-llm.html
[292] https://medium.com/@boutnaru/the-windows-concept-journey-windows-copilot-runtime-5f4c824d6f7b
[293] https://www.ibm.com/think/topics/small-language-models

RAG (Retrieval-Augmented Generation)

RAG (Retrieval-Augmented Generation) is an AI[294] framework which combines the capabilities of GenAI[295] and traditional retrieval systems like search\databases. By doing so we can get better results which are more accurate\up-to-date\relevant for specific needs[296]. By leveraging a RAG we don't have to regularly train the model on new data and update its parameters[297].

Overall, RAGs try to coupe with the known challenges of LLMs[298] such as: presenting false information when it does not have the answer, presenting out-of-date information, creating responses from non-authoritative sources and creating inaccurate responses due to terminology confusions. RAGs operate by first retrieving data from different data sources. Second, augment the LLM prompt using the retrieved data which. Thirud, send the full prompt to to an LLM which sends the response for the user question[299] - as shown below[300].

Lastly, there are different real-world use-case that can highly benefit from RAGs like: virtual assistants, content creation, medical diagnostics, clinical trial design optimization, code generation, sales automation, financial planning, customer support and knowledge management[301]. We can also use RAGs to build by trust by providing the model sources which it can cite. This allows users to check any claim given by the model as a response[302].



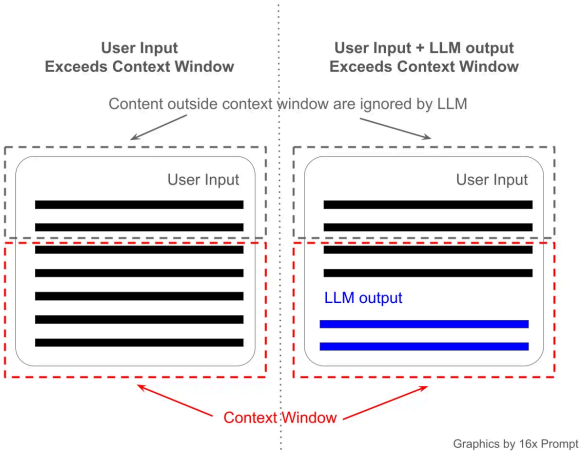
[294] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[295] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[296] https://cloud.google.com/use-cases/retrieval-augmented-generation
[297] https://www.codiste.com/what-is-a-retrieval-augmented-generation
[298] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[299] https://aws.amazon.com/what-is/retrieval-augmented-generation/
[300] https://nbkomputer.com/what-is-retrieval-augmented-generation-rag-embedding-model-vector/
[301] https://www.signitysolutions.com/blog/real-world-examples-of-retrieval-augmented-generation
[302] https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/

Context Window (aka Context Length)

Context Window (aka Context Length) is an attribute of an AI model. It is the amount of text (in tokens) that the model can consider\remember at any point of time. We can think about the context window like the "working memory" of an AI model. Thus, it determines how long of a conversation it can carry out without forgetting details from earlier in the exchange. Also determines the maximum size of documents or code samples that it can process at once[303] - as shown in the diagram below[304].

Overall, context windows are important because they help AI models recall information during a session. For example Gemini could process up to 32K tokens, while "Gemini 1.5 Pro" has a context window of up to 1 million tokens and as of April 2025 Google also tested up to 10 million token in their research[305]. For better understanding: "4K tokens is about 3K words and about 6 pages", "32K tokens are about 24 words and about 48 pages", "128K tokens are about 96Kwords and 192 pages" and "200K tokens are about 150K words and 300 pages"[306]. Based on those numbers we can conclude that "1M tokens are about 750K words and about 1500 pages".

Lastly, it is important to know that although a large context window has benefits like forming better connections between words and improved contextual information it also has drawbacks (the "context window paradox"). This is because it can cause: information overload (reducing focus and slowing performance), getting lost in data (prioritize edges and missing key middle info), understanding relationship becomes harder and poor management due to noise[307].



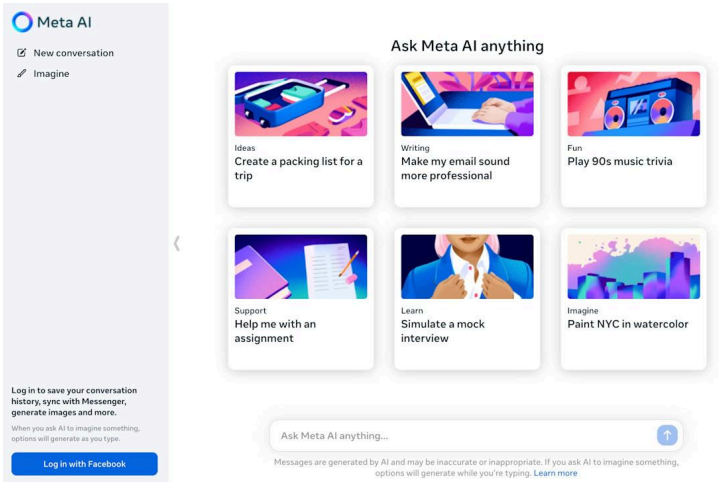
[303] https://www.ibm.com/think/topics/context-window
[304] https://prompt.16x.engineer/blog/claude-daily-usage-limit-quota
[305] https://blog.google/technology/ai/long-context-window-ai-models/
[306] https://lifearchitect.ai/gemini/
[307] https://datasciencedojo.com/blog/the-llm-context-window-paradox/

Llama (Large Language Model Meta AI)

Llama (Large Language Model Meta AI) is a family of LLMs[308] that has been released by "Meta" since February 2023. The models are trained on different parameter sizes between 1B-405B. By the way, Meta has added a virtual assistant feature to Facebook\WhatsApp based on the "Llama 3" model[309]. Beside the parameters sizes the models also differ in the context window size[310] for example: "Llama 3" has a 8K context window while Llama 3.1\Llama 3.2\Llama 3.3 has a context window of 128K.

Overall, Llama are open source models which we can fine-tune, distill and deploy anywhere. We can deploy a 1B/3B based model on end devices for different use-cases like summarizing. Also, we can use the 11B/90B models for multimodal interactions like transforming an existing image into something new[311].

Lastly, meta releases as part of "Llama" pre-trained models and instruction tuned models[312]. By the way, we can play with the "Meta AI" using a dedicated web application[313] - as shown in the screenshot below[314].



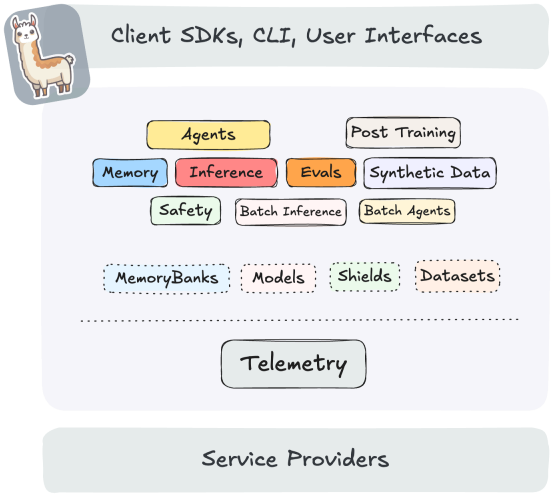
[308] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[309] https://en.wikipedia.org/wiki/Llama_(language_model)
[310] https://medium.com/@boutnaru/the-artificial-intelligence-journey-context-window-aka-context-length-9ead45714938
[311] https://www.llama.com/
[312] https://medium.com/@boutnaru/the-artificial-intelligence-journey-instruction-tuned-llm-f08585fded43
[313] https://meta.ai/
[314] https://zapier.com/blog/llama-meta/

Llama Stack

"Llama Stack" is a framework that can be used for developing and deploying GenAI[315] applications on top Meta's Llama models[316]. Llama stack provides APIs, components for different tasks (like safety, memory management and agent capabilities) - more on those in future writeups. Among the benefits of Llama stack we can find: standardization, synergy and smooth development[317].

Overall, llama stack ecosystem provides the following: unified API layer, plugin architecture, packaged verified distributions, multiple developer interfaces (CLI, SDK for Python\TypeScript\iOS\Android) and example standalone applications using Llama stack - as shown in the diagram below[318].

Lastly, there is also the "Llama Stack Playground" which showcases the capabilities and concepts of the Llama stack in an interactive environment. Also, it can be used for demoing end-to-end applications and provide a UI that helps users inspect and understand Llama Stack API providers and resources[319].



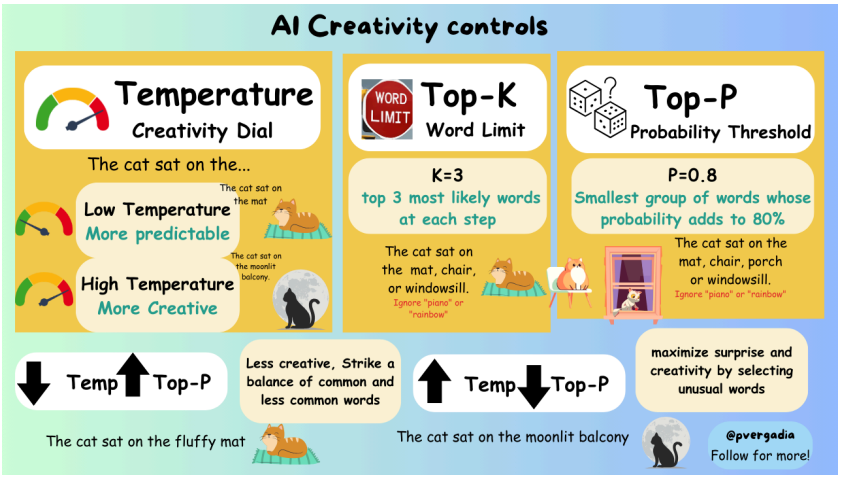
[315] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[316] https://medium.com/@boutnaru/the-artificial-intelligence-journey-llama-large-language-model-meta-ai-b40c9e0c1e65
[317] https://www.datacamp.com/tutorial/llama-stack
[318] https://github.com/meta-llama/llama-stack
[319] https://llama-stack.readthedocs.io/en/latest/playground/index.html

GenAI Model Parameters

Generative AI models[320] have the ability to set different parameters (affecting the sampling of the mode) in order to control the creativity\randomness of the generated results - as shown below[321]. Among those parameters are: “Temperature” (controls the randomness), “Top-k” (limiting the vocabulary) and “Top-p” (dynamic vocabulary limitations) and more[322]

Overall, those parameters influence the way in which content is generated[323]. The parameters can differ between models, thus we should check the relevant documentation of each model for specific information. More examples of such parameters are: stopping parameters (maximum output tokens and stop sequences), log probabilities of output tokens and token penalization parameters[324] - more on those in future writeups.

Lastly, when developing GenAI based apps we can set those parameters using the relevant APIs[325]. By setting them we can control how tokens are selected\predicted by the GenAI model[326].



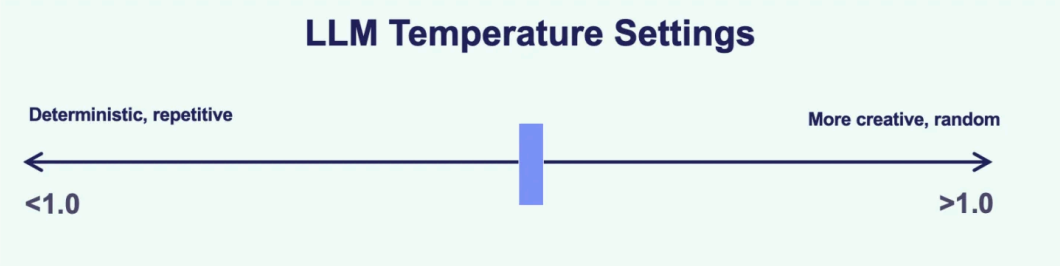
[320] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[321] https://www.thecloudgirl.dev/blog/mastering-ai-creativity-a-guide-to-temperature-top-k-and-top-p
[322] https://codefinity.com/blog/Understanding-Temperature,-Top-k,-and-Top-p-Sampling-in-Generative-Models
[323] https://www.phdata.io/blog/how-to-tune-llm-parameters-for-top-performance-understanding-temperature-top-k-and-top-p/
[324] https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/content-generation-parameters
[325] https://plainenglish.io/blog/mastering-llm-parameters-a-deep-dive-into-temperature-top-k-and-top-p
[326] https://ai.google.dev/gemini-api/docs/models/generative-models

LLM Temperature

A temperature is a number which controls the randomness of an LLM[327]. When using APIs for leveraging LLMs the temperature value has a specific range like from 0-2. We can think about it as adjusting how much the model is “explorative”\”conservative when answering[328].

Overall, LLMs iteratively generate tokens, which basically represent words\parts of words. Foreach optional token there is an assigned likelihood, the goal of the temperature is to dictate how to weigh those likelihoods[329].

Lastly, by default temperature can have a value of 1 (like in OpenAI’s ChatGPT), which balances between randomness and determinism. If we lower the value we get completions which are less random. Thus, if the temperature value approaches zero the model is more deterministic and repetitive[330]. The value range of temperature is divided to: low temperature (<1.0), high temperature (>1.0) and 1.0 - as demonstrated in the diagram below[331].



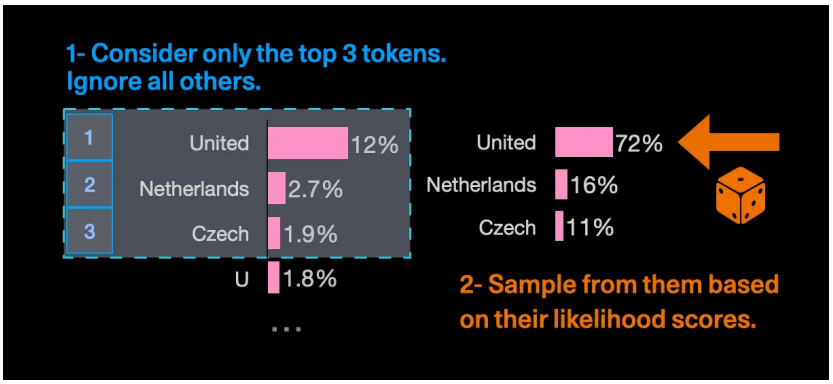
[327] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[328] https://towardsdatascience.com/a-comprehensive-guide-to-llm-temperature/
[329] https://www.vellum.ai/llm-parameters/temperature
[330] https://www.hopsworks.ai/dictionary/llm-temperature
[331] https://www.iguazio.com/glossary/llm-temperature/

TopK (Top-K)

TopK (aka “Top K”\”Top-K” is a parameter\setting which is supported by some LLMs[332]. By using it we can determine how many (K) of the most likely tokens should be considered when generating a response (they are sampled based on their likelihood) - as shown in the diagram below[333]. Due to the fact LLMs are trained on huge corpuses of text they feature massive dictionaries. There are specific words that are more likely to appear than others[334].

Overall, if for example we set “K=1” we basically state we want the very best token (the one with the most probability to be next). By the way, there is a default value of “K=50” used by different platforms like Huggingface’s libraries when using generation configuration[335].

Lastly, there are a couple of benefits when using “TopK” such as: reducing the chances of generating highly improbable words and providing a more controlled and predictable output. Among the drawbacks we have less diversity in outputs which can also lead to repetitiveness[336].



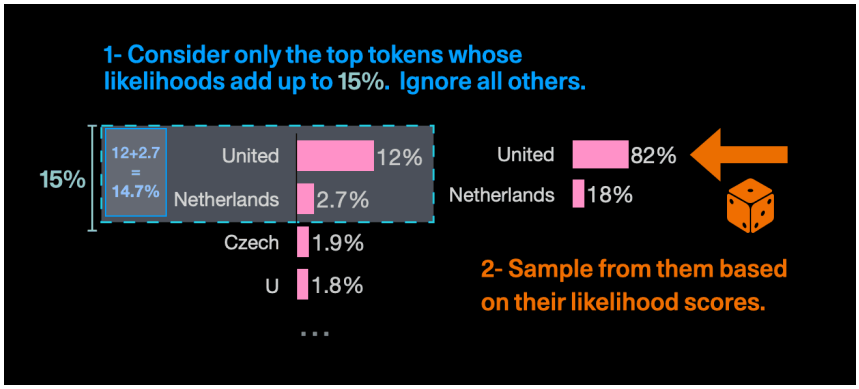
[332] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[333] https://cohere.com/blog/llm-parameters-best-outputs-language-ai
[334] https://www.vellum.ai/llm-parameters/top-k
[335] https://www.phdata.io/blog/how-to-tune-llm-parameters-for-top-performance-understanding-temperature-top-k-and-top-p/
[336] https://www.alphanome.ai/post/top-k-and-top-p-in-large-language-models-a-guide-for-investors

TopP (Top-P)

TopP (aka “Top P”\”Top-P” is a parameter\setting which is supported by some LLMs[337]. As opposed to “TopK”[338] which samples from the most likely K words, in TopP the sampling phase chooses from the smallest possible set of words whose cumulative probability exceeds the probability “p”[339] - as shown in the diagram below[340].

Overall, “TopP” is also called “nucleus sampling”. It is very useful for managing the reading level (of the generated text) and generating multiple variations for a specific response. We can set the TopP in different GenAI platforms: OpenAI (“top_p” which ranges from 0.0-1.0), Antropic (“top_p” which ranges from 0.0-1.0), Gemini (“topP” which ranges from 0.0-1.0) and more[341].

Lastly, “TopP” has also drawbacks such as: it is less predictable than TopK due to the fact the number of words in the subset can vary and if “P” is too high, implausible or nonsensical output can be generated. Among its benefits are (but not limited to): TopP is more dynamic than TopK and it captures more diversity in outputs, which reduces repetitiveness[342].



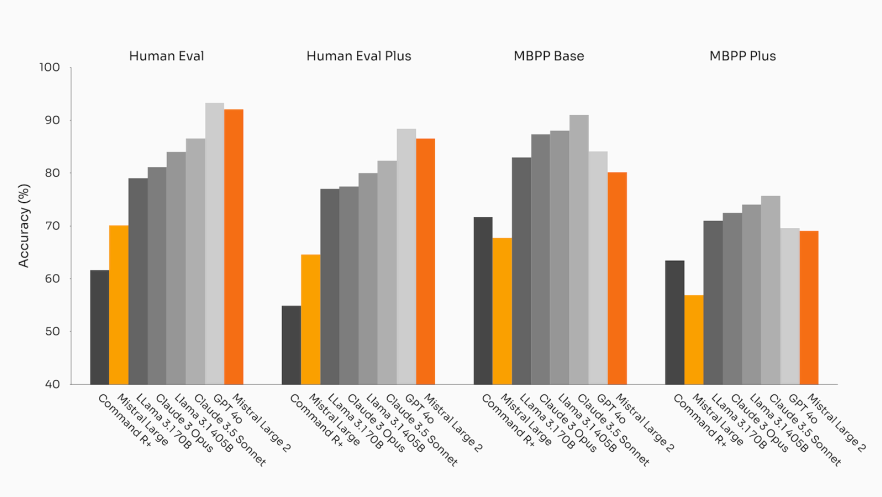
[337] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[338] https://medium.com/@boutnaru/the-artificial-intelligence-journey-topk-top-k-a926e61b96e2
[339] https://community.openai.com/t/a-better-explanation-of-top-p/2426
[340] https://cohere.com/blog/llm-parameters-best-outputs-language-ai
[341] https://www.vellum.ai/llm-parameters/top-p
[342] https://www.alphanome.ai/post/top-k-and-top-p-in-large-language-models-a-guide-for-investors

GenAI Model Benchmarks

As “Generative AI” models[343] are becoming more and more popular, we need a way to compare between them. Benchmarks are series of tests which are used to measure the capabilities of AI models[344]. By using the benchmarks results we can decide what model to use based on our requirements[345].

Overall, when benchmarking GenAI models we can cluster the measurements into different categories: AI quality, latency (time to first token), throughput (generated tokens per second) and cost. Of course each category has different metrics that are calculated[346].

Lastly, we can cluster benchmarks by their category: “Reasoning & Knowledge” (like MMLU-Pro and Humanity's Last Exam), “Coding” (like LiveCodeBench, SciCode and HumanEval), “Quantitative Reasoning” (like MATH-500), “Scientific Reasoning” (like GPQA Diamond) and “Competition Math” (like AIME 2024) - more on those benchmarks in future writeups. For each evaluation measured higher is better[347]. An example of such benchmarks results is shown below[348].



[343] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[344] https://medium.com/@charles.ide/an-introduction-to-genai-benchmarks-45f8357f0bdc
[345] https://cloud.google.com/vertex-ai/generative-ai/docs/models/evaluation-overview
[346] https://techcommunity.microsoft.com/blog/aiplatformblog/compare-and-select-models-with-new-benchmarking-tools-in-azure-ai-foundry/4292308
[347] https://artificialanalysis.ai/models
[348] https://tinyurl.com/hncx8k95

MMLU (Measuring Massive Multi-task Language Understanding)

MMLU (Measuring Massive Multi-task Language Understanding) is a benchmark[349] for evaluating the capabilities/accuracy of LLMs[350]. MMLU contains multiple-choice questions running in 57 different academic subjects. Among those subjects are: mathematics, medicine, computer science, US history, computer science, law,nutrition, formal logic and philosophy. The benchmark was published in 2020 by a team of researchers (Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song and Jacob Steinhardt)[351].

Overall, in case we want to download the dataset and\or review MMLU we can use HuggingFace for that - as shown in the screenshot below. We could see examples for models trained/fine tuned based on MMLU. Moreover, we can get to other spaces\projects that are using MMLU as part of their evaluation flow[352].

Lastly, researchers that have manually reviewed questions as part of MMLU found errors that obscure the true capabilities of LLMs. For example 57% of the analysed questions in the Virology subset contain errors[353]. For checking out papers using\leveraging MMLU we can use PapersWithCode[354].

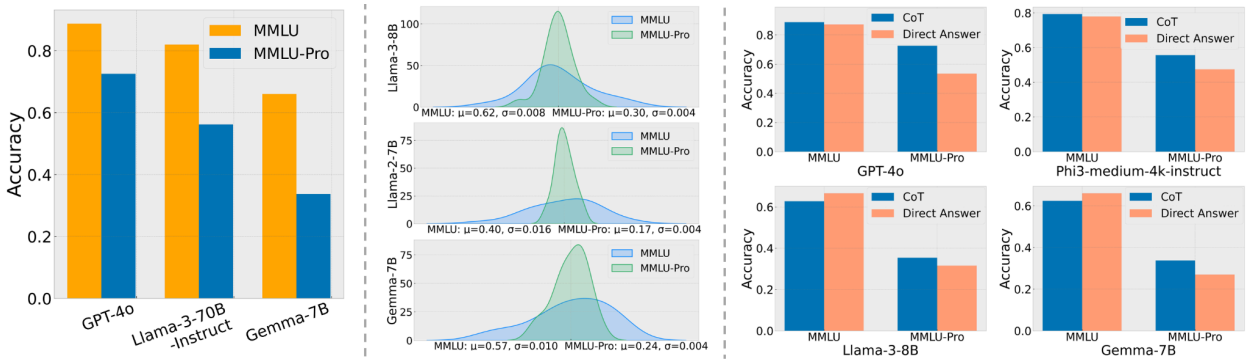
[349] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-model-benchmarks-659775c28411
[350] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[351] https://arxiv.org/abs/2009.03300
[352] https://huggingface.co/datasets/cais/mmlu
[353] https://arxiv.org/abs/2406.04127
[354] https://paperswithcode.com/dataset/mmlu

MMLU-Pro (Measuring Massive Multi-task Language Understanding Professional)

MMLU-Pro (Measuring Massive Multi-task Language Understanding Professional) is a benchmark framework[355] for evaluating the capabilities/accuracy of LLMs[356]. MMLU-Pro contains 12K multi-choice questions across different categories such as: engineering, computer science, health, economics, physics, math and more[357].

Overall, MMLU-Pro was introduced in 2024 (with a paper submitted by Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue and Wenhu Chen). It is an enhanced dataset extending the knowledge-driven MMLU benchmark[358] by integrating more challenging and reasoning-focused. It also expanded questions and the choice set from four to ten options[359].

Lastly, based on results MMLU-Pro reduces accuracy by 16%-33% when compared to MMLU and shows greater stability under varying prompts. Also, models utilizing Chain of Thought (CoT) reasoning achieved better performance on MMLU-Pro compared to direct answering (in contrast to MMLU). This demonstrates that MMLU-Pro includes more complex reasoning questions[360] - as shown in the graphs below.



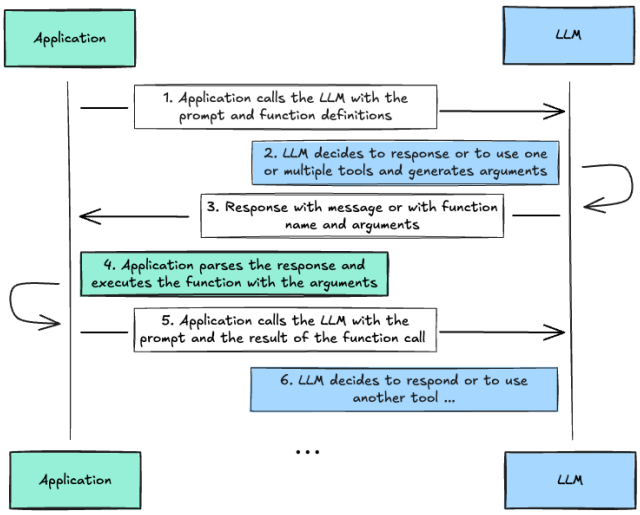
[355] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-model-benchmarks-659775c28411
[356] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[357] https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro
[358] https://medium.com/@boutnaru/the-artificial-intelligence-journey-mmlu-measuring-massive-multi-task-language-understanding-03e3d9b5b21b
[359] https://arxiv.org/abs/2406.01574
[360] https://github.com/TIGER-AI-Lab/MMLU-Pro

Tool Calling (aka Function Calling)

Tool calling (aka function calling) is the ability of AI models to interact with external tools in order to enhance their functionality. This can be done by leveraging APIs (Application Programming Interface). By doing so the AI model is not dependent only on pre-trained data due to that fact it can query databases, fetch real-time information, execute operations and more. This can be done in the following steps: recognize the need for a tool, select a tool, construct and send the query, process the response, present the information or take action and refine the query if needed[361] - the flow is shown in the diagram below[362].

Overall, we can leverage function calling for different use-case such as (but not limited to): integration with external API (like getting weather data, converting address to latitude/longitude coordinates, converting currencies and more), building advanced chatbots (answering customers questions about services\products, financial assistant and more), querying databases using natural language, multi-model function calling (use images\videos\PDFs as input for functions) and much more[363].

Lastly, examples (but not limited to) of LLMs[364] with function calling support are: OpenAI's GPT-4 and GPT-3.5 turbo, Google's Gemini (using Vertex AI and Google AI Studio), Antropic's Claude 3 family of LLMs, Cohere's Command R and Command R+ LLMs, Mistral 7B, Fireworks FireFunction and Gorilla OpenFunctions[365]. Also, function calling is a fundamental building block of agentic AI - more on that in future writeups.



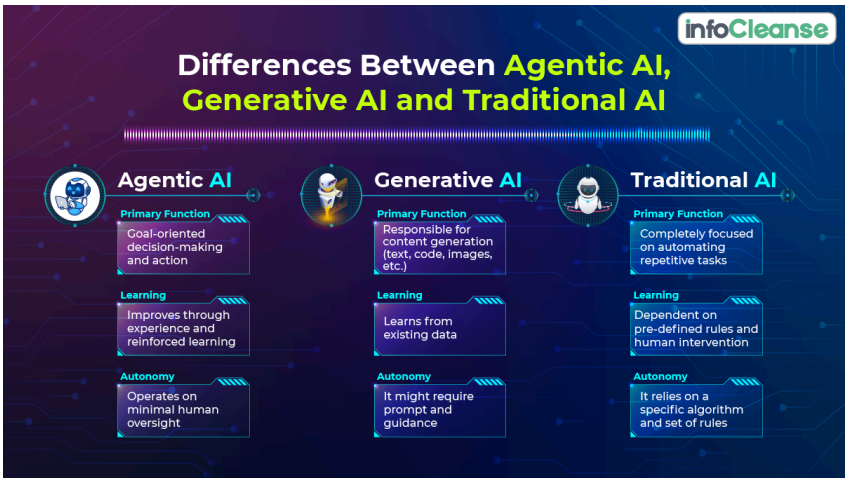
[361] https://www.ibm.com/think/topics/tool-calling
[362] https://huggingface.co/docs/hugs/en/guides/function-calling
[363] https://cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling
[364] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[365] https://thenewstack.io/a-comprehensive-guide-to-function-calling-in-llms/

Agentic AI (Agentic Artificial Intelligence)

Agentic AI is based on GenAI[366] by using LLMs[367] in order to operate - as comparison between traditional AI, GenAI and agentic AI is shown below[368]. In contrast to traditional AI models which require human intervention\work with predefined contracts, agentic AI is autonomous incorporating a goal-driven behaviour. For example a GenAI model can create text\images\videos\code while an agentic AI system can leverage the produced data for performing complex tasks by calling external functions\tools[369].

Overall, we can summarize the steps performed by agentic AI to the following: preparation (collecting data from the surrounding environment using APIs\databases\user interactions\etc), reasoning (intercepting user queries and understanding the broader context), goal setting (set objectives based on goals\user's inputs), decision making (evaluating multiple possible actions and choosing the optimal one based on different calculations and metrics), execution, learning and adoption (evaluating the results and improving further decisions) and orchestration (management of systems and agents)[370] - more on those in future writeups.

Lastly, we can say agentic AI is a probabilistic technology with the ability to adapt to changing environments and events. It uses patterns and likelihood for making decisions and taking actions. Among the benefits of agentic AI we can find: increased efficiency and productivity and enhanced customer experience. Examples of use-case for agentic AI are (but not limited to): streamlining insurance claims flows, optimizing logistic and supply chain management, empowering financial decision making, accelerating and optimizing testing and transforming customer service and support[371].



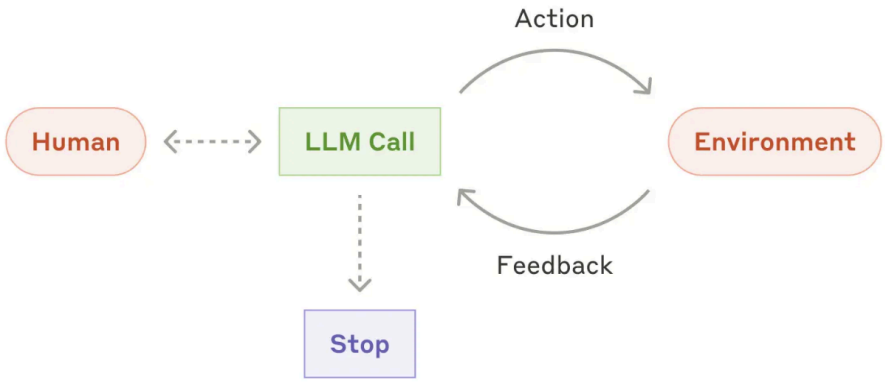
[366] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[367] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[368] https://www.infocleanse.com/agentic-ai-in-business/
[369] https://medium.com/@boutnaru/the-artificial-intelligence-journey-tool-calling-aka-function-calling-44d9ef5da397
[370] https://www.ibm.com/think/topics/agentic-ai
[371] https://www.uipath.com/ai/agentic-ai

AI Agent (Artificial Intelligence Agent)

An AI agent is a system which is capable of performing autonomous tasks on behalf of another system/user. This is done by leveraging different workflows and function calling\tool usage[372]. There are different types of AI agents such as (but not limited to): simple reflex agents (does not interact with other agents nor hold any memory), model-based reflex agents (leverages their current perception and memory to maintain an internal model of the world like a robot vacuum cleaner), goal-based agent (has an internal model of the world and a goal\set of goals to achieve like a navigation system recommending the fastest route), utility-based agent (select the sequence of actions that reach the goal and also maximize utility or reward) and learning agent (ame capabilities as the other agent types but are unique in their ability to learn)[373].

Overall, AI agents (based on LLMs) understand complex inputs, engage with reasoning\planning\external tools and recover from errors. When the task is clear an agent plans and operates independently. By the way, agents can pause for human feedback and interaction at specific checkpoint or in case of blockers. Thus a general architecture of an AI agent is shown below[374].

Lastly, among the use-cases which are relevant for AI agent we can find the following (not limited to): makating (improved decision making, precision targeting, task automation, content automation and conversational chatbots), sales (meeting scheduling, sales prospecting and sales content creation), technical support (personalised assistance, task categorization and reduced admin), finance (identifying fraud patterns and document verification for KYC), HR (inclusive hiring, employee sentiment analysis and workforce optimization), logistics (optimized delivery routing, warehouse automation and optimized stock levels) and customer service[375].



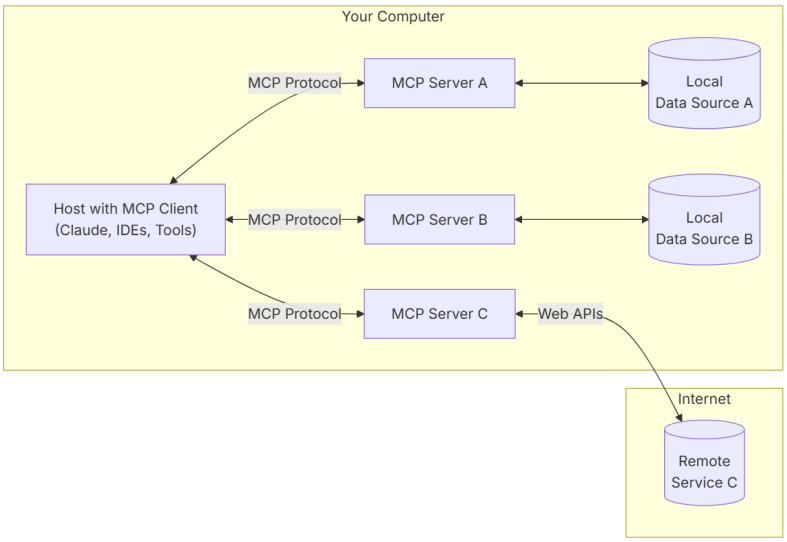
[372] https://medium.com/@boutnaru/the-artificial-intelligence-journey-tool-calling-aka-function-calling-44d9ef5da397
[373] https://www.ibm.com/think/topics/ai-agents
[374] https://www.anthropic.com/engineering/building-effective-agents
[375] https://www.ai21.com/knowledge/ai-agent-use-cases/

MCP (Model Context Protocol)

MCP (Model Context Protocol) is an open protocol\standard\framework which was created to standardize how applications can provide context to LLMs[376]. Thus, we can use MCP to connect AI models to different tools/data sources and build agents\complex workflows on top of LLMs. MCP was introduced by Antropic in November 2024, it was described as "USB-C of AI Applications"[377].

Overall, MCP is based on a client-server architecture which allows a host application to connect to multiple servers - as shown in the diagram below. Let us go over the different components in the architecture. First, MCP hosts are programs (Cursor\Claude Desktop\etc) or other AI tools which access data by leveraging MCP. Second, a MCP client connects to a server. Third, the MCP server exposes specific capabilities using the model context protocol. Fourth, local data sources are databases\services\files\etc that an MCP server can access securely. Fifth, remote services and systems (Internet accessible) that the MCP server can access using APIs[378].

Lastly, we can access MCP's protocol specification, documentation and implementations in different programming languages (like C#, Kotlin, Python and more) as part of the "Model Context Protocol" GitHub repository[379].



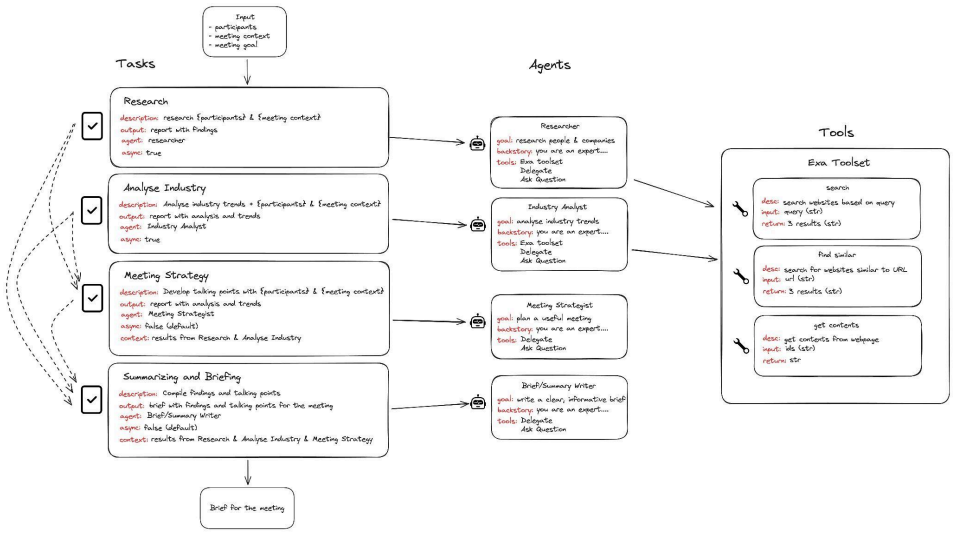
[376] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[377] https://en.wikipedia.org/wiki/Model_Context_Protocol
[378] https://modelcontextprotocol.io/introduction
[379] https://github.com/modelcontextprotocol

CrewAI

CrewAI is a platform for "multi-agent" automation. It allows streamlining workflows with AI agents. Using it we can build and deploy automated workflows using any LLM and cloud platform. Among the different use cases in which we can use CrewAI are: task automation, human resources, healthcare, finance, strategic planning, marketing, brand positioning, data enrichment and more[380].

Overall, CrewAI is a lean, lightning-fast Python framework built from scratch. By using it, developers are empowered with simplicity and precise control. This is ideal for creating autonomous AI[381]. For more information about multi-agent systems and how to build them using CrewAI there are relevant online materials[382].

Lastly, when creating a crew we have the following components: tasks (we want to perform, each task is assigned to an agent), agent (AI agent which is an expert of a specific task), tools (operations that our agent can perform like: searching, summerzing, translating and more) and process (the way in which are agents are going to work together) - an example is shown below[383].



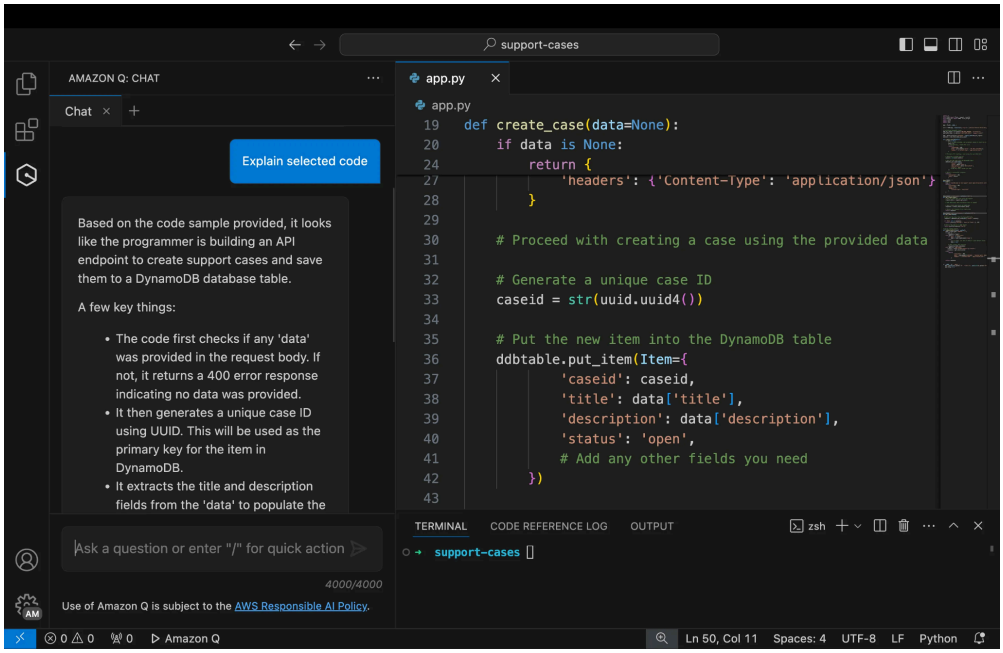
[380] https://www.crewai.com/
[381] https://github.com/crewAIInc/crewAI
[382] https://learn.crewai.com/
[383] https://alejandro-ao.com/crew-ai-crash-course-step-by-step/

Amazon Q Developer

"Amazon Q Developer" is a GenAI[384] powered assistant for software development. It is used as part of our code editor (like by leveraging extensions) such as: JetBrains, VS Code, Visual Studio, Eclipse - as shown in the screenshot below[385]. We can also leverage "Amazon Q Developer" using the command line. Among the benefits which "Amazon Q Developer" provides are: coding faster, improving readability, improving security, customizing code, building autonomous agents and having expert assistance[386].

Overall, as of today Amazon Q Developer has two pricing tiers: free and pro. They are features available only in the pro tier such as: analytics dashboard, user management, policy management and IP indemnity. Also, there are features which are available in both but have different limits like Chat (project-wide context) which is limited to 50 chat interactions per month as part of the free tier. Of course there are features which are identical like code compilation and general Q&A[387].

Lastly, because it had been created by Amazon it may help us with specific AWS operations we want to perform. It can act as an expert for managing\optimizing AWS environments. Examples are: analyze network reachability, EC2 instance operations and more[388].



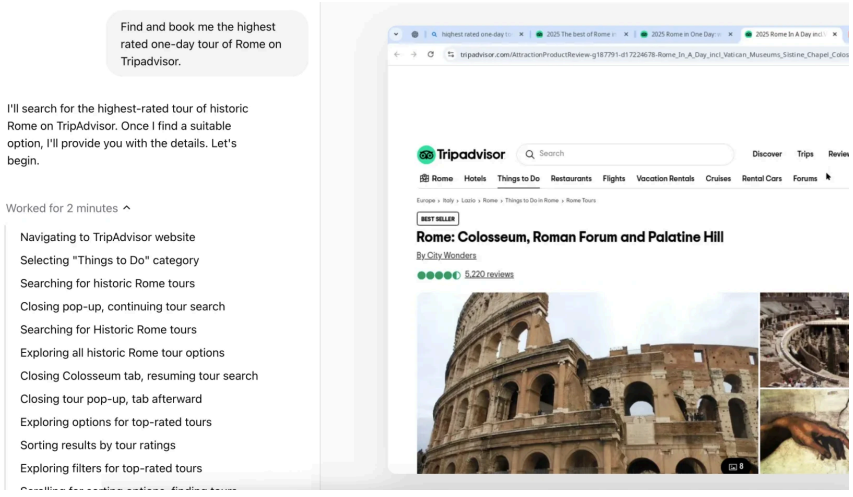
[384] https://medium.com/@boutnaru/the-artificial-intelligence-journey-genai-generative-artificial-intelligence-29d1228e905e
[385] https://community.aws/content/2fVw1hN4VeTF3qtVSZHfQiQUS16/getting-started-with-amazon-q-developer-in-visual-studio-code
[386] https://aws.amazon.com/q/developer/
[387] https://aws.amazon.com/q/developer/pricing/
[388] https://aws.amazon.com/q/developer/operate/

OpenAI Operator

Operator was created by OpenAI. It allows an agent that can use its own web browser for performing tasks on behalf of users (as shown below). Operator can be asked to handle a wide variety of repetitive browser tasks such (filling out forms\ordering groceries and more). As of today (mid 2025) it is a research preview which allows learning from our users and the broader ecosystem for refining and improving[389].

Overall, it is powered using a new model called CUA (Computer-Using Agent). It is built on GPT-4o (vision capabilities) for analyzing screenshots and interacting with browser controls. In case there is a need for authenticating (passwords\CAPTCHA\etc) Operator pauses and passes the control to the user. Due to the research nature there are specific high-stake automatic tasks which are blocked such as: conducting financial transactions, sending emails and deleting a calendar event[390].

Lastly, as of today Operator is only available in the US for “Pro” tier users. Due to safety and privacy concerns the following features are included: takeover mode, user conformations before completing a significant action, task limitations, watch mode, training opt out and transparent data management like deleting browsing data[391] - more on those and others in future writeups.



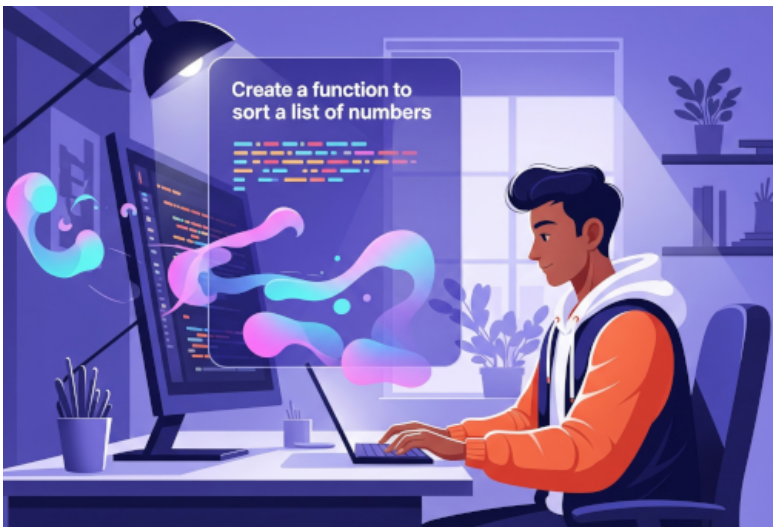
[389] https://help.openai.com/en/articles/10421097-operator
[390] https://openai.com/index/computer-using-agent/
[391] https://openai.com/index/introducing-operator/

Vibe Coding

"Vibe Coding" is the concept of leveraging AI[392] technologies such as LLMs[393] for helping in automating coding works. Hence, in vibe coding users express their intentions and plain text\speech and the AI tools perform that into source code - as shown in the image below (created using Google Gemini). We can summarize the phases of vibe coding into the following: choosing AI coding platform, defining requirements, code refinement and final code review and shipping[394].

Overall, advocates of vibe coding state it provides amateur programmers to create software without extensive training and skills. By the way, the term was introduced by Andrej Karpathy in February 2025[395]. Also, "Y Combinator" published that ¼ of the of startup companies in its Winter 2025 batch had codebases that were 95% created by AI based technologies[396].

Lastly, examples of vibe coding tools/platforms are (but not limited to): Cody (by Sourcegraph), Bolt.new (by StackBlitz), Cursor (by Anysphere), v0 (by Vercel), GoCodeo, Softgen (by Kortix) AI and Replit[397]. By using vibe coding we get the ability to prototype quickly, reduce risks of software projects, maximize impact and more[398].



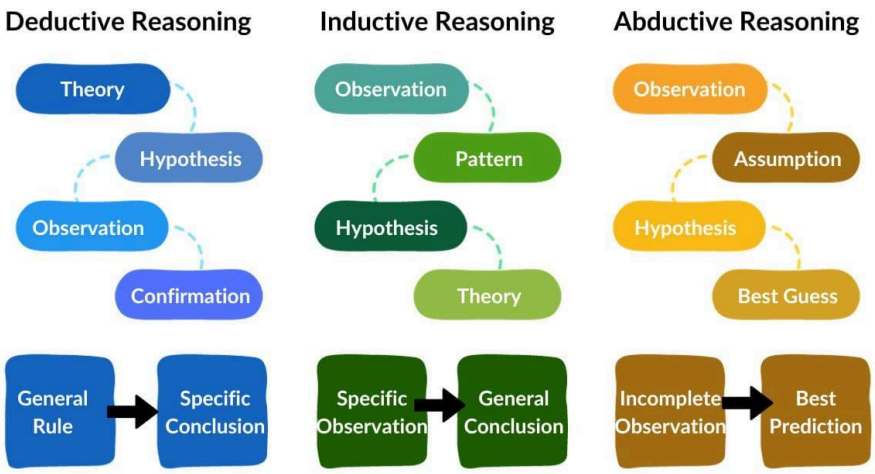
[392] https://medium.com/@boutnaru/introduction-to-ai-artificial-intelligence-8c71d4d25320
[393] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[394] https://www.ibm.com/think/topics/vibe-coding
[395] https://en.wikipedia.org/wiki/Vibe_coding
[396] https://techcrunch.com/2025/03/06/a-quarter-of-startups-in-ycs-current-cohort-have-codebases-that-are-almost-entirely-ai-generated/
[397] https://dev.to/therealmrmumba/top-10-vibe-coding-tools-that-feel-like-magic-in-2025-1md
[398] https://www.trickle.so/blog/hidden-benefits-of-vibe-coding-that-most-teams-miss

Reasoning Models

"Reasoning Models" are LLMs[399] which are trained using reinforcement learning[400] in order to perform complex reasoning. Reasoning models "think" before they answer. Thus, they produce a long internal chain of thought before responding. Due to that, these types of models excel in coding, complex problem solving, scientific reasoning and multi-step planning[401].

Overall, the ability of reasoning models to process things step by step are useful for autonomous tasks. Reasoning models are better than standard LLMs in tasks which require logical reasoning. However, they are much slower. For example, a typical reasoning query can take 2-3 minutes[402].

Lastly, using AI reasoning we leverage available information to generate predictions, make inferences and draw conclusions. Examples of such models are (but not limited to): DeepSeek-R1, Google's Gemini 2.0 Flash Thinking, IBM's Granite 3.2 and OpenAI's o1 series and o3-mini. There are different reasoning strategies like (but not limited to): "Abductive Reasoning", "Agentic Reasoning", "Analogical Reasoning", "Deductive Reasoning" and "Fuzzy Reasoning"[403]. A short demonstration regarding them is shown in the diagrams below[404].



[399] https://medium.com/@boutnaru/the-artificial-intelligence-llm-large-language-model-f3e1a3fb15d6
[400] https://medium.com/@boutnaru/the-artificial-intelligence-journey-reinforcement-learning-87f8c0cc8099
[401] https://platform.openai.com/docs/guides/reasoning
[402] https://zapier.com/blog/ai-reasoning/
[403] https://www.ibm.com/think/topics/ai-reasoning
[404] https://spotintelligence.com/2024/01/16/knowledge-representation-and-reasoning-ai/