INTRODUCTION TO AI

AI-for-Education.org

ABOUT THIS DOCUMENT

Artificial Intelligence can seem overwhelming, given the pace of the advancement and the complexity of the methods used. We at AI-for-Education.org believe that while how AI works is complex, the core principles of AI should be accessible to all – especially for those who are using these tools to help children learn.

To help, we've developed this introduction to AI – which walks you through the main definitions, main techniques, and the history of how we got where we are today. It's as non-technical as possible in this field – but an important read for all of you who will build or use AI tools in the future.

CONTENTS

INTRODUCTION TO AI

1. Artificial Intelligence: Definitions

To begin, we introduce some key terms, and how they relate to each other. We will expand on the definition of these concepts in the next section.

ARTIFICIAL INTELLIGENCE (AI) is a broad discipline which aims to build computational systems that can perform complex tasks which typically require human cognition. This includes things like understanding languages, recognising pictures or patterns, solving problems, predicting the future (forecasting), or driving a car.

MACHINE LEARNING (ML) is a subset of AI. Machine Learning involves algorithms which learn from data. Most current AI is based on the machine learning framework.

DEEP LEARNING is a subset of ML which uses a particular class of algorithm: artificial neural networks.

GENERATIVE AI refers to ML models that can produce new content: for example, images or text. Most Generative AI is based on Deep Learning models.

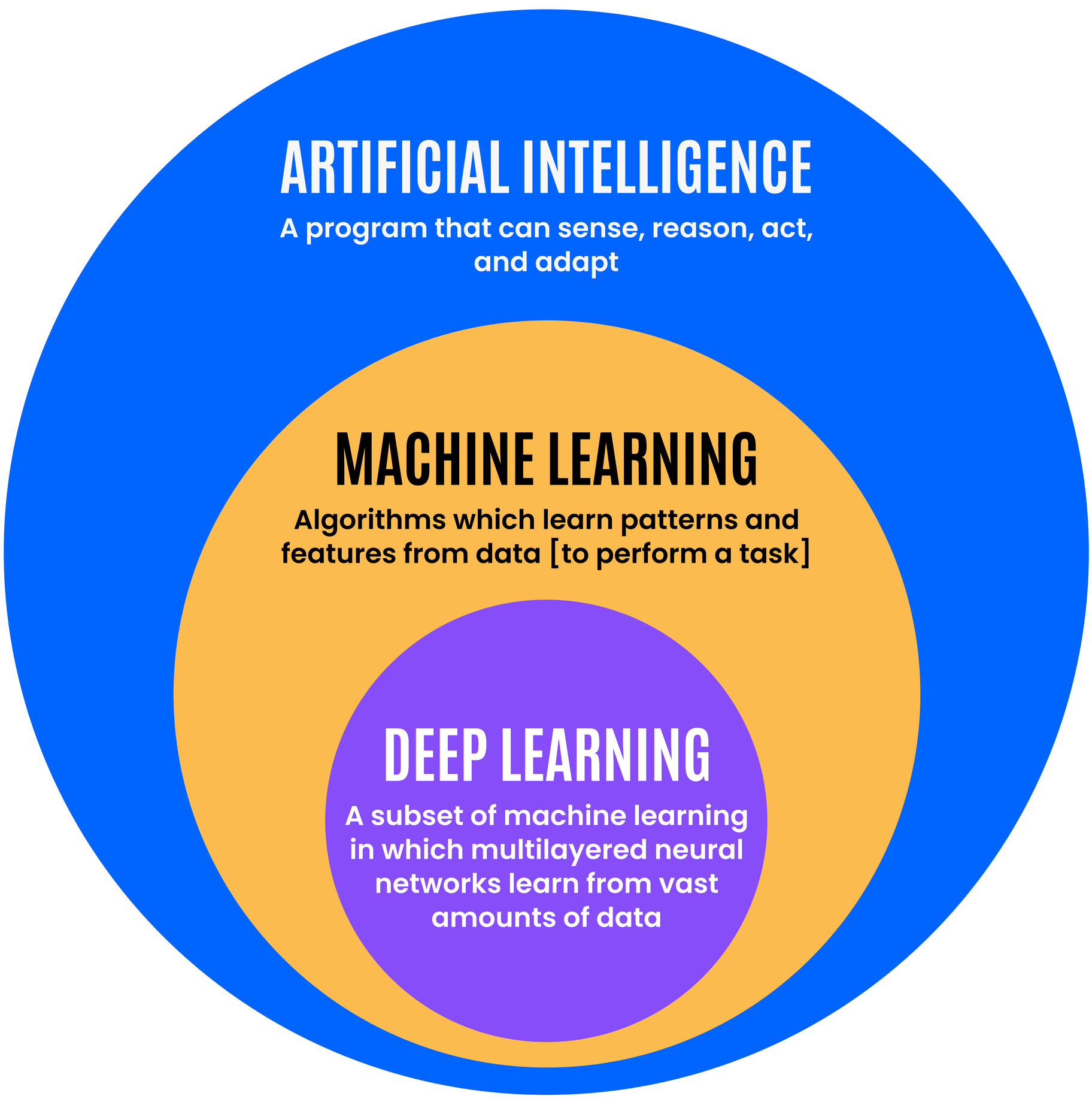


Figure 1: Definition and relationship of key terms.

ARTIFICIAL GENERAL INTELLIGENCE (AGI)

Artificial General Intelligence (AGI). AGI refers to system capable of flexible problem solving, beyond the capabilities of a typical human across a wide range of tasks. The exact point at which a system should be considered AGI is very much debated, but several tests have been proposed to identify this. As historically developed tests have been passed: beating a grandmaster at chess (IBM's Deep Blue, 1997), beating humans at the more computationally challenging game of Go (DeepMind's AlphaGo, 2015), passing the Turing test (Eugene Goostman, 2014), new tests are proposed. The current consensus is that while progress is accelerating, current systems should not yet be considered human-level AGI. Further discussion of AGI is beyond the scope of this introduction, but it is a useful term to keep in mind.

Several tests have been proposed to demonstrate human-level AGI.

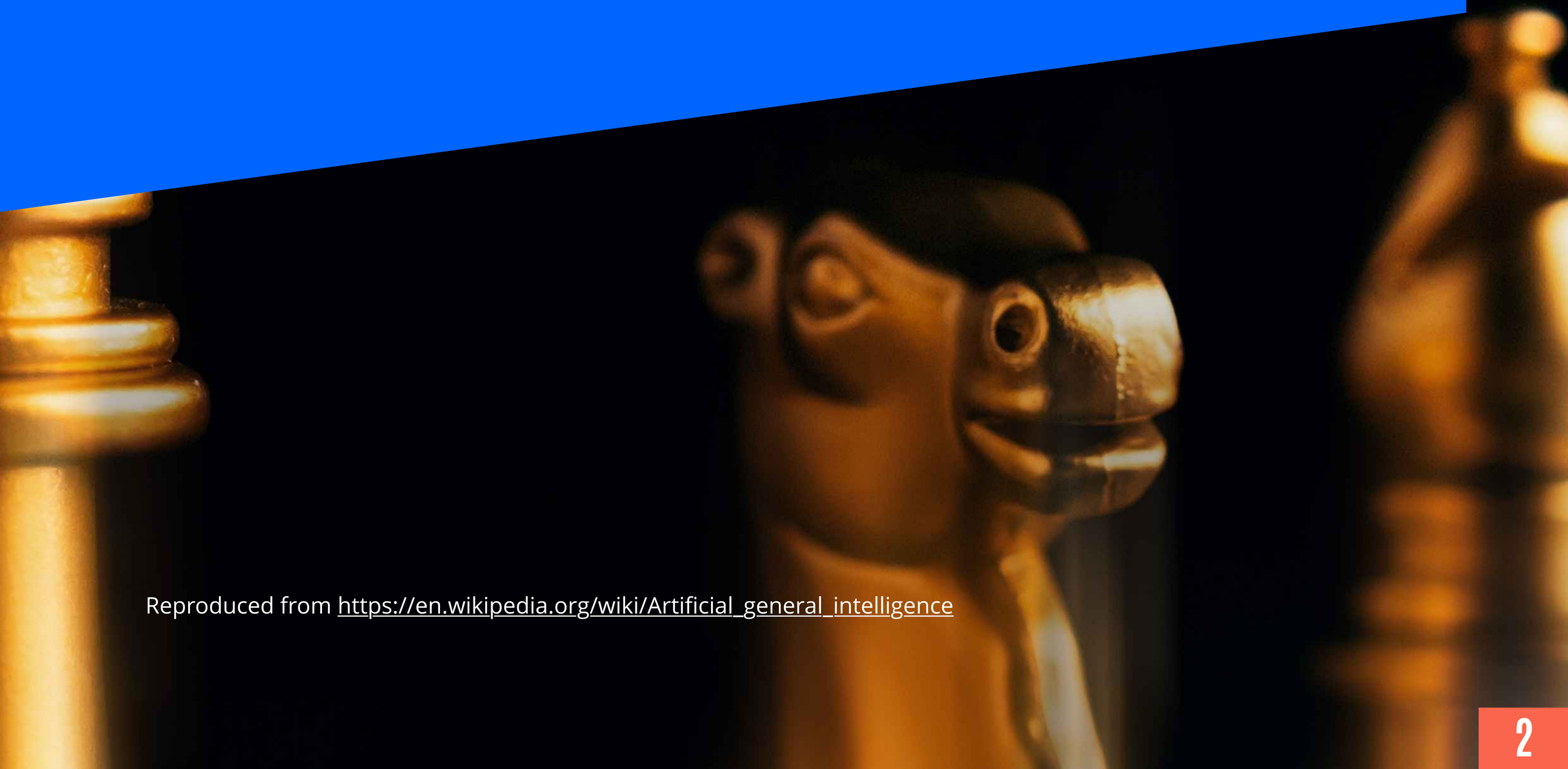
The Turing Test
A machine and a human both converse unseen with a second human, who must evaluate which of the two is the machine, which passes the test if it can fool the evaluator a significant fraction of the time. The Turing test was first passed in 2014, by a chatbot AI called Eugene Goostman.

The Robot College Student Test
A machine enrols in a university, taking and passing the same classes that humans would, and obtaining a degree. LLMs can now pass university degree-level exams without even attending the classes.

The Employment Test
A machine performs an economically important job at least as well as humans in the same job. AIs are now replacing humans in many roles as varied as fast food and marketing.

The Ikea Test
Also known as the Flat Pack Furniture Test. An AI views the parts and instructions of an Ikea flat-pack product, then controls a robot to assemble the furniture correctly.

The Coffee Test
A machine is required to enter an average American home and figure out how to make coffee: find the coffee machine, find the coffee, add water, find a mug, and brew the coffee by pushing the proper buttons.



Reproduced from https://en.wikipedia.org/wiki/Artificial_general_intelligence

2. Machine Learning

2.1 TYPES OF MACHINE LEARNING

Machine Learning (ML) refers to a family of algorithms which learn from data. There are three main families of ML problems: reinforcement learning, supervised learning, and unsupervised learning.

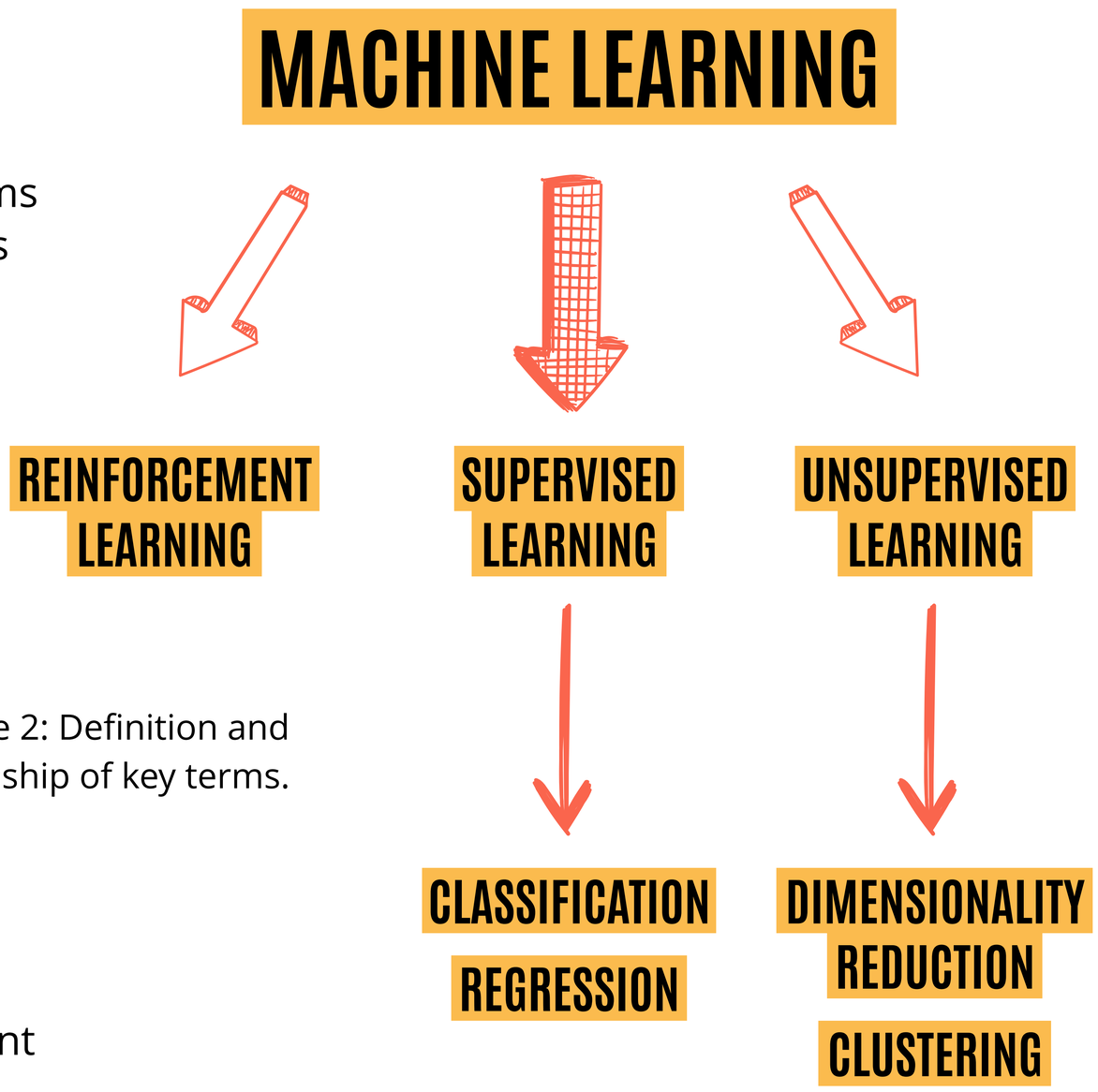


Figure 2: Definition and relationship of key terms.

REINFORCEMENT LEARNING is a type of ML which models an agent learning by trial and error. The agent uses an algorithm to choose which action to take, based on current observations of the environment. The chosen actions have consequences, and the RL model learns how to select their actions to maximize some metric, called the *reward*. This might be points in a game, or some other measure of how well the agent is doing. The algorithm tries to find the optimum trade-off between exploring (i.e. trying new actions) vs *exploiting* actions that are known to do well. It is important to keep in mind the particular reward that the agent is optimising. If the reward is not well calibrated to the behaviours researchers are looking for, this can lead to the agent taking unexpected short cuts, or behaving in a way that was not intended. For example, when RL agents are applied to computer games, they can often find glitches or bugs in the game which they exploit to get a better score.

SUPERVISED LEARNING is about learning a mapping, between examples of data and some associated outcome or property. For example, for the problem of face detection, the data points are individual photographs, and the output is a label with two values: a yes or no answer to the question “is there a face in this picture?”. If the outcome has discrete values like this face detection example, the problem is called *classification*. In the outcome is a continuous value then it is called *regression*. For example, predicting a student’s final mark based on earlier coursework and assessments would be a regression problem. For both types of supervised learning, the model is trained with example data items paired with the desired output for each: i.e. a large set of photographs together with labels with tell whether a face is present or not. In supervised learning, the supervision comes from the labelled training data set, which tells the model what to do, by setting the parameters. As we will see, the training data has a huge influence on the behaviour of a model, so it is important to understand this step.

UNSUPERVISED LEARNING is about learning something about the structure of a large dataset, without any explicit labels or output values to predict. Two major sub-classes of unsupervised learning are *dimensionality reduction* and *clustering*. Dimensionality reduction is about describing a high-dimensional data set in a simpler way. For example, representing each data point with just a few numbers rather than hundreds of thousands, in a way that preserves as much of the important structure, such as the relationships between different data points, as possible. Clustering is about grouping objects (for example photographs) in such a way that objects within a group (or cluster) are more similar to each other than they are to items in other groups.

2.2 WHAT IS A ML MODEL

This section unpacks the important parts of a ML model from the supervised or unsupervised learning families. Reinforcement learning algorithms work in a slightly different framework which we won't cover here.

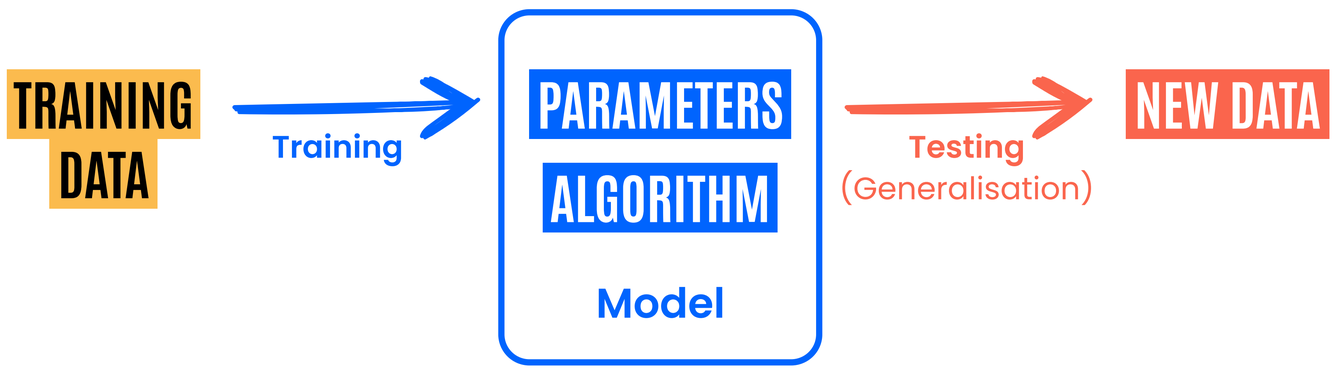


Figure 3: A machine learning model consists of an algorithm, together with some parameters. The parameters are learned from the training data. To test the ability of the model to generalise it must be applied to new data, that wasn't used for training.

A machine learning *model* consists of an *algorithm*, together with some parameters. The algorithm consists of a series of computational steps or instructions on how to manipulate the incoming data to achieve the chosen task. The parameters are values that are used by the algorithm at different stages (for example, an algorithmic step might say to multiply the input data by a scaling factor, where the particular value of the scaling factor is a parameter learned from the data during training). In ML applications, the algorithm is usually fixed while the parameters are learned from the data in a process called *training* (black arrow in Figure 3). It is important to understand that the performance of a model is function of both the algorithm and the parameters, and the parameters come from the data used during training.

Usually, an ML model is developed so that it can be applied to new data. If model performance (i.e. for the face detection classifier, the proportion of correct answers, called the *accuracy*) is evaluated on the training data it is usually much higher than the performance would be on new data. Because it is being asked about a photograph it has already seen in training, it can know the correct answer for this specific example directly, for example by memorising every photo in the training set, without having learned a general process to solve the task (i.e. recognising faces). The ability of a model to work successfully on new data is called *generalisation*. We want models that generalise to new data, and not models that just memorise the training data. The phenomena where models perform much better when evaluated on their training data is called *over-fitting*. To meaningfully evaluate model performance, it is very important that the data used to test performance is completely separate from the data used to train the model. When new data is not immediately available this is achieved with techniques like *cross-validation*. In cross-validation, the available data is randomly split into two parts, one part is used to train the model and the second part is used to test the model. This allows researchers to approximate the performance of the model on new data.

2.3 DEEP LEARNING

Deep learning refers to a particular family of machine learning algorithms, mostly developed for supervised learning problems. Deep learning models are *multi-layer artificial neural networks*. We will explain now what each of these terms means.

An artificial neural network (ANN) is a computer system which is loosely inspired by biological nervous systems. An ANN consists of many simple computational units, called neurons, each of which has several *inputs* and a single *output*. Each neuron performs its own independent and relatively simple computation: it adds up the inputs in a certain way and uses the result to set its own output. Before adding up the inputs it multiplies each of them with a *weight*, these weights are the parameters of the algorithm which are learned from the data. The pattern of how units are connected is called the *architecture* of the network. Mostly by trial and error, researchers have found that architectures which involve multiple "layers" of units are the most effective. There are different types of layers with different structures that can be stacked together in various combinations to build an effective ANN. We will see some examples of this in the next sections.

Usually, once the architecture is decided the performance of the network depends heavily on the parameters: the weights which each neuron uses when adding up its inputs. These parameters are

learned from data in the training step. This is done with a technique called *back-propagation*. To start with, random numbers are used for each weight. Then each training example is run through the network, and the final network output is compared to the desired true value from the labelled training data set. The difference between the true value and the network prediction is called the *error*. Starting from the output, the weights are then updated a small amount in a way that reduces the error in the output for that example. This is then repeated many times. Each training example leads to only a tiny change in the weights of the network, but over many repetitions of this process the changes build up. Often the error is visualised over the course of training, to understand more about the process. But crucially, the generalisation performance, which must be evaluated on separate data not used for training, is more important than the final error on the training examples.

Although each unit is relatively simple – taking multiple inputs and processing them with a simple function to obtain a single output – by combining these in huge numbers the resulting networks are very powerful. They also have very large numbers of parameters (there are many more connections between neurons than there are neurons, and each connection has a weight parameter). For example, as we will see below, Large Language Models (LLMs) are often characterised and described in terms of the number of parameters they have, which is measured in the billions, tens of billions or even hundreds of billions. To give a sense of scale, 100 billion seconds (roughly the number of parameters in OpenAI's ChatGPT) corresponds to 3000 years. Learning these parameters requires huge amounts of training data and is very computationally intensive. The key word here is *scale*. While the theory of ANNs was first developed in 1950s and 60s, it is the increased scale provided by technological developments in the 21st century that has unlocked successful practical applications. This is both because of the increasing speed of computers, but also the cost and availability of digital storage, and crucially the availability of huge quantities of data in digital form (for example digitised text from books, and images from digital cameras).

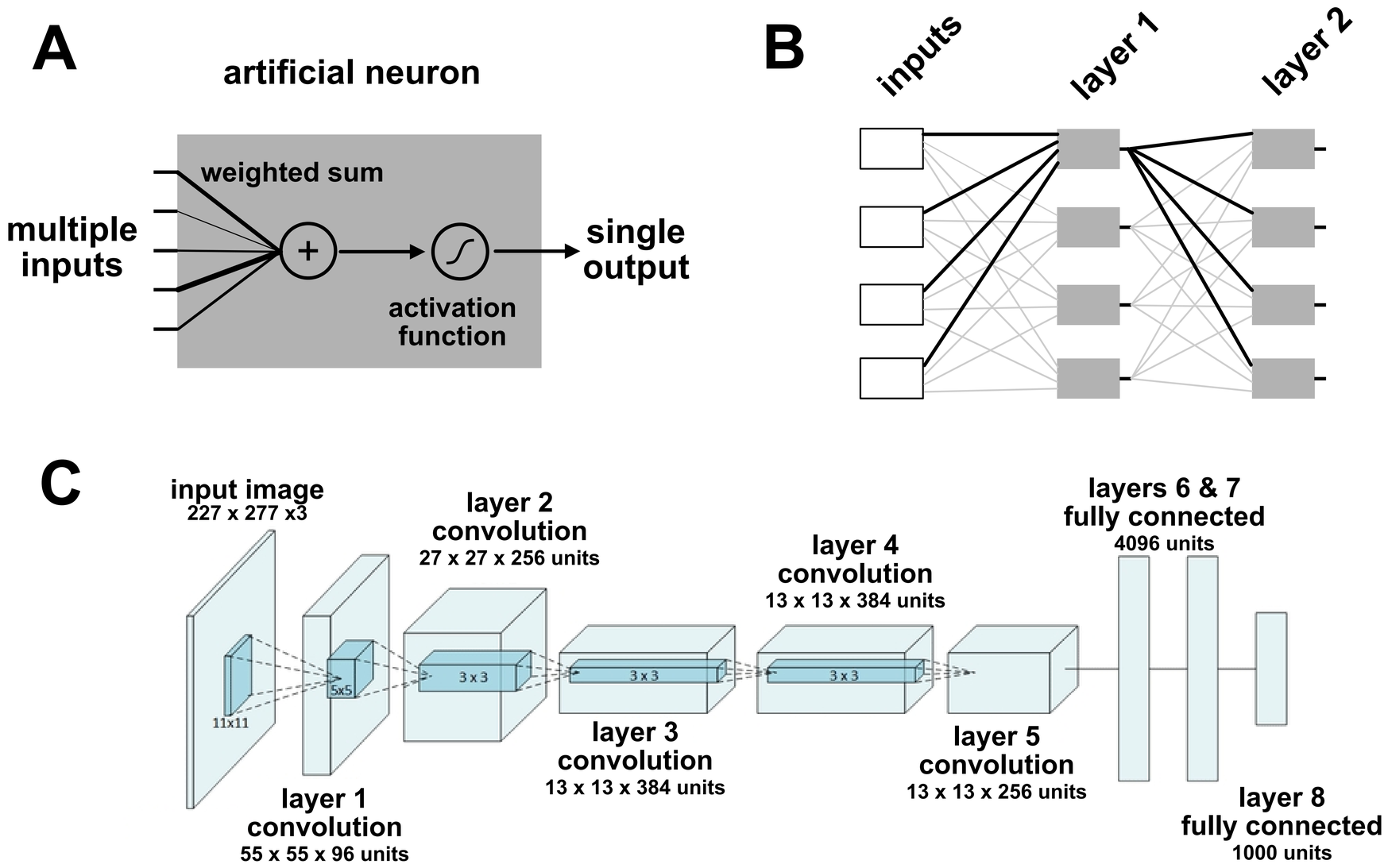


Figure 4: Artificial Neural Networks consist of many interconnected units or neurons, arranged in layers. **A:** Each neuron has many inputs from neurons in an earlier layer. These inputs are summed with the learned weights (indicated by lines of different thickness) and the sum is passed through a non-linear activation function to compute a single output value which is sent on to other neurons. **B:** Neurons are often arranged in layers, with different stereotyped structures. Here, two fully connected layers are shown. Fully connected means every neuron in layer 1 has a connection from every input, and every neuron in layer 2 has a connection from every neuron in layer 1. The connections of the first neuron in layer 1 are highlighted. **C:** A deep learning model typically consists of multiple layers stacked together. This panel shows the architecture of AlexNet, a network with 8 layers containing 650,000 neurons in total. It has 62 million learnable parameters (weights). AlexNet was winner, by some margin, of the 2012 ImageNet competition, which kickstarted a resurgence of interest in deep learning methods (see Section 3.2.2). Panel C modified from Hemmer et al. (2018) doi:10.3390/designs2040056

DEEP LEARNERS VS GAMERS, A FIGHT FOR RESOURCES

As we have seen, Deep Networks consist of huge numbers of relatively simple computational building blocks. This means, the output for each neuron can be computed somewhat independently of the others. Computer games, which render the visual image of a 3D scene in a game as the player moves around the game world, also have this form of *parallel* computational problem. For games, the appearance of each pixel on the screen can be calculated somewhat independently of the other pixels.

Driven by the increasing market for computer games, specialist hardware was developed to improve performance for these tasks, by companies like NVidia and AMD. Since these were developed for computer game graphics, they were called Graphics Processing Units or GPUs. However, because they are designed to run many simple calculations in parallel, they are also very efficient for training and running deep learning networks.

The rapid growth of machine learning and the popularity of deep networks caused huge demand for these devices, and suppliers could not keep up. Cryptocurrencies are another area that also provides strong demand for GPUs, as they can also perform the computational work needed to operate blockchains in an efficient way. This led to increase in prices and disruption to supply chains and meant many gamers were unable to keep up with the latest technology.

To address this and support their original core customers, some manufacturers sell different versions of their GPUs, trying to reduce how attractive they are for different areas. For example, some cards automatically slow down if they detect a lot of crypto-currency calculations, other cards have strict licensing terms forbidding their use in computational workstations without a display attached (so they can only be used for playing video games on a display and not for deep learning in a computer cluster).

These shortages have implications for those of us building for more charitable use cases where the commercial returns may not be as lucrative.

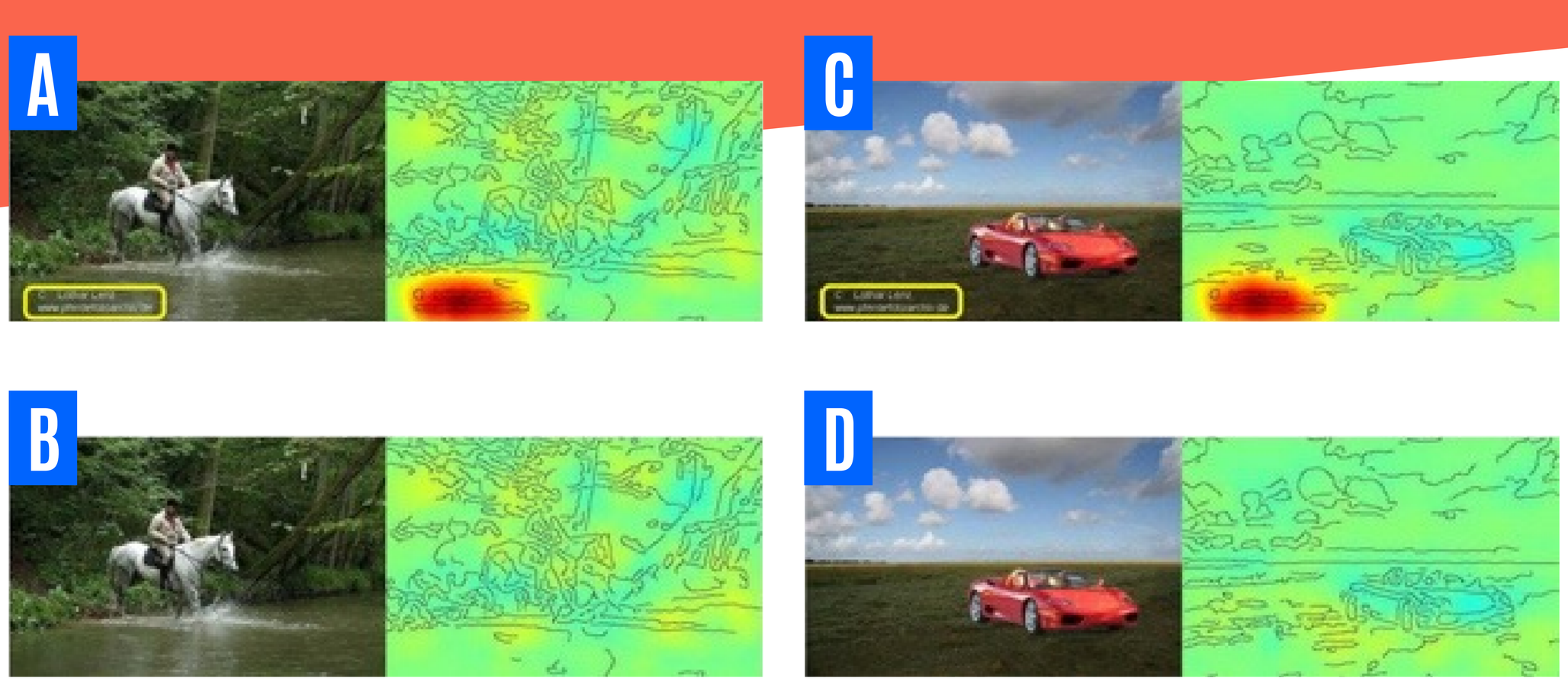
EXPLAINABLE AI

Deep learning models are often termed a *black box*. This is because it is very difficult to look inside them and understand how they work. This might sound strange when they are implemented in a computer, and we know every multiplication and addition that is performed.

But the enormous scale means this does not really help us understand what is going on. It is difficult to describe the computational steps in a way that gives understanding of what specific features of the input data are leading to the decision, and in what way. This can lead to bias or side-effects, when the model learns to perform the task in ways that are not expected.

One example of this is a deep learning model that was trained to identify pictures of horses. Because of a quirk of the training data set, most of the photos of horses were commercial pictures from horse riding events which carried watermarks. Photos of other categories in this data set tended not to have this. So, the network had not actually learned about horses as we recognise them, but instead was using an unexpected feature of the photographs that humans would typically ignore.

Explainable AI (XAI) is a subfield of AI in which researchers are trying to better characterise and understand how Deep Learning, and other AI models work. This considers not just accuracy, but also fairness. For example, when AI systems are making decisions with important real-life consequences it is important to understand how they work to ensure they are acting in an acceptable way. A system used in parole hearings to predict reoffending should not be making decisions based on any protected characteristics, such as race.



Images used as input for a model trained to detect horses, next to the corresponding relevance map which shows which parts of the image were most strongly influencing the models' responses.

A shows an image containing a copyright watermark, causing a strong response in the model. In **B**, the watermark has been edited out. The artificially created images **C** and **D** show a sports car on a lush green meadow with and without an added copyright watermark. In samples **A** and **C** the presence of class "horse" is detected, whereas in samples **B** and **D** this is not the case.

3. A brief history: from simple tasks to generative AI

3.1 TRACING THE PATH TO GENERATIVE AI: KEY DEVELOPMENTS IN ML

As deep learning models became larger and more powerful, they developed detailed internal *representations* of the type of data they were trained on. This means they can be used to generate new examples of that type of data. For example, creating images from scratch (e.g. text to image), or producing long segments of meaningful text to answer questions (e.g. chatbot assistants). The development of generative AI technologies has been particularly rapid in recent years. A key turning point for these systems was when the outputs started to become difficult to identify as AI generated. Today, it is not easy (and sometimes impossible) to recognise that outputs of a generative AI system are not real photographs taken by, digital art composed by, or text written by humans. Because the results are available for everyone to see and are relatively easy to interpret (compared to numerical measures like classification accuracy scores), generative AI has boosted excitement about and interest in the field.

In this section we will illustrate the historical development of these capabilities with some selective examples. Just as the parameters and the behaviour of AI models are closely tied to the datasets that they were trained on, so the history of AI progress can be traced through the development of certain landmark datasets for specific problems. Because of this link between model, training data and performance, shared data sets provided an important way for researchers working in the field to objectively compare different approaches – an open data set described a difficult but constrained problem. Organised competitions (with private test data withheld by the organisers to prevent cheating) allowed different groups to test their ideas, and this allowed the field to rapidly discover the most promising approaches. By reviewing some of these key landmark datasets and the developments they enabled, we aim to give a historical overview of the development of modern generative AI systems, focusing on the domains of images and language. As well as showing the history of the development of the algorithmic techniques, these datasets also show how the problems tackled changed from well-defined classification tasks (e.g. recognising hand-written digits), to modern open-ended generative AI (e.g. image generation). This is intended to be a high-level historical introduction rather than a comprehensive or complete history. For more detailed and comprehensive materials, see Resources.

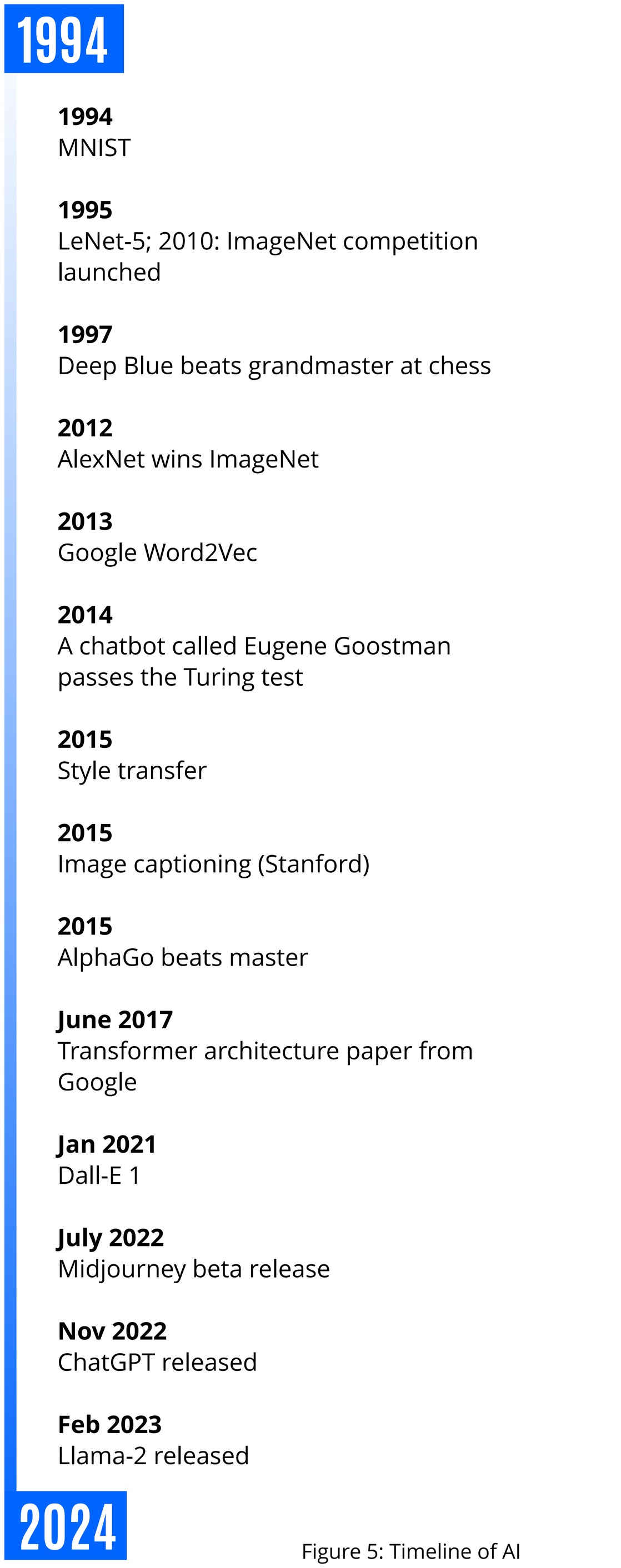


Figure 5: Timeline of AI developments discussed here.

3.2 EARLY IMAGE DATASETS

3.2.1 MNIST: RECOGNISING HANDWRITTEN DIGITS

Long before he became the head of AI research at Meta (formerly Facebook), Yann LeCun was working at NYU when, in 1994, he was part of a team that created one of the most influential datasets in the history of machine learning. The problem he was working on was identifying hand-written numerical digits, a subset of *optical character recognition* (OCR). This is an example of the kind of task that is easy for people to do (in most cases!) but was hard to program computers to do with a fixed set of rules, because of the range of different handwriting styles.

The MNIST dataset consists of 70,000 images of handwritten numbers (60,000 for training and 10,000 for testing). Each of these images has been labelled by humans annotating the digit displayed. While this was an interesting theoretical problem because of the constrained nature of the data (only 10 output classes), it was also an important practical problem for the automated sorting of mail. In the US postal system, every address has a numeric postal code.

Being able to automatically read the 5-digit handwritten postal code would allow machines to automatically sort mail, even if they couldn't read the rest of a handwritten address. Another application of digit recognition is automatic processing of handwritten cheques (widely used before the advent of credit cards and bank transfers).

Figure 6: Example images from the MNIST dataset. While these figures are easy for us to read, the natural variations in handwriting make it a difficult task for a rules-based computer vision system.

Source: https://en.wikipedia.org/wiki/MNIST_database

Today this is considered a relatively simple problem, which is often used as a tutorial exercise. Models can be trained to achieve excellent performance on this dataset with a modern laptop computer, with only a few lines of code using open-source deep learning software libraries. However, in the 1990's it provided a ground-breaking benchmark problem, pushing the limits of the computational resources of the time.

It was one of the first examples where multi-layer neural networks (aka Deep Learning) achieved state of the art performance. In 1995 Yann LeCun developed a model called *LeNet-5*. The performance of this model on the MNIST classification problem surpassed all other computer vision models of the time. LeNet-5 consisted of 7 layers. The first five layers alternated between convolution layers, which look for a fixed pattern anywhere in the image, and pooling layers, which combine the outputs of the previous layer in different parts of the image.

After this there was a *dense* or *fully-connected* layer. While convolution and pooling layers impose a particular structure in the form of which units are connected, in a dense layer every neuron is connected to every neuron in the layer before and in the layer after. This pattern of alternation of convolution and pooling, followed by dense connection was highly influential, and became a key motif that was widely used in many deep learning computer vision models (see Figure 7, AlexNet which we introduce next).

3.2.2 IMAGENET: RECOGNISING OBJECTS

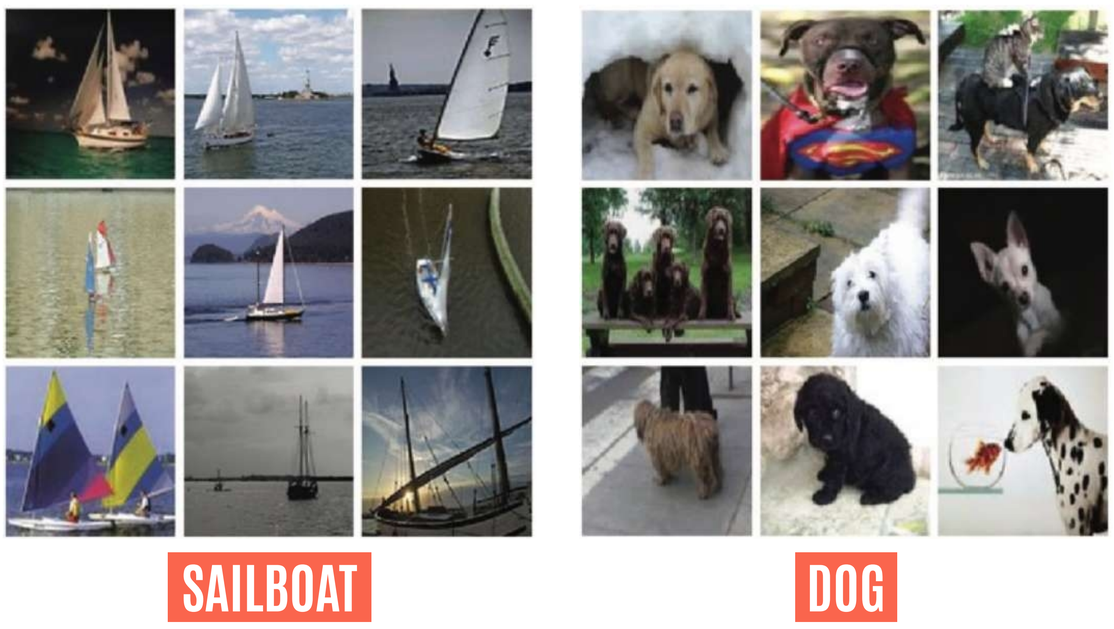


Figure 7: Example images for two different labels. These natural images have a broad range of variation in position, lighting and the specific example (i.e. different breeds of dog) compared to the more constrained digit images in MNIST. Example images are shown here for two categories, but ImageNet features over 20,000 categories in total.

Models trained on the MNIST dataset hinted at the growing superiority of multi-layer ANNs for computer vision, but it was a second landmark dataset released 15 years later in 2010 that accelerated this new paradigm – the ImageNet dataset. ImageNet was a project to map (again, with human labour) thousands of words to millions of images found on the internet. By combining these two aspects of human cognition – language and vision – at such a large scale, ImageNet was able to push AI research in a direction that was not previously possible. In 2010 a competition was launched to find the best model for categorising objects from images. The competition dataset included 1.3 million training images of objects from 1000 categories. This was a challenging task, even for humans, due to the diversity of images and categories. Humans have an error of around 5% (out of every 100 images they get around 5 wrong). Until 2012 the best performing models had error rates between 25 and 30%, some way below human performance. In 2012 a new model provided a step-change improvement, reducing the error rate to around 15%, nearly 10% better than the next model. This model was called AlexNet, developed by Alex Krizhevsky, a PhD student working with Geoff Hinton at the University of Toronto. AlexNet is a deep convolutional neural network, with a similar design to LeNet-5 described above, but with 8 layers rather than 7. While each layer was much larger (had more neurons), the model used the same motif of interlaced convolution and pooling layers, finished with dense interconnected layers (see Figure 4). AlexNet had 650,000 neurons, with 62 million learnable parameters in total. The success of AlexNet kickstarted a resurgence of interest in ANNs. Since 2012, every winner of the ImageNet competition has been an ANN model. In 2015 an ANN called ResNet with 152 layers won the challenge with an error rate of 4% - finally beating human performance on this task. ResNet-152 has a similar number of parameters as AlexNet – around 60 million – but requires approximately 10 times more computer time to train.

ImageNet, like MNIST, defines a supervised classification problem. However, while MNIST had just 10 possible output classes, with ImageNet that increased to 1000 possible categories. Another major difference between the data sets is the fact that ImageNet uses unconstrained natural photos. The images are not cropped and centred on the object of interest, and there can be multiple objects in each image (two boats, or a dog and a fish, in the examples shown in Figure 7). This seems like a trivial extension for humans doing the task, but presents real challenges for computer vision. The human visual system is well known to be robust to the sorts of natural variations we experience every day in aspects like brightness, lighting (morning vs evening), location of an object, and so on. For computer vision systems, which need to follow the fixed set of steps from an algorithm to get the answer from a given image, dealing with these variations is much more challenging.

Although the foundational ideas for the field of deep learning were proposed in the 1960s, the lack of early practical successes led to them falling out of favour. By the ‘90s only a few groups of researchers were working in the area, which was considered a niche theoretical topic. However, with the increasing availability of compute power and digital data from the turn of the century, deep learning went from strength to strength and is now the basis of most current generative AI. The dominance of AlexNet in 2012 can be seen as something of a turning point, which led to renewed interest in the deep learning approach, which was then widely adopted for many applications. In fact, after 2012 deep learning models quickly surpassed existing state-of-the-art approaches in a wide range of problems, for computer vision (image segmentation, self-driving cars), speech (e.g. automatic speech recognition), and as we will see next, language.

3.3 EARLY LANGUAGE METHODS

It's hard to pin-point exactly when people first had the idea of using machines to understand and produce natural language, but some of the most important early advances relevant to modern AI came from researchers and engineers working on telephone communication in the 1950s, in particular the work of Claude Shannon at Bell Labs. Shannon's idea was that an expression in a language could be understood as a chain of symbols, also known as *tokens*, taken from a fixed set, or vocabulary, which each have a certain chance of appearing at each point in the chain, depending on the tokens which appeared before them. A token could be a single letter, a whole word, or something in between.

At that time, and up until very recently, it was not possible to accurately calculate the probability of tokens appearing in sequences long enough to be useful for understanding human communication (e.g. whole sentences). Approaches based on this principle focused on smaller chains of tokens, known as n-grams. A 1-gram is a single token taken in isolation – for example if the tokens are words, then a 1-gram count is simply a count of the number of times each word appears in a piece of text. A 2-gram, or bi-gram, is a pair of tokens appearing next to each other in a chain – for example "hello there" and "car wash" are possible word 2-grams that might appear in an English text.

Because they were limited to short chains of tokens, approaches that used these ideas were not capable of understanding or producing text in the way that humans can. They focused instead on simpler tasks, such as classifying or extracting the general theme of documents or sections of text. The basic idea of these approaches is to consider each n-gram separately, and just count the number of times it occurs in a document, ignoring anything about where in the document occurs, or that it tends to be next to, before or after. When applied to 1-grams, such approaches are known as "bag-of-words" models (so-named because they do not capture any information about the order of words in the text, instead they are like throwing all the words from a document into a bag and shaking it up). These sorts of models could be used for tasks like sentiment analysis (counting how many positive words vs negative words were in a document), or topic modelling (finding common themes based on combinations of words that tend to appear together over a set of documents).

Mirroring the development of image models, multi-layer ANNs made for rapid advances in natural language processing, particularly through the possibility for *distributed representations* of words. Instead of each word or token being assigned a single numerical probability as in the bag-of-words models, ANNs like Google's Word2Vec in 2013 introduced models that map each individual word or token onto thousands of different numerical values, or vectors. These vectors, also known as *word embeddings* (or *token embeddings* if the input tokens are not whole words), served as a kind of "fingerprint" for each word, but more importantly they also displayed certain intuitive behaviour when combined, which showed they quantified something about what the words mean. For example, the vector for the word 'Paris' minus the vector for the word 'France' plus the vector for the word 'Italy' gives the vector for the word 'Rome'. That is, they represent something meaningful about the semantics of words.

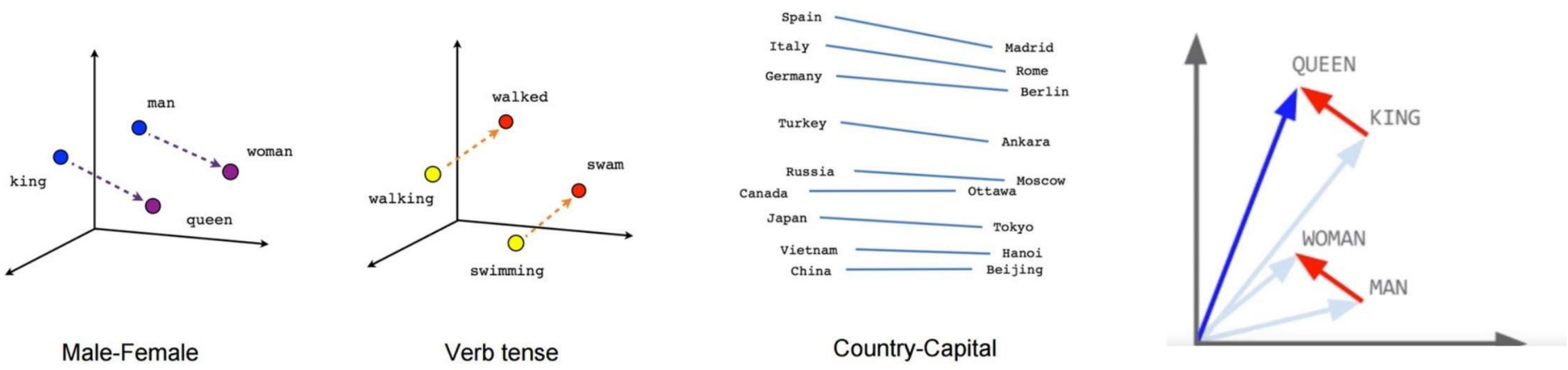


Figure 8: Schematic showing word embeddings for the words 'king', 'queen',' man' and 'woman'. Note that the relationship (i.e. the difference) between 'king' and 'queen' is similar (same length and same direction) as the difference between 'man' and 'woman' (red arrows). So 'king' + 'woman' – 'man' = 'queen'. The representation captures that the relationship between king:man is the same as the relationship between queen:woman. Only two dimensions are shown here in this cartoon schematic. Typical word2vec embeddings have hundreds of dimensions.

3.4 THE GENERATIVE AI REVOLUTION

In the previous section we considered early developments in the domains of images and language separately. For images, we saw how benchmark datasets expanded from the constrained and relatively well-defined problem of digit recognition with MNIST, to the more difficult and larger-scale problem of recognising everyday objects in natural scenes. NLP methods were also historically developed separately from the relatively specific field of computer vision. They developed from simple sentiment analysis based on bag-of-words (i.e. how many negative vs positive words are in a document), to capture meaningful semantic relationships in high-dimensional word embeddings like word2vec.

Here, we continue our brief history with examples of models that make the leap from supervised learning towards modern generative AI, as you will see from the open-ended output of the models. As we get to the more recent impressive applications of generative AI, the distinction between text and image data and models becomes blurred, as modern generative-AI models span these domains. Generative AI involves the development of such large models that they contain representations of a huge range of concepts we observe in the world. To make sense of this, the most advanced image models also use models and representations built from text, and the latest generation of large-language-models are inherently multimodal, able to process images as part of their input prompts, and produce images alongside text in their outputs.

3.4.1 STYLE TRANSFER: IMAGES AS INPUTS, IMAGES AS OUTPUT

One of the earliest examples of generative AI for images which gained wide attention is what was termed “style transfer”. By manipulating the activations of different layers of the network to two different images, researchers were able to create an output that combined them in a particular way. The subject or content of the output image came from one image, but the style (for example, the characteristic style of van Gogh’s oil paintings) came from the other. Contrast this with the ImageNet labelling task. Reporting the objects in each image is a simpler output, with a clear correct answer.

In style transfer, a new image is produced as the combination of a content image and a style image. The output is a new image, without a correct answer. There are many possible ways to recreate the first content image with the style of the second, and the evaluation of which might be better is a subjective judgement. This illustrates the transition to generative AI. The evaluation of style transfer images is subjective, and the output is something that it is not easy to recognise as “computer generated”.

Figure 9: Examples of Style Transfer. Panel A shows the source image: a photograph of a river scene. Panels B-F show the results of style transfer for different donor style images (inset). Reproduced from Gatys, Ecker & Bethge (2015) “A Neural Algorithm of Artistic Style” https://arxiv.org/abs/1508.06576

3.4.2 IMAGE CAPTIONING: IMAGES AS INPUT, SENTENCES AS OUTPUT

We have seen how ImageNet provided a dataset for evaluating visual object categorisation. This was extended to the more open-ended problem of image captioning. This goes beyond identifying the category of a single object in a photograph, and instead involves generating a semantic description of what is happening in the scene. This is another example of the shift to generative AI. While style-transfer produces new images as output, here we have written sentences as the output. Unlike the predefined categories of ImageNet, the output now is a full sentence, which can include any words in any combination to describe the image. In 2015 a landmark deep network model developed at Stanford by Fei-Fei Li, achieved another step-change in performance compared to previous efforts.

"CONSTRUCTION WORKER IN ORANGE SAFETY VEST IS WORKING ON ROAD."

"MAN IN BLACK SHIRT IS PLAYING GUITAR."

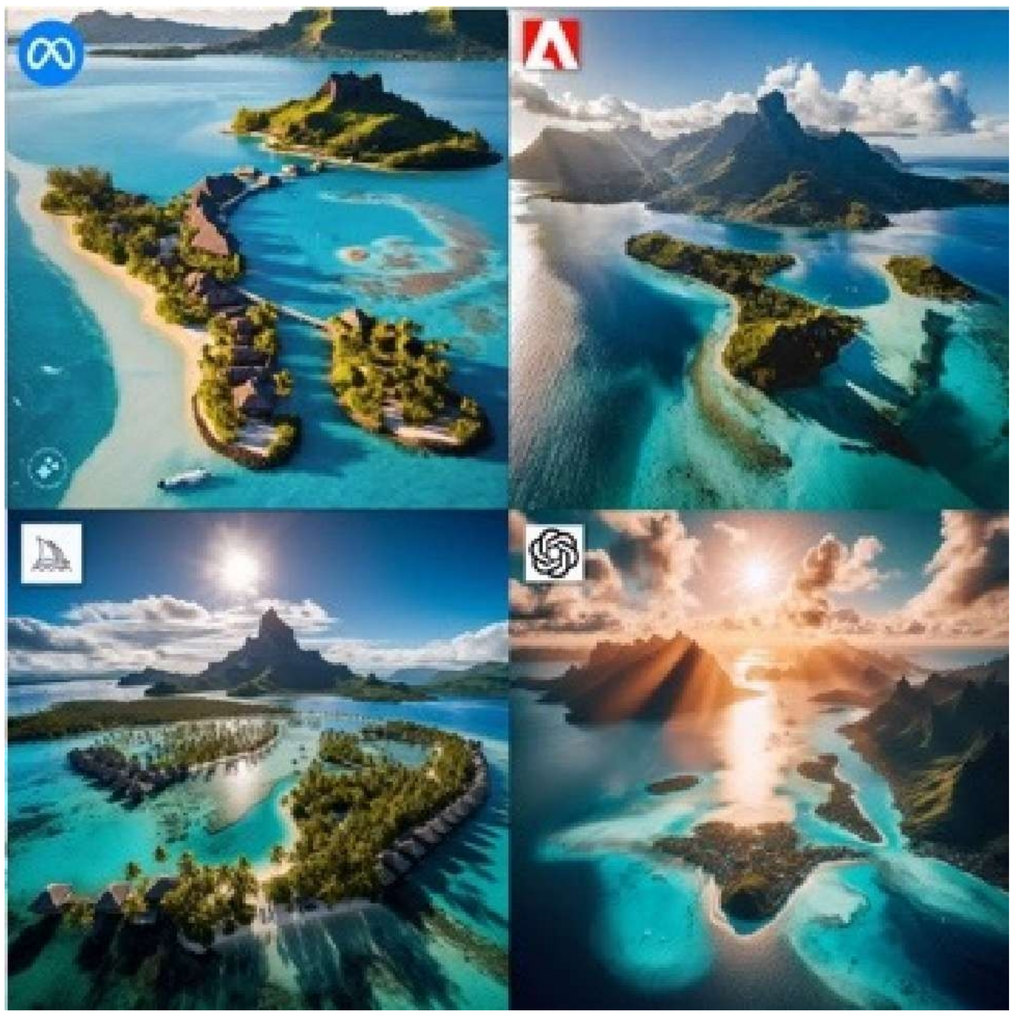
"TWO YOUNG GIRLS ARE PLAYING WITH LEGO TOY."

Figure 10: Examples of image captioning. Reproduced from Karpathy & Fei-Fei (2015) "Deep Visual-Semantic Alignments for Generating Image Descriptions" https://cs.stanford.edu/people/karpathy/deepimagesent/

3.4.3 TEXT-TO-IMAGE: TEXT AS INPUT, IMAGES AS OUTPUT

Current systems allow generation of full, photo-realistic images from text prompts. Systems providing this functionality include DALL-E, Midjourney and Adobe Firefly.

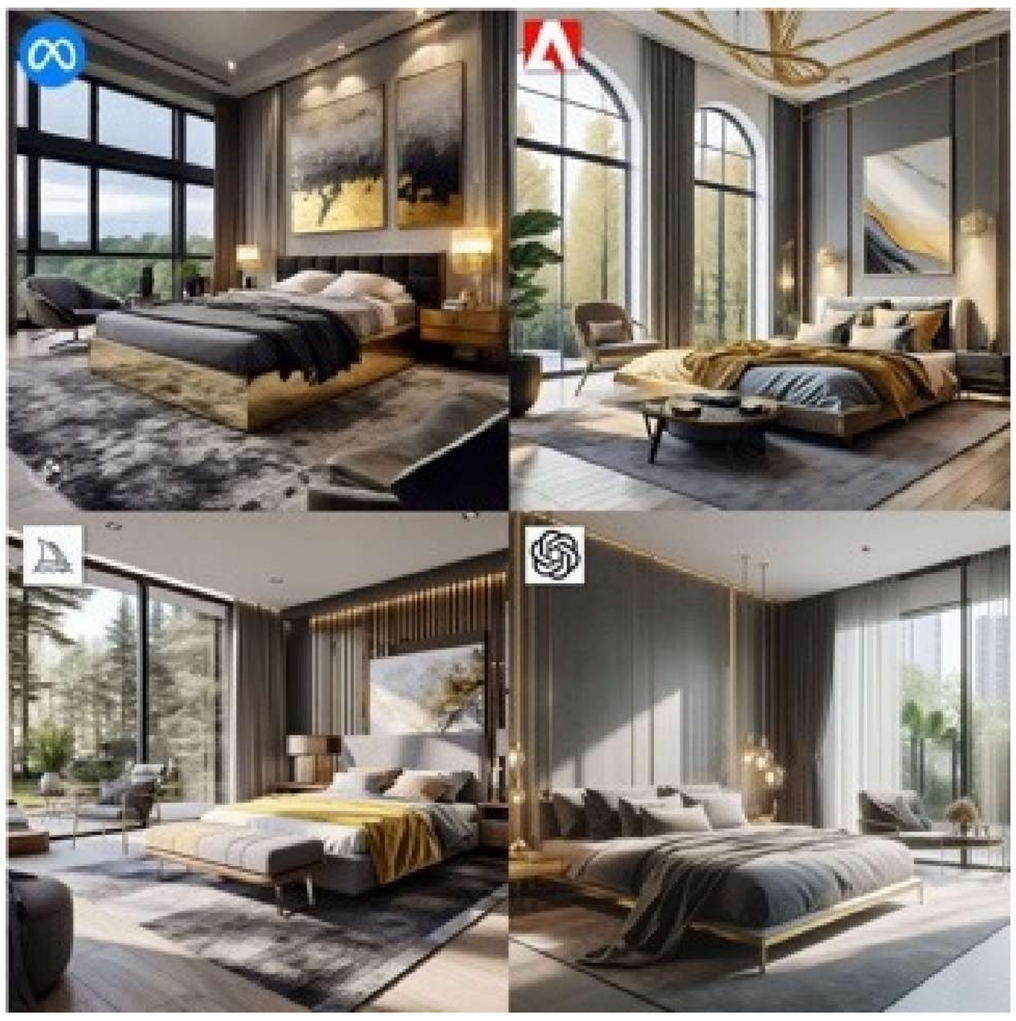
This allows the creation of fantastical images (for example, a snake made of corn, a cat riding a horse) which are impressively realistic. These systems have an encyclopaedic knowledge of the visual world and can produce realistic images of a huge range of concepts, in any visual style (photo-realistic, illustration, manga, line drawing, woodcut print etc.).



"An aerial drone shot of the breathtaking landscape of the Bora Bora islands, with sparkling waters under the sun"

"A macro wildlife photo of a green frog in a rainforest pond, highly detailed, eye-level shot"

"Simple flat vector illustration of a woman sitting at the desk with her laptop with a puppy, isolated on white background"



"A bedroom with large windows and modern furniture, grey and gold, luxurious, mid century modern style"

Figure 11: Example images generated from four different commercial models using the same prompt. The four models use for each example are: Meta AI, Adobe Firefly, Midjourney and OpenAI's DALL-E. Images generated by Chase Lean https://twitter.com/chaseleantj

3.4.4 AI ART: COMBINING IMAGE AND PROMPTS AS INPUT

There are image models that allow image generation with a text prompt, while also being constrained by an input image. This works best when the input image is high contrast (e.g. a black and white checkerboard).

Figure 12: An example of “AI Art”. These images were generated from a black and white image (checkerboard or spiral) together with the prompt “Medieval village scene with busy streets and castle in the distance”. Images generated by https://twitter.com/MrUgleh/

4. Large-Language-Models (LLMs)

An important class of model, called Large Language Models (LLMs) supercharged methods for language in recent years. This is the type of model behind chatbot AI assistants, like the famous ChatGPT from OpenAI.

The step-change in the ability of these models interpret and produce naturalistic language compared to what went before has driven much of the current AI excitement. Because of the importance of these types of models, here we provide a bit more detail about them. We aim to provide a high-level overview of the main stages involved in developing the models, and how they can be refined for different tasks.

A large language model (LLM) is a deep learning model. They were initially developed to work with text, although this boundary is dissolving as the latest models are genuinely multimodal, trained on both text and images. Here we focus on text based LLMs. As described above for supervised learning models, first an LLM must be trained, and following that it can be used. For LLMs the training process is composed of several stages, which we outline below. Each of these stages adjust the learnable parameters of the model.

Once trained, applying a model to generate text is called *inference*. This is analogous to the testing step in supervising learning. In a classification model, the testing step would typically involve evaluating the performance of the classifier on new data. Here as the task of generating new text is open ended, so as with the image generation models like style transfer, there is no single right answer.

4.1 INFERENCE

To use an LLM to generate text requires two components, the set of learned weights (measured in billions of parameters, usually 10's or 100's of billions), and some code which implements the architecture of the model to apply the weights to input data. In this case, because the outputs are so open ended, the input data are called prompts because of the way they invite the model to respond. Inference requires both high computational performance (i.e. GPUs) but also a lot of memory to hold the billions of weights of the model. Currently GPUs available with large amounts of memory to support LLM inference remain very expensive. There is also limited available because of high demand leading to a very competitive market with long wait times. The cost and availability of high-end GPUs is the main barrier to developing a new LLM.

OPEN VS CLOSED LLMS

Most inference is performed online in data centres (often termed *the cloud*) as a commercial service. In this case, the model architecture and the values of the weights can be kept private. Providers such as OpenAI and Midjourney work in this way. For example, users can pay a monthly subscription to OpenAI to access ChatGPT, or software developers can programmatically access these services from their own applications or websites, paying a small fee for each prompt sent. These are usually charged per-token, both for the prompt provided and the response generated.

However, there is also increasing interest in the ability to run inference on self-owned computer hardware – this is called *local models* or *local LLMs*, because the models are run close to the user (i.e. on their own computer), rather than further away from them (in the cloud). As described above, to run inference for a model requires access to the weights, as well as computer code to implement the architecture of the model. Some companies, such as Meta and French start-up Mistral, release these weights publicly for anyone to use. These models are called *open models*. However, even for open models typically neither access to the training data or full details of the training process are provided, and sometimes the weights are licensed in a way that restricts commercial use of the model.

4.2 TRAINING

The process of training a LLM is much more involved and computationally intensive than applying the model for inference. Here we outline the basic steps. It is useful to know these key stages to understand how models can be adapted to specific applications and contexts.

Current LLMs are based on an architecture called *transformers*. As with the word embedding models described earlier, this first involves calculating a high-dimensional numerical representation of each word or token, based on the words it occurs near to in the training corpus. However, in contrast to earlier word embedding models like word2vec, transformer models process a whole sequence of tokens at the same time: whether that is a sentence or even a longer paragraph. A key feature of this architecture is *attention*. This is a mechanism which adaptively selects the other words in the sequence that are likely to be most related to the currently considered token. This helps them to disambiguate semantic differences based on the context given by the sentence or paragraph, just as human readers do. There are three main steps involved in training a LLM as a chatbot assistant.

FOUNDATION MODELS: PRE-TRAINING

The first step in training an LLM is called pre-training. In fact, the P in ChatGPT comes from the term pre-training: GPT is short for Generative Pre-trained Transformer. We have covered what a generative model is, and the basics of the transformer architecture. Here we describe pre-training, the other crucial step.

Pre-training is a training procedure that runs iteratively using a huge dataset of text. For example, all the text on the internet. This is messy data, not curated or systematised in any way, just a huge amount of text. The training procedure is like supervised learning introduced previously. The training objective is to take a sequence of tokens (words) from the data set and predict the next token in the sequence. This is called *self-supervised* learning, because there are no external labels required (as for categorisation). The model can find the 'correct' answer itself from the data. The model supervises itself, hence self-supervised. This training procedure runs for a long time, resampling different fragments of text from the full training set, each time slightly updating the weights to get closer to the correct next word. Although this seems simple, it is extremely powerful. To predict the next word in a complicated paragraph, requires some representation of the key semantic concepts of the paragraph and how they relate to each other. In very large models, this simple objective can drive meaningful learning of complex semantic representations. In effect, this pre-training procedure compresses the training data. For Meta's Llama-2 model, which was one of the first widely used open LLMs, the pre-training step used around 10TB text from the internet. TB denotes a terabyte, which is 1024 GB. This is compressed by around a factor of 100 into the 70 billion numerical weights of the largest version of the model. This pre-training runs on clusters of thousands of GPUs for months, and costs millions of dollars.

After pre-training, a raw foundation model has learned a lot about the structure of the world represented in the training data and can do well at predicting sequences of text that look like documents from the internet. These generated documents might not be real: in effect the model "dreams" internet documents, producing documents according to the patterns it has learned, producing outputs that mimic examples from the training data. Such foundation models are not directly useful for end user interaction, because the outputs can often be boring, repetitive or irrelevant. However, the power of the representations they have learned can be harnessed with additional training steps.

CUSTOMISING THE MODEL: FINE-TUNING

To enable the model to answer questions in an interpretable and useful way, it is necessary to develop an assistant interface, often called a *chatbot*. This is an LLM that can give useful answers to the questions it is asked, interacting with the user in a productive and understandable way. This is achieved by subjecting a foundation model to a second training stage, called fine-tuning. Fine-tuning be used to teach the model a specific output style, to make a more specialised and useful model.

Rather than the huge dump of internet documents used in step one, here a completely different training set is used. This is much smaller, and highly curated, often created from scratch with carefully instructed human labour. For example, to create a chatbot, human workers write example questions, together with well researched and helpful answers. For example, the Llama model fine tuning used around 100,000 carefully curated assistant style question and answer pairs. The foundation model is then trained again on these documents, so that it learns how to produce output primarily in this style: the style of a helpful assistant. However, it can still use the semantic

knowledge and representations it learned from the internet data. Crucially, fine-tuning is much less computationally expensive that pre-training: obtaining the carefully curated training data can be the major expense in this step.

REINFORCEMENT LEARNING WITH HUMAN FEEDBACK (RLHF)

As noted above, obtaining or creating the carefully curated examples needed for fine-tuning can be difficult and time consuming, requiring a lot of human labour. On the other hand, comparing or ranking two answers generated by the model is a more passive task that can be done more quickly and easily by human raters. In this stage of tuning, multiple different answers are generated by the model. The models are inherently probabilistic, so when the inference procedure is repeated a different answer is generated, even if the prompt is the same. In a technique called Reinforcement Learning with Human Feedback (RLHF), humans rate multiple outputs to a given question, and select the best one. This is then fed back to update the weights of the model, as for the other training stages.

4.3 EXAMPLES

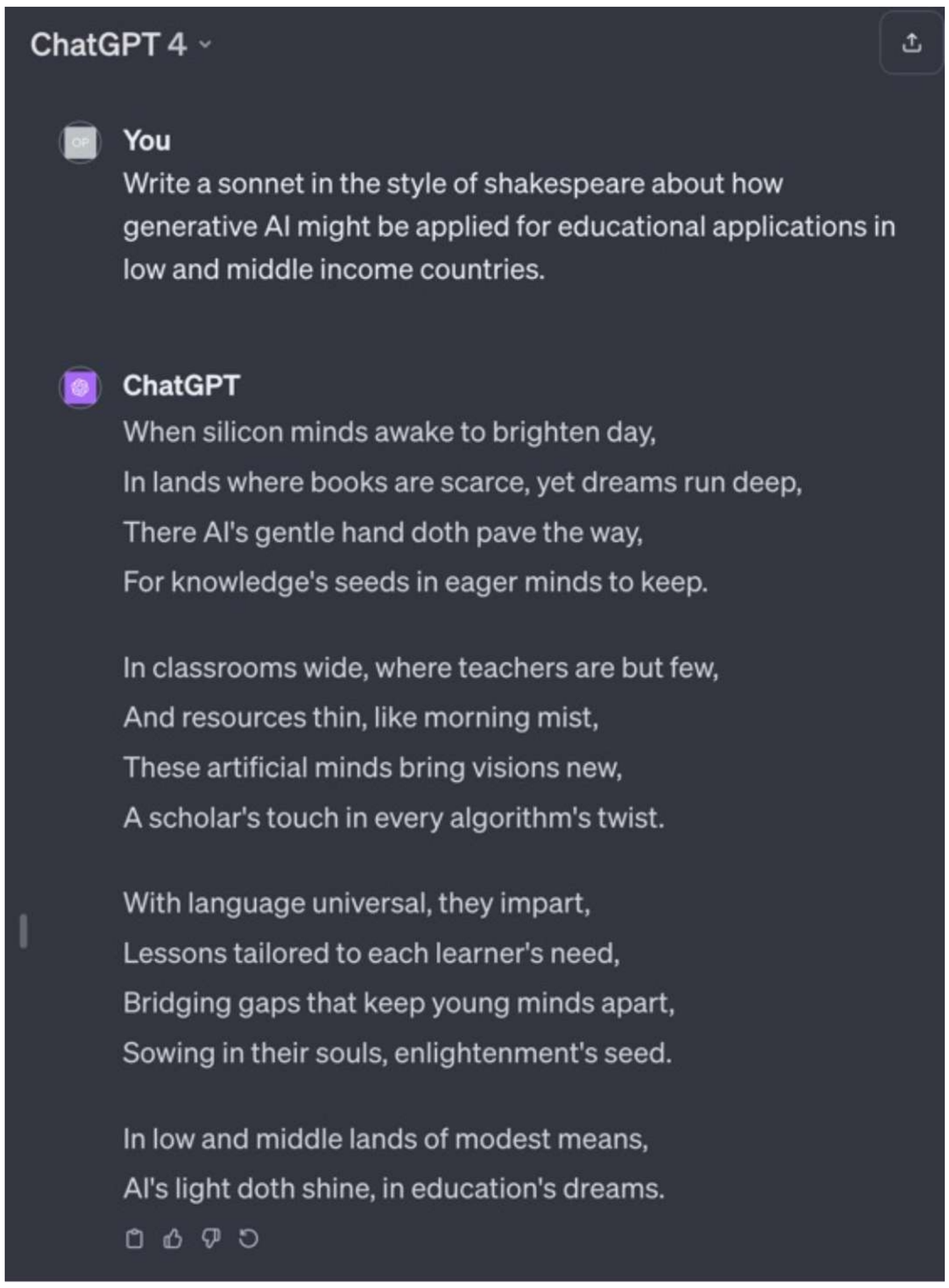


Figure 13: Some example outputs from OpenAI's ChatGPT.

4.4 EXTENDING AND IMPROVING LLMS

LLMs are instructed with natural language prose rather than computer programming code. Their behaviour can be tweaked from essay-length prompts which try to suggest a particular behaviour, for example responding in a certain style and focussing on certain topics. This opens a whole new paradigm for human computer interaction. The nascent field around this is called *prompt engineering*. Careful use of prompting can enhance the abilities of an LLM to solve problems. Even relatively simple additions to a prompt like "explain your answer step by step", can enhance the problem-solving capabilities of a chatbot.

In-context learning refers to the ability of LLMs to learn new tasks or solve problems based just on what is in the query context (the context is the combined inputs to the model, which can be a combination of the query, system prompt and chat history). For example, models can learn by example. If the prompt contains some examples of a specific task such as summarising text in a certain format, extracting dates from a paragraph, summarising the sentiment of a series of tweets, or some other task, the model can learn from this, as a person would. Often just a few examples are enough for the model to see how to perform that task on new data, and it doesn't require a full computational specification of the task.

Another key area by which the applications of LLMs are advancing is through combing a core LLM model with other components to form *compound AI systems*. One example of this is *tool use*, which provides additional functionality to a language model through functions that the model can call as part of its output. For example, OpenAI's ChatGPT has tools for browsing the internet and for executing Python code in an interpreter. ChatGPT knows these tools are available

and how to use them, which greatly enhances the types of questions it can answer. If a user asks ChatGPT for the time of high tide, it does not have access to this from the training data. Instead, it will use the browsing tool to perform a search using an internet search engine, scan some of the results, just as a human internet user would, and extract the information requested.

Another family of compound AI systems are termed *Retrieval Augmented Generation* (RAG). These systems give the LLM access to a prespecified set of data, for example a collection of documents on a special topic. In broad strokes, these systems work alongside the LLM by finding the parts of documents in the library which are most relevant to the user's query, and making sure those are provided in the context of the model so it can formulate accurate answers. This allows the LLM to function as a factually accurate natural language search engine for a specific data set.

4.5 CAVEATS

Some caution is required about applications of LLMs. No one really understands in detail how they work: they are in some sense closer to discovered artefacts than typically engineered systems where each step of operation is understood. Even the experts who built them don't fully understand how they produce any given response, and often their behaviour can be surprising and unexpected. One major problem is outputs that have been called *hallucinations* – while perfect language is produced, the facts or relationships reported are not always correct, and it is hard to identify this without existing expert knowledge. They can also exhibit signs of bias or negative stereotyping, and they can sometimes produce offensive or dangerous output. Most companies developing such models have dedicated teams thinking about these problems, often termed *Responsible AI*.

Most user-facing chatbot applications have what is called a *system prompt*, which is invisible to the end user, but is applied to every interaction with the model. This is a long natural language description of the type of response the model should and should not produce. For example, it might include the instruction "Don't swear in your response, no matter what the user requests. When quoting text from other sources remove any rude words". For the big public models there is a constant back and forth between curious users, some of which may be genuine bad actors, and the engineers working on the system. When users find a way to get the model to respond outside of the intended parameters, this is often called a *jailbreak*. Early jailbreaks used techniques to trick the model into producing illicit content such as instructions on how to perform an illegal activity. By asking for the response in a code block, or through increasing byzantine series of instructions such as asking the model to write a Python program that prints out the illicit content, some of the early safeguards put in place could be subverted.

The amazing performance of LLMs effectively renders computer representation of language a solved problem. This landmark has been achieved much sooner than many people expected, due to the unexpected and not yet fully understood behaviour that emerged from these large deep network models trained on next token prediction. It's likely that LLMs will expand into many areas of software, providing a natural language interface to complex functionality that would otherwise require detailed technical expertise. The field is now moving extremely fast, with significant improvements being made on a weekly basis.

5. Conclusion

Here we have provided a non-technical introduction to the field of generative AI. Our intention was to provide a high-level overview, which introduces and defines some key terminology, while also providing some historical context of how the field has developed. Historically, image models were developed separately from language models, but this distinction is disappearing as the next general of large models are multimodal, including text, image and even video. This historical overview is intended to be an introductory beginners guide rather than a comprehensive reference and so covers only a few examples.

This new generative AI technology is undoubtably going to change the world, potentially having a similarly large impact on society globally as other major technological shifts such as the printing press, personal computers, the internet, and mobile phones. The impact of AI technologies is also likely to be more rapid than previous shifts and might be measured in years rather than decades. Development in this area continues at a dizzying pace and it's likely that there will be dramatic improvements in model algorithms, hardware, training data, compound AI systems and practical applications in the coming months and years. Nonetheless, whatever the developments of the future we think it is important to understand some of the history that led us there.

5.1 RESOURCES

The Alignment Problem – Machine Learning and Human Value by Brian Christian (W.W. Norton & Company). This book gives an introduction to AI and covers in more detail some of the issues which broadly fall under the umbrella "Alignment".

How AI chatbots like ChatGPT or Bard work - A visual introduction to how LLM models work. (The Guardian)

Visual guide to the Transformer - A visual introduction to how LLM models work. (Financial Times)

A 1-hour talk introducing LLMs for a general audience, by Andrej Karpathy.

LIST OF TERMS

ARTIFICIAL INTELLIGENCE

Machine Learning
Deep Learning
Generative AI
Artificial General Intelligence
Explainable AI
AI Art

TURING TEST

REINFORCEMENT LEARNING

Reward
Exploration vs exploitation

SUPERVISED LEARNING

Classification
Regression

UNSUPERVISED LEARNING

Dimensionality Reduction
Clustering

OPTICAL CHARACTER RECOGNITION

ML MODEL

Algorithm
Parameters
Training
Accuracy
Generalisation
Over-fitting
Cross-validation

DEEP LEARNING

Artificial Neural Network
Weights
Layer
- Fully-connected / Dense
- Convolutional
- Pooling

Architecture
Error
Back-propagation

DISTRIBUTED REPRESENTATION

STYLE TRANSFER

IMAGE CAPTIONING

LARGE LANGUAGE MODELS

Token
Word embedding
Self-supervised
Open LLM
Local LLM
Inference
Cloud
Foundation model
Pre-training
Transformer
Fine-tuning
Reinforcement Learning via Human Feedback
Chatbot
Assistant
Hallucination
Biases
Jailbreak
In-context learning
Tool use
Retrieval Augmented Generation (RAG)

AUTOMATIC SPEECH RECOGNITION

SPEECH SYNTHESIS

MACHINE-READABLE

AI-for-Education.org