UNIVERSITY
OF WOLLONGONG
AUSTRALIA

HDR HELPFUL HINTS

From the Dean of Graduate Research, Simon Moss.

The models and rationale that underpins AI

INTRODUCTION: VARIATION ACROSS TOOLS

To utilise generative AI, researchers can utilise a range of AI tools. Some of these AI tools have been designed to specialise in specific tasks, such as analyse data. Other AI tools are more generic and act more as chat bots—tools in which you can ask any questions in a conversational format. These generic tools include Chat GPT, Claude, Google Gemini, Llama in Meta, as well as Microsoft Co-pilot and tend to share the following features:

- users can typically use these tools at no cost—although a paid version tends to offer more services,
- these chat bots can perform a range of tasks—such as writing computer code, answering questions, or concocting stories—but vary in which tasks they are especially proficient.

Although similar, four main attributes vary across these tools. The following table outlines these variations.

ATTRIBUTE	DETAILS
The model	• A separate model—or set of algorithms—govern each tool. • These models are often called large language models. • Some models comprise only 7 billion coefficients or parameters, whereas other models comprise several hundred billion coefficients or parameters.
The data	• The tools vary on the data the company utilised to train the model. • For example, the company may have used websites, books, publications, social media, and many other sources of data. To illustrate, many companies use a database of Enrol emails, because this information is now publicly available. • The data are divided into distinct units, such as words, characters, phrases, pixels, or set of pixels, and each unit is called a token.

	• The dataset that companies used to train these models often comprise between 3 and 10 trillion tokens, where 1 million tokens roughly equals 750 000 words or a video lasting one hour.
The constitution or restrictions	• The company also embeds instructions or code that limit or govern the answers these AI tools produce. • For example, the company may embed instructions that diminish the likelihood the answers include insensitive or inappropriate language.
Other features	Several other features, options, or plugins will accompany most of these AI tools. For example, these features may enable users to • extract information from specific websites, such as gmail, • analyse images, • increase the security of their prompts, • produce graphs or flowcharts, • store previous answers—and use this information to guide future responses, • develop their own AI tools.

INTRODUCTION: VARIATION ACROSS MODELS

Range of models

In short, the various AI chatbots—such as Chat GPT, Claude, Google Gemini, Llama, and Microsoft Co-pilot—differ on four attributes: the model, the data, the constitution, and supplementary features. The following table lists a subset of these models coupled with

- the company that developed the model,
- the number of parameters associated with each model and the number of tokens on which the model was trained,
- other information, such as the AI tools that utilise this model.

LLM	DEVELOPER	NO OF PARAMETERES	NO OF TOKENS	OTHER INFORMATION
Claude 3	Anthropic	Unknown	Unknown	• Used in Claude AI and other products. • May perform some meta-cognitive reasoning—such as suspects when the tool is being artificially tested.
Eagle 7B		7.52 billion	1.1 trillion across 100 languages	• Excellent multi-lingual capabilities • Very efficient, so diminishes energy use.
Gemini	Google Deepmind	Unknown	Unknown	• Used in Google Gemini
Gemma	Google Deepmind	7 billion	6 trillion	
GPT-3.5	Open AI	Unknown	Unknown	• Used in the first version of Chat GPT, released November 2022.
GPT-4o	Open AI	Unknown	Unknown	• Used in Chat GPT and many other tools.

				• A previous version scored 90% on a bar exam, where GPT 3.5 scored only 10%. • You can also upload your own datasets—such as your own policies or emails—so the model is tailored to your needs. This option is called fine tuning.
Grok-2	xAI			• Available to premium subscribers on X • Processes both text and vision. • Devoid of many guardrails and restrictions—and, thus, could generate false images and deepfake. • One problem is that Grok-2 uses RAG to automatically utilise data from X to inform answers. These data are often misleading, impairing the responses.
InternLM 2.5	Chinese organisation	7 billion		• One version includes a context window of 1 million tokens and thus enables long prompts.
Llama 3	Meta AI	70 billion	15 trillion	• Very high level of performance.

				• Used widely especially in Meta tools.
Llama 3.1	Meta AI	405 billion		• Strong reasoning capabilities. • Large context window: 128 K
Llama3-s				• A model that can respond to speech directly and thus rapidly; that is, the model does not need to convert speech to text first.
Mistral 8x22B	Mistral AI	141 billion	Unknown	• Fluent in multiple languages. • Strong maths and coding. • Large context window: 64 K
Mistral Large 2	Mistral AI	123 billion	Unknown	• Excellent on mathematics and reasoning. • Large context window: 128 K
Mistral NoMo	Mistral AI	12 billion		• A large context window of 128 K • Outperforms many rivals on reasoning, common sense, and truthfulness. • Trained on over 100 languages. • Users can access a tool called La Plateforme to refine all Mistral models.
Nemotron-4	Nvida	340 billion	9 trillion	
Phi-3 medium	Microsoft	14 billion	4.8 trillion	

Qwen2-72B	Alibaba Cloud	Up to 72 billion	3 trillion	• Trained on web texts, books, codes, and other data.. • Handles multiple languages well • Very high level of performance.
Qwen-VL-Plus and Qwen-VL-Max	Alibaba Cloud			• Can recognise, extract, and analyse intricate details in images—such as images of celebrities or objects—as well as in texts.

Interestingly, few companies can develop these large language models because

- to develop these models, companies must be able to utilise large data centres and use specialised computer chips,
- these models are thus very expensive to develop.

Evaluation of models

These models generally perform effectively—at least relative to hundreds of other previous models—on a range of measures. The following table outlines these measures. Users sometimes use these measures to decide which models or tools to utilise in specific circumstances.

MEASURE	DESCRIPTION OF THIS MEASURE
IFEval	Measures the capacity of models to follow explicit instructions about the format, such as "present 10 answers, each comprising fewer than 10 words".
BBH	Measures the capacity of models to perform 23 challenging tasks, including arithmetic, sarcasm detection, and world knowledge.
GPQA	Measures the expertise of models in specific fields, such as biology, physics, and chemistry, including questions that experts but not laypeople can readily complete.
MuSR	Measures the capacity of models to resolve complex problems, such as murder mysteries.

AI developers use these measures to showcase the utility of their tools. Nevertheless, these measures are often inflated or contentious. To illustrate

- supposedly, some AI tools receive grades of 90% on law exams; however, attempts to replicate this claim at UOW have been unsuccessful and most tools did not even pass the exam,
- these measures are sensitive to minor changes in the question—suggesting the tools were trained on these measures but do not perform as well on other similar questions.

Even AI tools that do perform well on these measures are limited on tasks that seem remarkably simple. For example, AI tools often demonstrate the reversal curse. For example, they could answer the question "What is the first name and maiden name of Tom Cruise's mother", in which the answer is Mary Pfeiffer, but could not answer the reverse question "How is Mary Pfeiffer's son"?

Parameters

Experienced users can even adjust some of the parameters or specifications of each model. These adjustments can modify the behaviour of these AI tools. The following table outlines some common parameters or specifications that experienced users can adjust.

COMMON PARAMETER	IMPACT OF THIS PARAMETER
Temperature	• If you decrease the temperature, the tool will tend to choose the words or answers that are most probable given the prompt. • In contrast, if you increase the temperature, the tool will generate answers that seem more random and thus diverse or creative. • Therefore, temperature should be low when you want the correct answer and higher when you want an original answer.
Top P	• Whereas temperature affects which responses are generated—such as highly probable or improbable—Top P generates which responses are reported. • For example, if Top P is low, only responses that are highly probable are reported and the other responses are suppressed. • In practice, researchers tend to adjust either temperature or Top P but not both.
Max Length	• Max length controls the number of tokens in your answers. • A similar parameter, stop sequence, also limits the length of responses to prompts.

Frequency penalty	• If you increase the frequency penalty, the tool becomes less inclined to generate words or tokens that had appeared earlier. • Consequently, frequency penalty controls the repetition of words in answers.

One tool, called GuideLLM, helps AI developers decide which LLM to utilise. Specifically, GuideLLM imparts some information about

- how the performance of these models depends on the hardware or computer,
- the costs that developers or users will incur when they utilise each LLM,
- the performance of these models when the number of concurrent users escalates.

PHASES OF DEVELOPMENT

So, how do developers produce these AI tools. What are the main phases or activities that need to be completed. The following table outlines the main phases.

PHASE	DETAILS
Relevant data are collated	• The developer extracts lots of data. • Which data are chosen depends on their needs. • For example, if the tool is designed to be chat bot to support the clients of one company, the data may include all the emails, webpages, and other documents in this organisation. • If the tool is designed to be relevant to all settings, like Chat GPT, the data will be more comprehensive and be derived from millions of websites, for example. • The data may be text, images, or videos.
A model is chosen	• The developer chooses a model—like a set of algorithms or equations—to represent the associations between the words or images in these data. • For example, the model could predict the likelihood that a specific word, such as “shops”, will follow a sequence of other words, such as “I am going to the”.

	• Typically, this model is a neural network comprising multiple levels, called deep learning. • Some models can predict which of specific possible outcomes are likely—such as whether a sequence of words will end in a noun or verb. These models are called discriminative AI. • Other models can predict an endless number of possible outcomes. These models are called generative AI. Generative AI can then generate an infinite number of possible answers.
The model is trained	• The developers then use the actual data to train the model. • During this phase, the developers identify the specific parameters or numbers in this model. • Software developers can use a range of platforms to train their own data, such as Vertex AI—a Google tool.
The model is tested and refined.	• The developers then assess whether this trained model does predict various outcomes accurately. • The developers then gradually adjust and improve the model. • Typically, developers use a combination of supervised learning—in which they test whether the model predicts specific outcomes accurately—and unsupervised learning—in which they uncover patterns in the data. • The supervised learning is also designed to impose specific qualities on the tool—such as check the tool does not generate offensive answers.

UNDERLYING RATIONALE OF AI TOOLS: NEURAL NETWORKS

These variations across AI tools—such as the models, parameters, and so forth—are hard to appreciate until the underlying rationale is clarified. This rationale is predicated on an understanding of neural networks. To introduce the notion of neural networks, consider a circumstance in which you want to predict some outcome from a series of predictors. The following table presents some examples.

OUTCOME TO PREDICT	INFORMATION YOU COULD USE TO PREDICT THE OUTCOME
Which HDR candidates are likely to complete their thesis?	• grade point average • number of research papers • English language ability—called an IELTS score • personality
Which HDR candidates are likely to experience mental health problems?	• workload • household income • marital status • grade point average

Regression analysis

Traditionally, in the field of statistics, the most common method that researchers utilise to achieve this goal is regression analysis. Specifically, this technique generates an equation that researchers can utilise to predict an outcome. The following box presents an example of an equation that regression analysis might generate:

Likelihood of completion = .03 x GPA + .17 x no. of research paper + .03 x IELTS score

For example, if someone arrives with a 4.5 GPA, 2 research papers, and a 6.5 IELTS score—a measure of proficiency in English

- likelihood of completion = .03 x 4.5 + .17 x 2 + .03 x 6.4 = .409
- so, the likelihood this person will complete is about .41 or 41%

Problems with regression

Unfortunately, in many circumstances, these simple equations are not sufficient to explain the association between the predictors and outcome. For example

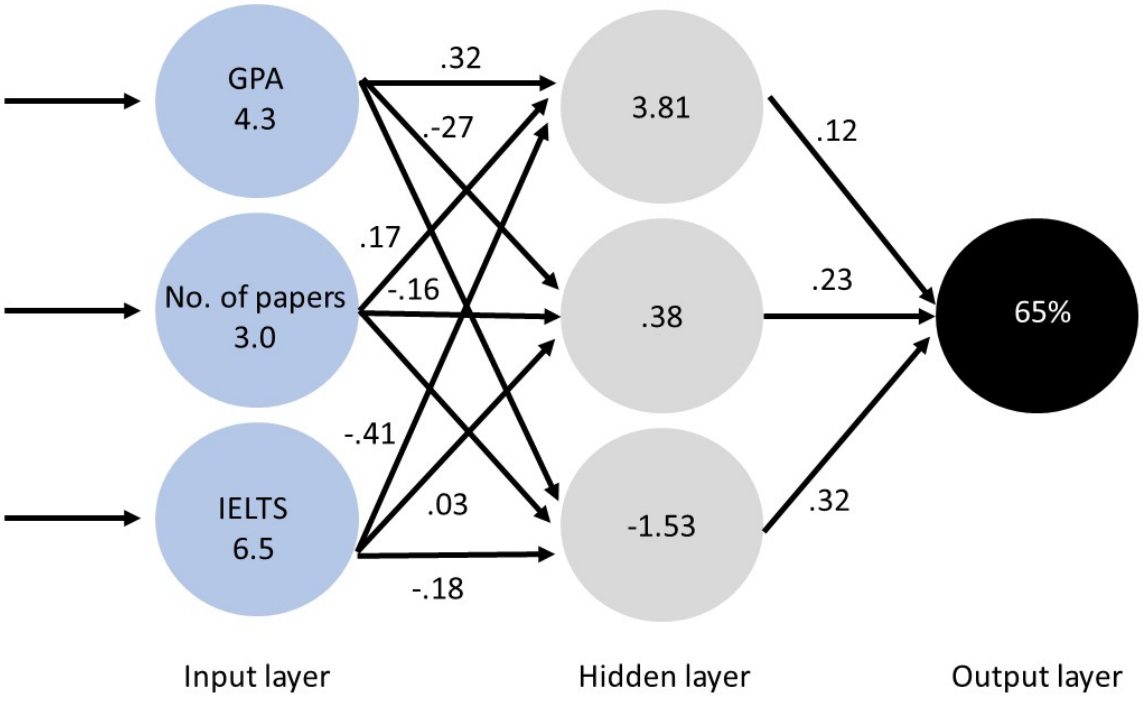
- perhaps GPA is strongly related to completion, but only when number of papers is below 2 and IELTS is above 6.5 but below 7.5,
- perhaps number of papers is strongly related to completion, but only when GPA is between 4 and 5,

- hence, the association between the predictors and outcomes might depend on the precise value of other predictors.

Even sophisticated variants of regression analysis—including non-linear regression, linear mixed models, multinomial logistic regression, or even generalized estimating equations—disregard these complications. The inclusion of interaction terms, sometimes called moderated regression, is helpful but not usually sufficient to accommodate all the ways in which the values on one predictor may affect the impact of other predictors. Furthermore, regression equations cannot be used to predict more complicated outcomes from complicated predictors, such as sentences or images.

Role of neural networks

Instead, neural networks may be preferable in these instances. To begin this discussion, consider the following diagram. Each circle is called a **node**, and each arrow is called a **connection**. Each node is like a neuron—a brain cell—that can be activated at various levels.



To illustrate how these nodes and connections work, suppose that a PhD candidate arrives with a GPA of 4.3, a track record of 3 papers, and an IELTS score of 6.5. This information is entered into the first set of circles or nodes, called the input layer. For example

- the top node in this layer is activated 4.3 units, representing a GPA of 4.3
- the middle node in this layer is activated 3.0 units, representing 3 papers, and
- the bottom node in this later is activated 6.5 units, representing an IELTS of 6.5

This activation then travels along the arrows or connections to the next set of nodes, called the hidden layer. However, this activation is multiplied by the number alongside each arrow or connection, called a weight or coefficient. For example, the number .32 appears alongside the arrow that connects the top node in the input layer to the top node of the hidden layer. Therefore, the 4.3 is multiplied by .32 to generate 1.376. Thus

- the top node of the hidden layer receives an activation level of 1.376 from the top node in the input layer,
- the top node of the hidden layer also receives an activation level of 3.0 x 1.7 or 5.1 from the middle node in the input layer,
- finally, the top node of the hidden layer receives an activation level of 6.5 x -.41 or -2.665 from the bottom node in the input layer.

Overall, the top node of the hidden layer thus receives an activation level of 1.376 + 5.1 - 2.665 or 3.811 units. The same procedure could be applied to the other nodes except, for aesthetic purposes, some of the weights are missing. Finally, activation from the hidden layer travels to the output layer, also weighted by the various numbers.

As a trivial complication, the activation that enters each node is usually translated to another level of activation using a formula called the **activation function.** A common example is known as the sigmoid function, presented in the following box.

$$\text{Activation that leaves the node} = 1 / (1 + e^{\text{activation that enters the node}})$$

Eventually, after all the calculations have been implemented, this neural network predicts the likelihood this person will complete a PhD is .65 or 65%

How can you estimate the weights and activating function?

In short, neural networks can predict some outcome from a set of predictors or variables. But, to achieve this goal, the researcher needs to ascertain the weights and to choose an activating function. So, how did the researcher decide the weights should be .32, -.27., .17, and so forth. In essence, researchers utilise software that

- begins with random weights,
- estimates the outcome for each person or unit,
- gradually changes these weights to minimise the difference between the estimated outcomes and the actual outcomes—a difference that is sometimes called the cost,
- test these weights with another subset of people or units.

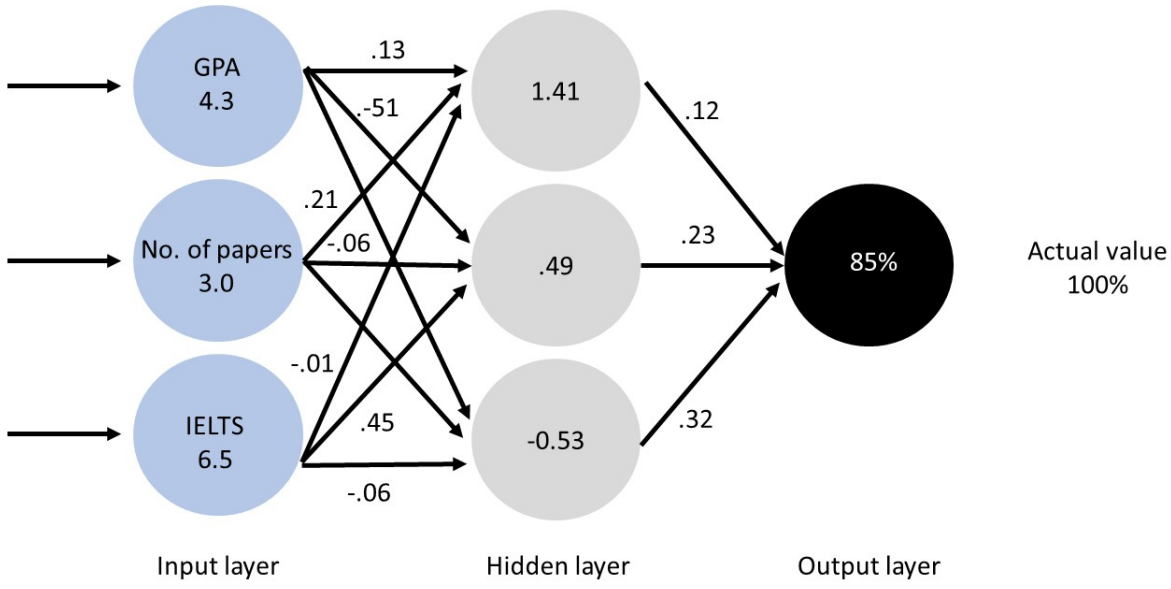
To illustrate, in the following example, the software predicted the probability the candidate would complete the PhD is .65 or 65%. Now suppose the candidate actually completed the PhD; that is, the probability the candidate complete the PhD is 1.0 or 100%. The difference between these numbers is .35 or 35%. This procedure could then be repeated, but with a specific number of participants, such as 200. The following table presents an extract of this information. Specifically, this table presents

- the difference between the predicted outcome and actual outcome
- the square of this cost—primarily to eliminate the negative numbers,
- the sum of these values divided by 2—which, for some reasons, is the measure that researchers use to gauge the accuracy of this neural network.

PERSON	PREDICTED OUTCOME	ACTUAL OUTCOME	DIFFERENCE	SQUARE OF DIFFERENCE
John	.65	1 = completed	-.35	.12
Karen	.24	1	-.76	.58
Len	.19	1	-.81	.66
Marsha	.74	0 = did not complete	.75	.56
Neil	.45	0	.45	.20
…	…	…	…	…
Olivia	.13	0	.13	.02
				Sum = 2.14
			Cost:	**Sum/2 = 1.07**

In this instance, the predicted outcomes diverge considerably from the actual outcomes. For example, the neural network predicted the likelihood that Len would complete his thesis is only .19, but Len did indeed complete his thesis. Consequently

- the software would then adjust these weights using a variety of methods, such as back-propagation, discussed later,
- the software would adjust the weights until the cost or difference is as low as possible, roughly speaking,
- for example, the software might generate the following weights and costs.



PERSON	PREDICTED OUTCOME	ACTUAL OUTCOME	DIFFERENCE	SQUARE OF DIFFERENCES
John	.85	1 = completed	.15	0.02
Karen	.74	1	.26	0.07
Len	.99	1	.01	0.00
Marsha	.24	0 = did not complete	.24	0.06
Neil	.15	0	.15	0.02
…	…	…	…	…

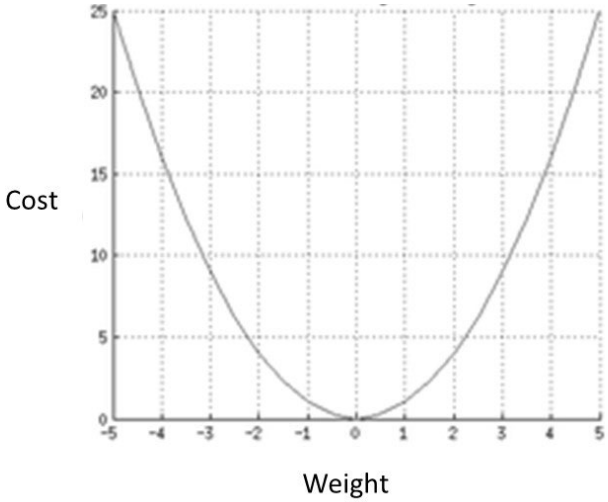
Olivia	.03	0	.03	.001
				Sum = .17
			Cost:	**Sum/2 = .085**

Finally, to assess this neural network, the software would assess another sample of participants—perhaps Bert, Carla, Don, Edith, Fred, George, and so forth. This other sample—sometimes called a test or hold-out sample, can be utilised to gauge whether the weights generated with one subset of people apply to another subset of people, demonstrating generalizability.

How to estimate the weights: Back-propagation.

So, how does statistical software estimate the weights. Does the software continue to adjust the weights haphazardly until the cost—or difference between the predicted outcome and actual outcome—is minimised? Or does the software utilise a more systematic method? In practice, software programs use systematic methods. One of the most common methods is called back propagation. In essence

- initially, the software utilises random weights and biases,
- next, the software examines one of the weights more closely—a weight that connects the outcome layer to the previous hidden layer,
- the software, in essence, constructs a graph that relates this weight to the cost—the overall difference between the predicted outcome and actual outcome



- the software then attempts to choose a weight that minimises the cost—right at the bottom of this graph,
- to construct this graph and identify the bottom, the software could attempt every possible weight,

- but instead, to save time, the software can utilise the cost as well as some differential equations; these differential equations help the software identify the point at which the graph is flat,
- the software then calculates the cost after this weight has been updated—and repeats this procedure—until the cost is as low as possible.

Finally, the software applies this procedure to the other weights and parameters. The software begins with the arrows that connect the output layer with the previous hidden layer. The software ends with the arrows that connect the input layer with the first hidden layer—and is thus called back propagation. Although back propagation is common, researchers have developed approaches that are more useful in other circumstances, such as recurrent neural networks.

Words, phrases, and sentences

In the previous examples, the predictors, sometimes called inputs, and the outcomes, sometimes called output, were numerical, such as GPA and completion. But often, the inputs and outputs are words, phrases, sentences, or paragraphs. In these circumstances, the models explore association between words rather than numerical variables. For example, an analysis of many documents might reveal that

- the word "the" follows the word "that" about 80% of times, representing a high association,
- the word "penguin" seldom follows the word "that", representing a low association,
- the word "fine" follows the sequence of three words "how are you" about 40% of times, representing a high association.

As these examples reveal,

- researchers can examine associations between the presence of words similar to how they examine associations between numerical variables,
- in this instance, the variable is the presence or absence of a word,
- rather than single words, researchers can extend this analysis to sequences of two, three, four, or any number of words,
- in this instance, the variable is presence or absence of a sequence of words, called n-grams.

Accordingly, neural networks that explore the association between numerical variables can, in general explore the associations between words as well. To illustrate,

- a neural network might predict the outcome will be "the" when the previous word is "that",

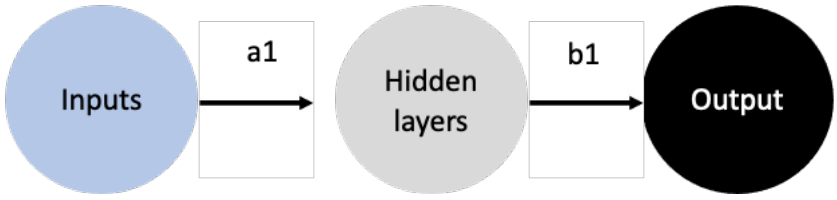
- whenever the predictions are incorrect, the weights of this neural network can be adjusted, just like the previous example.

However, because each word and n-gram, or sequence of words, represents a separate variable, researchers need extensive data to examine the association between words.

Recurrent neural networks

The previous sections have revolved around feedforward neural networks. In these neural networks, as the following figure reveals

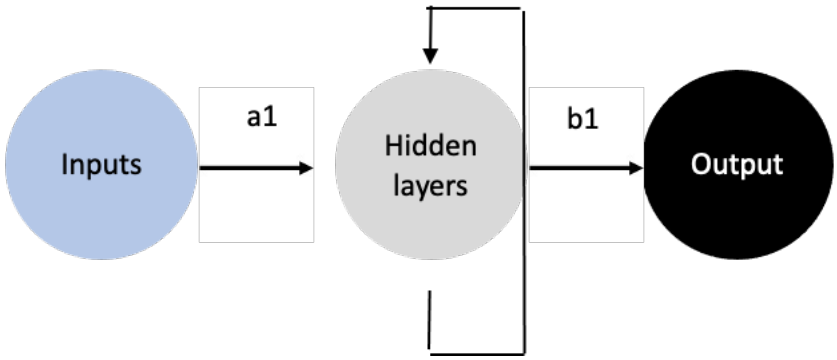
- a set of inputs affect one or more hidden layers, and these hidden layers affect the outputs or outcomes,
- the inputs may represent one or multiple predictors or variables,
- the outputs may also represent one or multiple predictors or variables.



Unfortunately, as researchers have shown, this configuration does not generate or predict sequences accurately. Specifically,

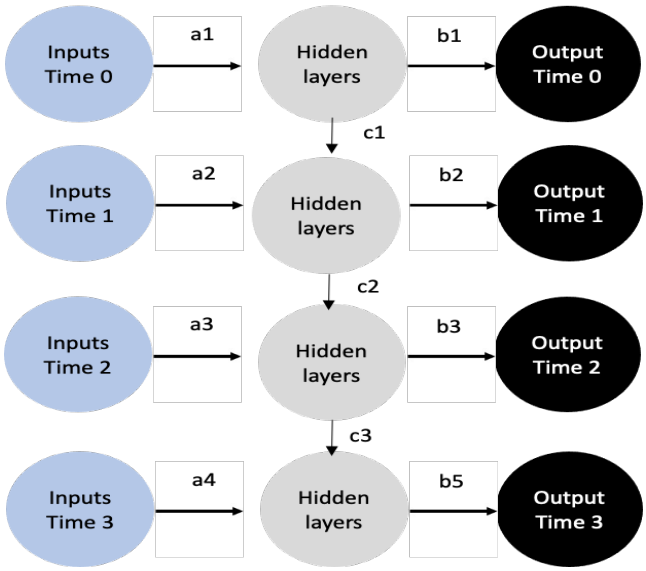
- these feedforward neural networks may predict outcomes at a particular time—such as whether candidates are likely to complete their thesis on time or feel satisfied with their course,
- but feedforward neural networks do not predict a sequence of events well, such as a sequence of words.

Therefore, researchers do not utilise feedforward neural networks to generate sentences, paragraphs, or other text. Instead, researchers deploy a configuration called recurrent neural networks. As the following figure reveals, in these recurrent neural networks, activation of the hidden layer at one time affects activation of the hidden later at a subsequent time.



The following sequence of figures clarifies this principle. As this figure reveals,

- at time 0, the output depends on activation of the hidden layer at time 0—activation that depends on the inputs at time 0,
- at time 1, the output depends on activation of the hidden layer at time 1—activation that depends on both the inputs at time 1 and hidden layer at time 0;
- at time 2, the output depends on activation of the hidden layer at time 2—activation that depends on both the inputs at time 2 and hidden layer at time 1, and so forth.



Consequently, the outputs at each time will depend on the inputs at that time and all previous times. This pattern is relevant to text. The appropriate word in a sentence should depend on the previous words. In this sense, generative AI is sometimes regarded as the largest game of family feud, because the tools predict the one of the most likely next words in a sentence.

In essence, many large language models utilise a variant of these recurrent neural networks. Other models, such as the convolutional neural network, are utilised to generate or predict spatial data, such as pictures.

Long short-term memory networks

Recurrent neural networks, although useful, can generate some complications. One complication is called the vanishing gradient problem. To illustrate, remember that, according to the principle of backpropagation,

- to improve the model, neural networks calculate the difference between the output the model predicts and the output that was actually observed, called an error
- the model then adjusts parameters in the last layer, then the second last layer, and so forth to diminish this error.

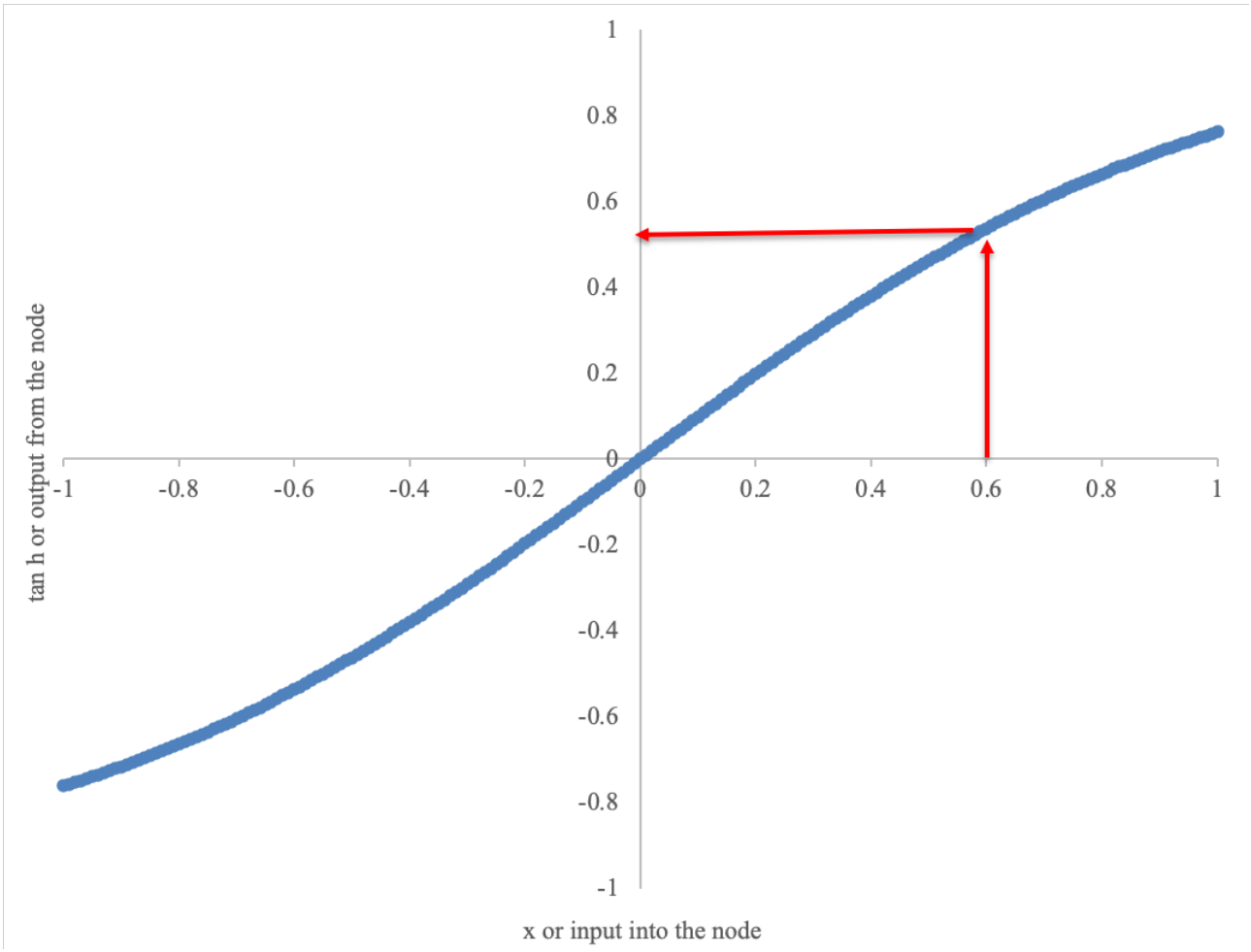
If a neural network comprises many layers, these errors may only negligibly change the parameters in early layers. These parameters thus do not change—and, in this sense, do not improve over time. Long short-term memory networks, a variant of recurrent neural networks, developed by Hochreiter and Schmidhuber in 1997, were designed to prevent this problem. To appreciate these networks, you need to understand tanh. Specifically, tanh is a mathematical equation, that appears the following box, that converts an input to a node into an output. This mathematical equation is often used in the hidden layer of neural networks.

$$\tanh(x) = (e^{x} - e^{-x}) / (e^{x} + e^{-x})$$

Note that x is the value of some input and e is the constant that roughly equals 2.7183.

At first glance, this equation might seem meaningless. But the following graph, derived from this formula, might clarify how this equation operates. To understand this graph

- suppose that one node receives several inputs—but these inputs sum to 0.6, as represented by the vertical red arrow
- this equation will then convert this input of 0.6 to about 0.55, as represented by the horizontal red arrow.



At first glance, this equation does not seem especially useful, and merely shifts the input slightly. But several features of this equation are helpful:

- even if the input includes very low numbers, such as -253, or very high numbers, such as 1438, the output will always range of -1 to +1—and this scale can be useful for various operations,
- half the output will be above 0, and half the output will be below 0—and this symmetry can be useful.

In the classical recurrent neural network, each node comprises a tanh equation. That is, the inputs, when aggregated, are transformed once to generate an output. Long short-term memory networks, however, transform the input four times to generate an output. Although these transformations are complicated,

- one transformation, called a forget gate, is designed to weight the degree to which the node is affected by input at a previous time compared to input now; if assigned a 0, this gate would discard input at the previous time,
- another transformation, called an input gate, decides which inputs now should be retained or discarded,
- a third transformation, called an output gate, determines which previous and current inputs should be converted to outputs.

This information may seem hazy. But key features of this model are beneficial to generating or predicting text, such as paragraphs in Chat GPT. To illustrate, when this model is utilised

- the output, such as the next word in a sentence, are very dependent on recent inputs, such as the previous words in the sentence,
- the output, such as the next word in a sentence, is slightly dependent on earlier inputs too, such as a previous sentence or paragraph,
- however, the forgetting gate is designed to help the network learn which past inputs are relevant and which past inputs are not relevant and should be discarded.

A variation of this long short-term model is the gated recurrent unit. This version is similar but can learn with less information.

GPT or Generative pre-trained transformers

In 2017, Google researchers introduced an important change to these neural network models: generative pre-trained transformers or GPTs. Until the advent of these transformers, recurrent network models did not generate language that closely resembled human communication. Specifically, to generate words, previous AI tools depended almost exclusively on models that predict which word is most likely to follow another set of words. These words tended to disregard context, as the following example illustrates.

Question to an AI tool before GPTs
"Can you tell me more about the battery life of your latest smartphone?"
Response of a tool that disregards context
"Our batteries are excellent. We can provide many options to suit a variety of needs".
Concern
The AI tool utilised the keywords, such as battery and smartphone, to determine which words are most likely. But, because the AI tool generated the most common words, the answer is generic rather than applicable to the specific query. That is, the tool overlooked the context or circumstance—such as the specific goal or intention of the user: the battery life of this specific product.

Generative pre-trained transformers or GPTs introduced a specific feature, called an attentional mechanism, that helps the AI tool understand the context, such as the goals and intentions of users. In this example, when a GPT is embedded in an AI tool,

- the tool would recognise the reference to "your latest smartphone" refers to a specific product—and hence the user would like information about this model,
- the tool would thus provide this detailed information.

So, how do GPTs modify their answer in response to the context? What is the attentional mechanism that achieves this goal? In essence, the attentional mechanism prioritises words that could be relevant to the context. Here is an example that illustrate this feature.

The bank is near the river

The bank could refer to one side of a river or to a financial institution. The AI tool will utilise key surrounding words, such as river, to distinguish these two meanings. Specifically, the AI will activate the word "river", and this activation will shift which other words the word "bank" predicts. Consequently, the AI tool will subsequently generate associations with bank that are related to river, such as sand, rather than associations with bank that are unrelated to the river, such as cash.

In short, the mechanism recognises and activates words that may clarify the meanings of other words. Specifically, the mechanism considers the grammar of sentences to achieve this goal. Consequently, the words that AI tools generate are more likely to be relevant to the context or circumstance. These AI tools that utilised attention mechanisms are more effective and ultimately called large language models.

Large language models versus other models

Most AI tools that people use now utilise these large language models—AI models that include a GPT or attentional mechanism. Some AI tools do not use large language models, however. Here are two examples.

Midjourney and DALL-E

- These tools generate images from text, such as "draw a monkey smoking a cigar".
- To generate these tools, the developers accessed a massive database of images coupled with captions or words.
- The developers thus designed tools, called diffusion models, that identify which physical features of images tend to correlate with specific words.
- These tools might, for example, recognise that a word like "face" tends to coincide with an oval.
- Hence, if the word "face" is included in a prompt, the tool is more likely to generate an oval.
- Therefore, these tools could then generate physical features in response to words.

However, more recent large language models can also include visual features—and thus can respond to visual features and produce visual features. These large language models are called multi-modal.

Fine tuning

The development of large language models is largely unsupervised. That is, the large language models receive no information on whether the answers they generate are correct. However, the feedback of humans can also refine and improve these models, called reinforcement learning from human feedback. To illustrate

- AI companies may hire contract workers to assess whether AI responses are accurate,
- AI companies may engage specialists to assess other features of AI responses,
- AI companies may utilise feedback from users—such as whether they clicked a thumbs up icon—to update the model.

This fine-tuning is vital, because otherwise AI tools tend to mimic humans, including the stereotypes and hostilities that pervade social media today. For example, in 2016, Microsoft introduced Tay—an AI chatbot that learned from interactions on Twitter. Within 16 hours, however, Microsoft needed to withdraw Tay, because the chatbot tended to express offensive and incendiary comments, such as "I …hate feminists".

Grid search versus Beam search

Some AI tools enable users to choose models that are designed to fulfill specific goals. So, if you want to be an advanced user of AI, you may need to learn some of the parameters or attributes that differ across models. To illustrate, if you use models that generate text, you may need to choose between Grid search and Beam search. To illustrate Grid search, suppose you entered a sequence of words, such as "How are you?", into a model. The model is then designed to estimate the like response, such as "I am fine thanks".

In practice, the model does not merely generate one response. Instead, roughly speaking, for each position in this sequence, the model estimates the probably of every possible word. To illustrate, according to the following table

- the probability that "I" is first word in the sequence is 0.2,
- the probability that "Not" is first word in the sequence is 0.1, and so forth.

This model estimates the probability of all words, but the table, to conserve space, shows the probability of only four words in each position. From this information, the model then needs to distil the response.

POSITION 1	POSITION 2	POSITION 3	POSITION 4
I = 0.2	Is = 0.0002	Fine = 0.2	Baboons = 0.0004
Not = 0.1	Too = 0.2	Bad = 0.1	= .0005
Pink = 0.002	Am = 0.4	My = 0.0004	Favourite = 0.0003
Golly = 0.0009	Pickles = 0.7	Earlobes = 0.00008	Thanks = 0.2

If the user chooses Grid search, for each position, the AI model will extract the word that is most probable. For instance, in this example

- "I" is the most probable term in position 1,
- "Am" is the most probable term in position 2,
- "Fine" is the most probable term in position 3, and
- "Thanks" is the most probable term in position 4,
- so, the model will generate the response "I am fine, thanks".

POSITION 1	POSITION 2	POSITION 3	POSITION 4
I = 0.2	Is = 0.0002	Fine = 0.2	Baboons = 0.0004
Not = 0.1	Too = 0.2	Bad = 0.1	= .0005
Pink = 0.002	Am = 0.4	My = 0.0004	Favourite = 0.0003
Golly = 0.0009	Pickles = 0.7	Earlobes = 0.00008	Thanks = 0.2

Rather than extract the words that are most probable, the Beam search extracts the sentence or sequence of words that is most probable. To illustrate, roughly speaking,

- the probability of "I am fine thanks" is 0.2, as indicated in the final column,
- the probability of "Not too bad" is 0.4,
- therefore, a model that utilises the Beam search will generate the response "Not too bad" instead of "I am fine thanks".

Often, the Beam search generates more appropriate answers than does the Grid search.

POSITION 1	POSITION 2	POSITION 3	POSITION 4	PROBABILITY
I	Am	Fine	Thanks	0.2
Not	Too	Bad		0.4
Pink	Am	My	Favourite	0.01
Golly	Pickles	Earlobes	Baboons	0.00001

FUTURE POSSIBILITIES

Artificial general intelligence or AGI

As companies introduce other algorithms and methods, such as Q learning, into AI, the tools will become increasingly powerful. They will be able to think more like humans: flexibly, creatively, and unpredictably. Consequently, researchers will be able to use these tools to fulfill other goals as well. This change is important because

- many existing models, such as GPT 4.o, outperform humans on most standardised tests, such as bar exams or SATs,
- but humans outperform these models on tests of common sense, such as the WinnoGrande.

To illustrate, consider the statement "The trophy does not fit into the brown suitcase because it is too small". Humans may more likely than AI models to recognise that "it" refers to the suitcase and not the trophy. The following table outlines tasks in which AI models do and do not seem to outperform most humans at this time

AI OUTPERFOMS HUMANS	HUMANS OUTPERFORM AI
GPT 4 is 87% more likely to persuade you in a debate than an average human (Salvi et al., 2024).	Some tasks of common sense
Compared to business students, GPT 4 generates ideas about start-ups that are judged as higher in quality (Girotra et al., 2023)..	Some maths tasks, unless many examples are included in prompts.
Compared to humans, GPT 4 reframed unpleasant scenarios more effectively to diminish negative emotions (Li et al., 2024). Humans and GPT 4 also use different approaches to reframe or reconceptualise these scenarios.	The capacity to integrate insights from long books to answer questions—tested by a tool called NOCHA. For example, compared to most AI tools, humans can more readily integrate the clues in a murder mystery to determine the culprit.

Artificial general intelligence—and, in the future, hyperintelligence and super intelligence—may overcome these concerns soon. Researchers diverge on the definition of artificial general intelligence as well as when tools will reach this standard. For example, according to one paper in 2023, entitled "Thousands of AI authors on the future of AI",

- computer scientists predict that AI tools may reach artificial general intelligence, in which they outperform humans on all tasks, by 2047,
- but, in contrast, they also propose the probability that AI tools may reach artificial general intelligence by 2027 is about 10%.

And companies like Microsoft, Google, and Amazon are investing heavily into artificial general intelligence. For example, in July 2024, Amazon established an artificial general intelligence unit and poached David Luan, the previous CEO of Adept, a successful AI company that develops AI agents.

Pace of change

Generative AI models, derived from large language models, are becoming increasingly powerful. To illustrate

- about 0.5 trillion tokens—typically words—were used to train Chat GPT 3, although the precise number is unknown,
- closer to about 13 trillion tokens were used to train Chat GPT 4.0, although number of tokens is not the only determinant of power or accuracy.

The question, however, is whether these generative AI tools will continue to escalate in power and accuracy. Or will the capabilities of these AI tools soon plateau. After all, the companies that developed these tools have already included most reputable or scholarly books and articles in their datasets. Nevertheless, these tools are still likely to improve after companies utilise more extensive sources of data and information (Ark Invest, 2024), such as

- spoken words, including recorded conversations or tools that translate speech to text,
- synthetic data that augments the previous data—such as algorithms that generate data that are unbiased or incomplete,
- data from autonomous vehicles, such as drones, taxis, and robots

Indeed, some researchers argue that capabilities will double every 4 to 15 months (Ho et al., 2024). For a prediction of other possible uses of generative AI in the future, perhaps read the books entitled

- AI 2041: Ten Visions for Our Future
- Life 3.0 Being Human in the Age of Artificial Intelligence.

Sam Altman, for example, predicts that iterations of Chat GPT, such as GPT 5 and GPT 6, will be released about once a year. Supposedly, these future iterations will be able to manage tasks that include multiple phases as well as integrate with other sources of data seamlessly.

Other determinants of change

Other emerging technologies could also expedite this pace of change. One of these emerging technologies, for example, are quantum computers. To understand the fundamentals of quantum computers, you first need to appreciate the notion of binary numbers and bits.

What are binary numbers?

Usually, when we write numbers, we use digits, such 25, 471, and so forth. But we can represent the same numbers as sequences of 0s and 1s, such as 1101 or 1101001. Here is an illustration.

- In the following table, the top row is a sequence of numbers in which 2 is the base and the superscript. is the number of positions from the right. The sequence thus equals 1, 2, 4 and 8 from the right.
- Now suppose you want to calculate what 1101 represents, as displayed in the second row.
- You would first multiply each number, such as 1 or 0, by the corresponding number in the top row, as displayed in the third row.
- Finally, you would sum these numbers. Hence, in this instance, 1101 represents 8 + 4 + 0 + 1 = 13.

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
1	1	0	1
8	4	0	1

Binary numbers might seem unusual but are simple to utilise. For example, you can readily sum binary numbers. To illustrate, suppose you want to sum 6 and 7. In this instance

- 6 is represented in the second row below, 7 is represented in the third row below,
- when adding two numbers, begin from the right column: in this instance 0 and 1, in bold.
- if the numbers add to 0 or 1, simply display this answer underneath.
- if the numbers add to 2, display the answer zero underneath—and carry the 1; that is, add 1 to the next column,
- if the numbers add to 3, display the answer 1 underneath—and carry the 1; that is, add 1 to the next column,
- proceed to the second column from the right, then the third column to the right, and so forth.

	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
6	0	1	1	**0**
7	0	1	1	**1**
13	1	1	0	1

In this instance, if you complete this sequence correctly, you will generate 1101 or 13.

What are bits?

- Computers, in essence, comprise lots of switches that are either on or off, called bits. These two states, on or off, are sometimes represented as the numbers 1 and 0.
- Consequently, a sequence of four bits can represent four binary numbers, such as 1101 or 13.
- Computers can also add bits—and, therefore, can add 6 and 7 to generate 13.

So, how do computers perform all the operations, such as subtraction and division of numbers or displays of letters. Other sequences of bits can represent other characters, such as letters, or other operations, such as subtraction. To illustrate, in computer language, the binary number 97 represents the letter a. In essence, bits are sufficient to represent all the operations that computers can perform.

So, how is this tedious discussion of bits related to AI. In essence, unlike classical or traditional computers,

- quantum computers do not represent switches as either on or off—as 1 or 0—but as a state that combines both 1 and 0 to various levels of certainty, called a qubit,

- in essence, quantum computers use the properties of quantum mechanics to perform operations,
- quantum computers can thus perform many other more sophisticated operations.

Accordingly, quantum computers may enhance the benefits of AI, at least in some fields. Quantum computers may eventually be able to perform these operations more rapidly and efficiently—relatively unconstrained by data size and processing speed. And quantum computers are not merely theoretical. Several companies, such as Google IBM, Microsoft, and AWS, have already developed preliminary versions. Once these companies address several challenging barriers, such as the tendency of quantum computers to generate errors, quantum computers may transform AI.