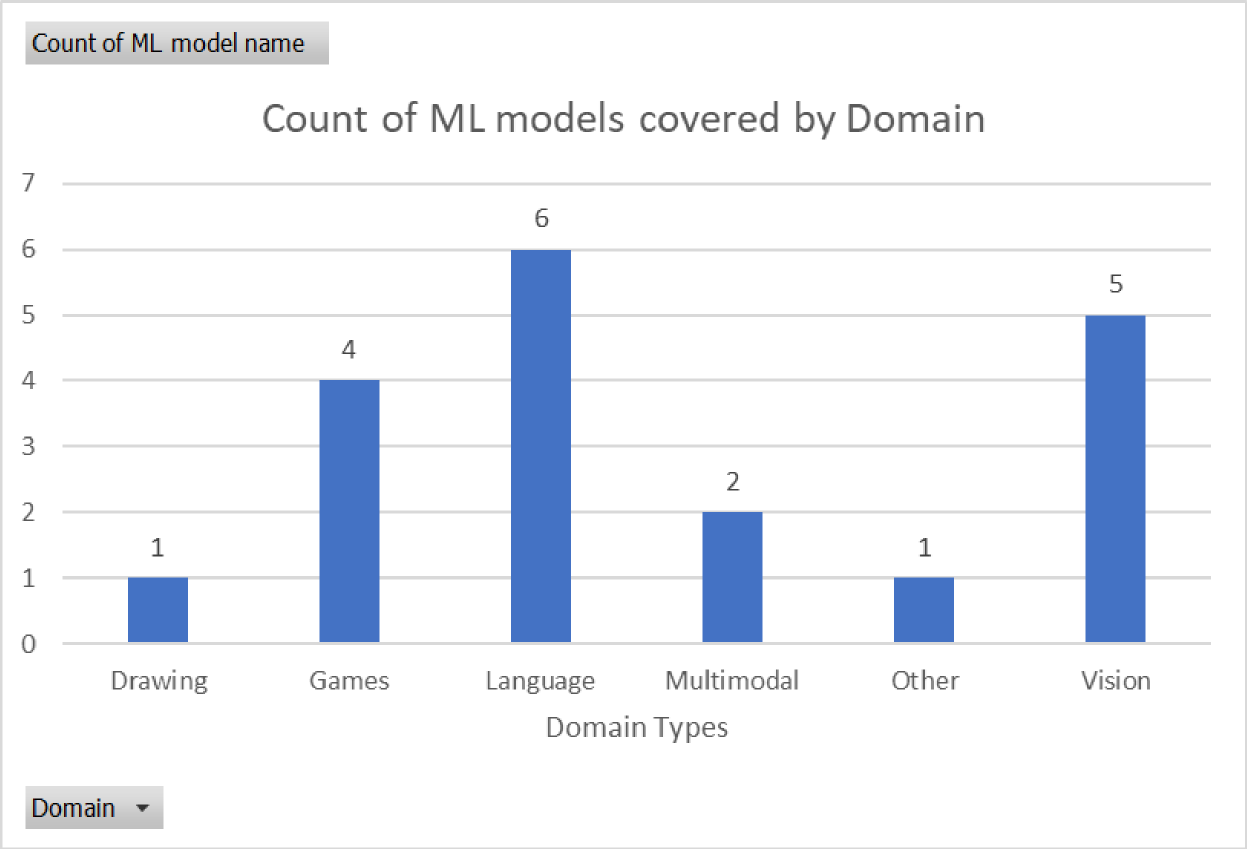
In this article, we introduce a Machine Learning (ML) Scale Map, a list of ML models and their respective sizes and features. The motivation behind composing a ML scale map is primarily to examine how, from the inception of a machine learning model to now, the capabilities of a ML advanced with their increase in size. In this article, we will survey the landscape of diverse tasks of the forerunner and current state-of-the-art ML models. We will categorize by the particular architecture and task of the ML models. At the bottom of each model, a collection of feature statistics such as the number of parameters, layers, nodes, depth, FLOP/s (Floating Point Operations Per Second), training data in bytes, and cost for training will be displayed. Below is a graphic representation of the findings.

[graphic]

Summary:
We hypothesize a plausible correlation between the parameter count of ML models and their capabilities & complexity. However, more recently, we see that the cost of training for ML models is becoming less of an indicator of their complexity because of technological advancements[1] and the accessibility of obtaining training data[2]. Generally, we notice a decrease in ML model training time as the complexity of the model increases.



[1] Technological advancements here include Cloud GPUs, which is being preferred over on-premise GPUs because it is less expensive.

[2] In the very beginning, training data was scarce due to the need to manually label data.

| ML model name | Function/Tasks |
|---|---|
| Perceptron Mark I | Binary classification |
| LeNet-5 | Character recognition |
| AlexNet | Image classification |
| VGG19 | Image classification |
| AlphaGo Lee | Go |
| ResNeXt-50 | Image classification |
| AlphaZero | Shogi, Go, Chess |
| GPT | Text autocompletion |
| BERT | Next sentence prediction |
| GPT-2 XL | Text autocompletion |
| AlphaStar | StarCraft |
| MuZero | Atari Games |
| AlphaFold | Protein folding prediction |
| GPT-3 175B | Text autocompletion |
| DALL-E | Text-to-image |
| CLIP | Zero-shot image classification |
| PaLM (540B) | Language modelling |

| | |
|---|---|
| Gato | Weak generalist |
| Minerva (540B) | Quantitative Reasoning Problems |

The list of Machine Learning models that we will examine are:

1. **The Perceptron/ Mark I Perceptron**
2. **Digit Classifier (LeNet)**
3. **Image Classifiers Section**
    - **AlexNet**
    - **VGG 19**
    - **ResNet**
    - **CLIP**
4. **Language models Section**
    - **Generative Pre-trained Transformer (GPT)**
    - **GPT-2**
    - **GPT-3**
    - **BERT**
5. **Game-Playing AI section**
    - **AlphaStar**
    - **AlphaGo**
    - **AlphaZero**
    - **MuZero**
6. **AlphaFold**
7. **Gato**
8. **DALL-E**
9. **PaLM**

## The Perceptron/ Mark1 Perceptron

1958[3]

Frank Rosenblatt's single-layer perceptron, or single-layer neural network, was implemented in 1958. The linear classifier is the first of many feedforward neural networks, with features like the feature vector and corresponding weights. Here we do not reference multilayered neural networks.

[3] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf

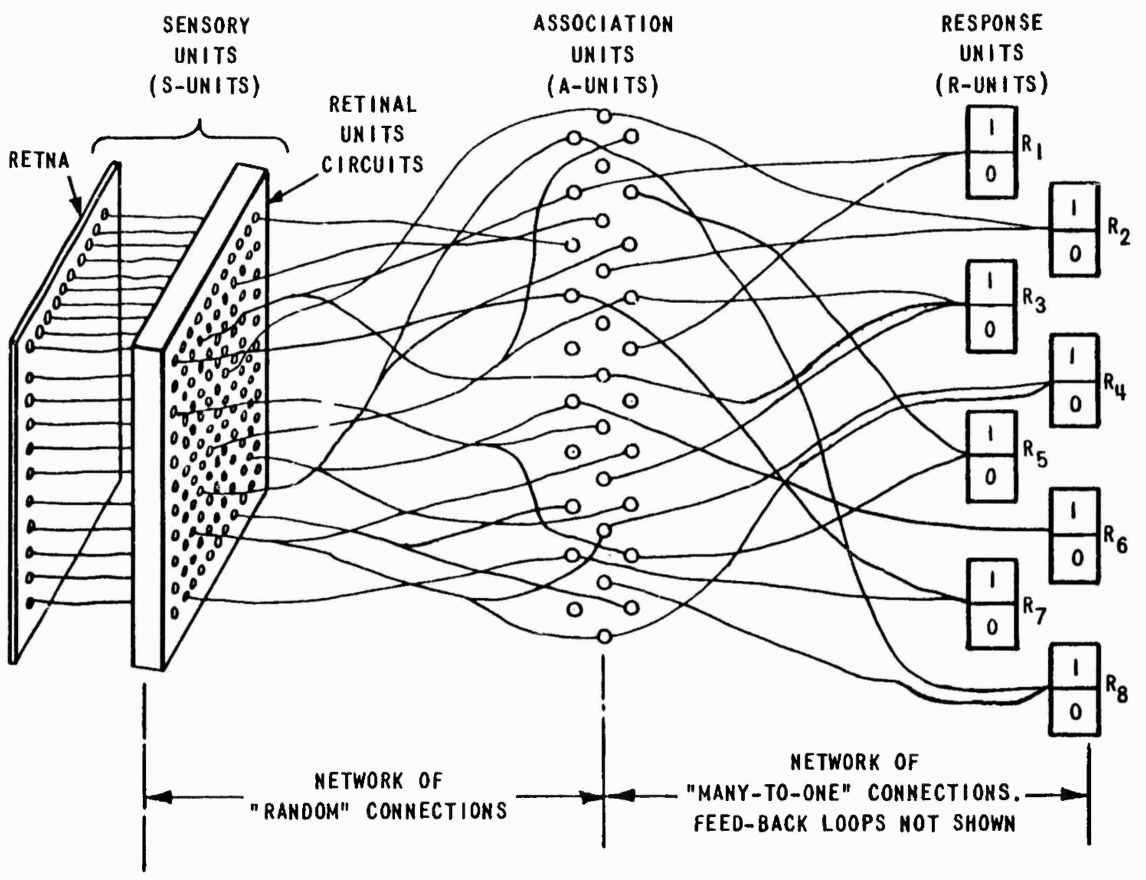

SENSORY UNITS (S-UNITS)
RETNA
RETINAL UNITS CIRCUITS
ASSOCIATION UNITS (A-UNITS)
RESPONSE UNITS (R-UNITS)
R1
R2
R3
R4
R5
R6
R7
R8
1
0
NETWORK OF "RANDOM" CONNECTIONS
NETWORK OF "MANY-TO-ONE" CONNECTIONS. FEED-BACK LOOPS NOT SHOWN

Figure I ORGANIZATION OF THE MARK I PERCEPTRON

1. Number of parameters: 400[4]
2. Layers: 2 (input layer and output layer), no hidden layer
3. Nodes: n amount of real-value non-neuron input, m amounts of neuron output by the vector

[4] "Figure 4.8 Illustration of the Mark 1 perceptron hardware. The photograph on the left shows how the inputs were obtained using a simple camera system in which an input scene, in this case a printed character, was illuminated by powerful lights, and an image focussed onto a 20 × 20 array of cadmium sulphide photocells, giving a primitive 400 pixel image. The perceptron also had a patch board, shown in the middle photograph, which allowed different configurations of input features to be tried. Often these were wired up at random to demonstrate the ability of the perceptron to learn without the need for precise wiring, in contrast to a modern digital computer. The photograph on the right shows one of the racks of adaptive weights. Each weight was implemented using a rotary variable resistor, also called a potentiometer, driven by an electric motor thereby allowing the value of the weight to be adjusted automatically by the learning algorithm." The perceptron asociated one weight per input, for a total of 20x20=400 parameters.
source: Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning

function f:R^n ! {0,1}^m

4. Depth[5]: 0 (no hidden layer)
5. FLOPs to measure complexity of the model: [calculations]
6. Training data in bytes: not documented
7. Cost for training: not documented

Function & Task

As a linear binary classifier, the Mark1 perceptron is capable of distinguishing, or “classifying” hundreds of pixels in front of the perceptron’s camera to be 0 or 1 depending on whether a colored square appeared on the right or left side of the canvas. It conducts on the basis of geometric similarities and differences[6]. The first perceptrons also could classify images, for example, it could distinguish a man from a woman and one alphabet letter from another alphabet letter.

## Image Classifiers Section

### Digit classifier trained with Modified National Institute of Standards and Technology (MNIST) dataset

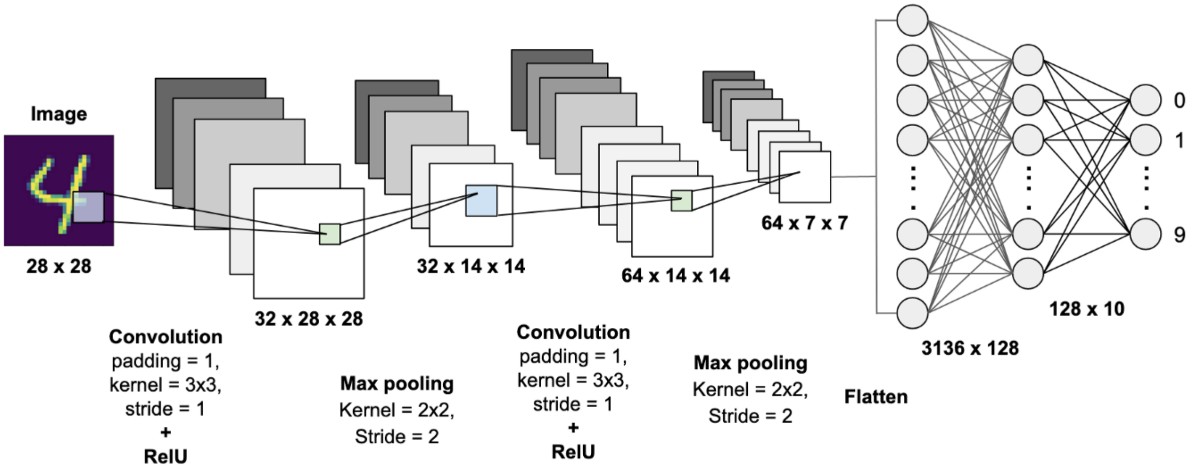
1989[7]

The digit classifier is an image classification model, also categorized as a convolutional neural network. It takes in a public MNIST dataset from the Keras library, with 70,000 total greyscale images representing handwritten digits from 0 to 9 with 28x28 pixels. The model architecture has a baseline model that has a single convolutional layer and 32 filters (or nodes), finally pooling after applying weights in the max pooling layer. A critical shortcoming of the digit classifier would be the confounding nature of “flattening” the images, the spatial distribution is lost. The current model achieves a classification accuracy of above 99%, an error rate between 0.4 and 0.2.

[5] Here we denote depth as the amount of hidden layers in the model

[6] https://anatomiesofintelligence.github.io/posts/2019-06-21-organization-mark-i-perceptron

[7] http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf



1. Number of parameters: 60000[8]
2. Layers: 3 (input layer, 1 hidden layer, output layer)
3. Nodes: 32 nodes in the hidden layer
4. Depth: 6 hidden layer
5. FLOPs to measure complexity of the model: [to be calculated]
6. Training data in bytes: 47.07 megabytes (60000 images * 28x28 pixels) of handwritten digits from 0 to 9. No non-trainable training data
7. Testing data in bytes: 7.84 megabytes (10000 images * 28x28 pixels) of handwritten digits from 0 to 9
8. Cost for training: not documented

Function & Task

Yann LeCun's digit classifier was first implemented for zip code recognition[9]. The MNIST dataset is even more so the standard dataset for deep learning and computer vision. According to Christian's *The Alignment Problem*, this model also cashed checks in ATMs— by the 1990s, LeCun's networks were processing 10 to 20% of all checks in the United States.

## AlexNet

2012[10]

In 2006, Fei-Fei Li created a catalog of computer vision training data. What started as an "image atlas of concepts" with 500 to 1000 images per concept quickly changed into ImageNet, which was composed of

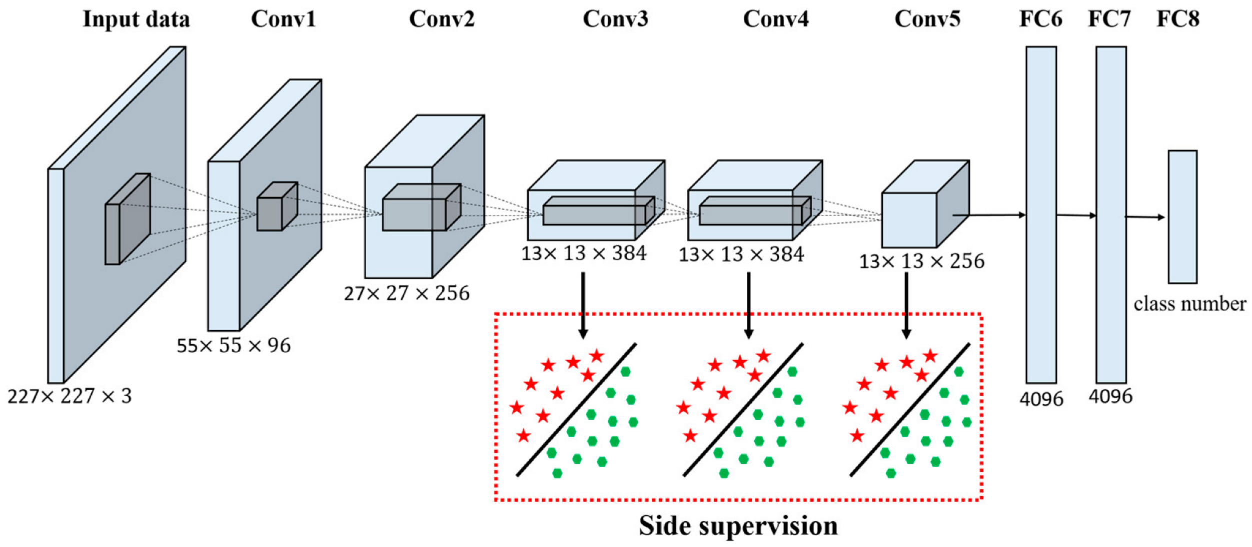
[8] paper.dvi (stanford.edu)

[9] https://web.archive.org/web/20100730053801id_/http://srl.csdl.tamu.edu/courses/SR2008/papers/others/LeCun.pdf

[10] https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

14,197,122 human-annotated images. In 2010, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) prompted researchers to submit algorithms that accurately identified objects and classified images. Scaling large inputs from the ImageNet database, AlexNet was the contest winner of ILSVRC.

AlexNet's excelling feature[11] was the splitting of the network into two— because GPUs at the time were limited to 3 gigabytes— which helped significantly its training module. Another feature of AlexNet was the use of a 0.5 dropout rate, as this will reduce the tendency to overfit. Overfitting means to pay special attention to external noise or converge too quickly. But it doesn't come without a cost: the model's training time is doubled[12].



1. Number of parameters: 62,378,344

Number of weights in convolutional layer = ( size of kernels^2 ) * number of channels from input * number of kernels

Number of parameters in this convolutional layer = number of weights in convolutional layer + number of biases in the convolutional layer

Parameters(convolution 1) = (11^2 * 3 * 96) + 96 = 34,944
Parameters(convolution 2) = 614,656
Parameters(convolution 3) = 885,120
Parameters(convolution 4) = 1,327,488
Parameters(convolution 5) = 884,992

We calculate the number of parameters of a fully connected layer connected to the convolutional layer to be:

Number of weights in a Fully Connected layer connected to a convolutional layer = (width of the

[11] https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/
[12] https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637

output image of the previous convolutional layer ^ 2 ) * number of kernels * number of neurons in fully connected layer + Biases

Parameters(Fully Connected layer) = 6^2 * 256 * 4096  + 4096 = 37,752,832

For from a Fully connected layer to another Fully connected layer, the process is more straightforward:

number of neurons of the first fully connected layer * number of neurons of the second fully connected layer + bias of the second layer

Parameters(Fully connected layer to another Fully connected layer) = 4097*1000 + 1000 =  4,097,000
Parameters(second fully connected layer to another fully connected layer) = 4096*4096 + 4096 = 16,781,312

The max-pooling layers, the image input layer, output layer do not have respective parameters.
The total is 62,378,344 parameters when added altogether.

2. Layers: 13 layers (input, output, 5 convolutional layers, 3 max-pooling layers, 3 fully connected layers)
3. Nodes: 8282
   Number of nodes from the 3 fully connected layers→ 4096 (Fully Connected 1) + 4096 (Fully connected 2) + 100 (fully connected 3) = 8,282
4. Depth: 8 hidden layers (5 convolutional layers, 3 fully connected layers)
5. FLOP/s to measure complexity of the model: 725M[13]
6. Estimated training data in bytes: 78.6432 gigabytes
   A typical ImageNet image is 256x256 pixels, AlexNet was reported to have been trained on 1.2 million ImageNet images.
   Training data in bytes = 256* 256 * 1,200,000 = 78,643,200,000 bytes
7. Cost for training: not documented

Function & Task

AlexNet is an image classifier. It has been used commercially for sports field shots, i.e., classifying the shots into long, medium, close-up, and out-of-the-field shots[14].

Since the publication of AlexNet, many more computer vision related papers have been using GPUs and CNNs (and especially the usage of the ReLU activation function), citing Krizhevsky's work.

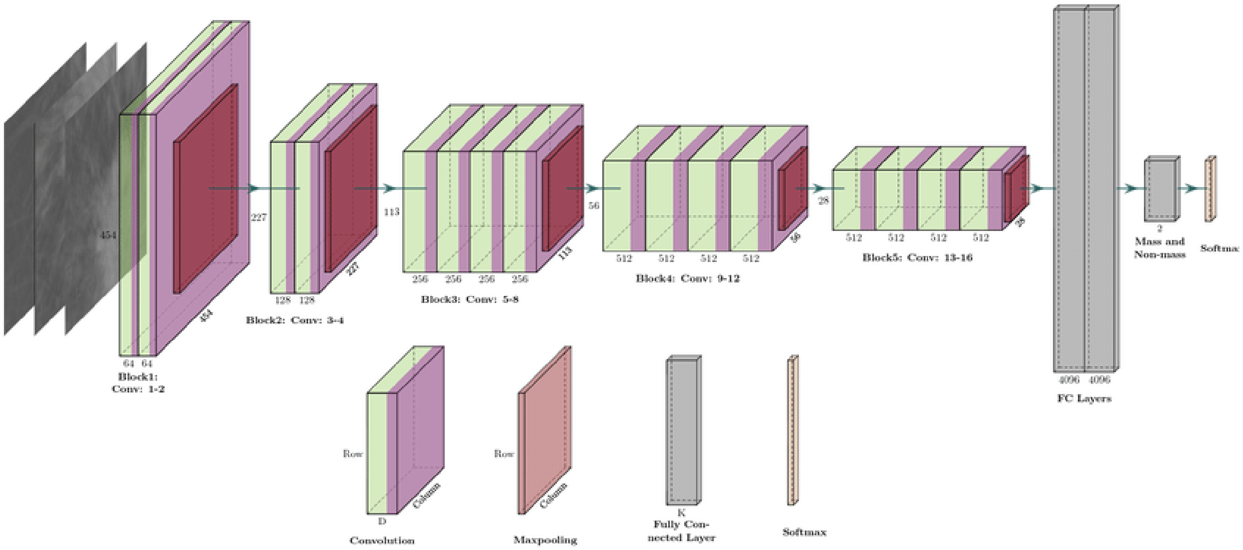
## VGG-19

### 2014

Visual Geometric Group-19 was the successor of VGG-16, both of which are one of the first image classifiers that used transfer learning. Transfer learning is the process of running a previous dataset on the

[13] https://arxiv.org/pdf/1703.08651.pdf
[14] https://mdpi-res.com/d_attachment/applsci/applsci-09-00483/article_deploy/applsci-09-00483.pdf

model and the model retaining the learned knowledge, such that the learned knowledge can be applied to a new dataset. In this way, models are called “pre-trained”.



1. Amount of parameters: 143,667, 240[15]
2. Layers: 19 layers (16 convolution layers, 3 fully connected layers, 5 max-pooling layers, 1 softmax layer)
3. Nodes: 9192 from three fully connected layers
   Each of the two fully connected layers that are connected to a convolutional layer has 4096 nodes, and the one fully connected layer that is connected to another fully connected layer has 1000 nodes.
   4096*2 + 1000 = 9192
4. Depth: 19 layers
5. FLOPs to measure complexity of the model: 19.6B [16]
6. Training data in bytes: 78.6432 gigabytes
   A typical post-processed ImageNet image is 224 by 224 pixels, VGG-19 was reported to have been trained on 1.28 million post-processed ImageNet images.
7. Cost for training: Not documented
8. Source code:
   https://gist.github.com/ksimonyan/3785162f95cd2d5fee77#file-readme-md

Function & Task

VGG-19’s transfer learning method has wide applicabilities such as disease prognosis in X-ray scans, malware detection, workers with/without masks detection. These publications all show novel modifications of the original VGG-19 framework (changing hyperparameters).

## ResNet50

[15] https://resources.wolframcloud.com/NeuralNetRepository/resources/VGG-19-Trained-on-ImageNet-Competition-Data

[16] https://iq.opengenus.org/vgg19-architecture/

## 2015

It has been established that increasing the number of layers in a neural network does not entirely correspond to a higher testing accuracy[17], even without considering the vanishing gradient[18] or the exploding gradient problem. This means that as the network depth increases, accuracy gets more and more saturated and then degrades. For example, the ResNet paper compares the 56 layers neural network with that of the 20 layers, and we note both a higher training and testing error in th 56 layers one.

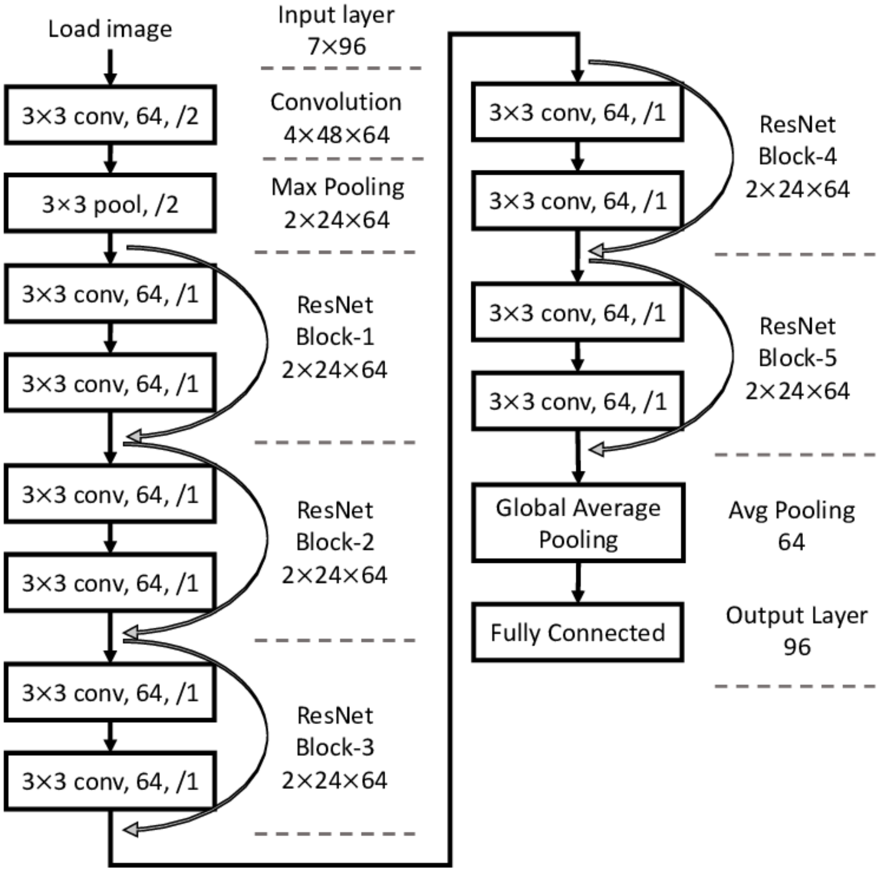
Functionality-wise, as an ImageNet pre-trained convolutional neural network, ResNet50 can classify images from 1000 object categories. It is a model with 50 layers, hence the name, ResNet50. Structurally, the model has five stages each with a convolution block and an identity block, each convolution block has 3 convolution layers and each identity block has 3 convolution layers.

The main feature of ResNet50 is its pioneering lead in Residual Learning. The idea is that instead some layers can "skip" and feed into other layers using skip connections farther in the network. The layers that are most helpful to skip are the layers that do not add value to overall test accuracy. ResNet50 was the first image classifier model that used this feature. The paper states that the ResNet50 model has a top 1 error[19] of 20.74% and a top 5 error of 5.25%.

[17] https://arxiv.org/pdf/1512.03385.pdf

[18] The vanishing gradient problem is when you have too many layers, taking the partial derivative of the error function with respect to the weight can sometimes get too close to zero. During the backpropagation, meaning the weights earlier in the network can't update.

[19] The Top-1 error is the percentage of the time the classifier does not provide the highest score to the correct class, Top-5 error rate is the percentage of times the classifier failed to include the proper class among its top five guesses



1. Amount of parameters: 23 million parameters
2. Layers: 50
3. Nodes: The fully connected layer has 1000 nodes [20]
4. Depth: 48 convolutional layers with 1 Maxpool and 1 average pool layer
5. FLOPs to measure complexity of the model: 3.8B[21]
6. Training data in bytes: 83.89 gigabytes
   A typical ImageNet image is 256x256 pixels, AlexNet was reported to have been trained on 1.28 million ImageNet images.
   Training data in bytes = 256* 256 * 1,280,000 = 83,886,080,000 bytes

7. Cost for training: Not documented

8. Source code: http://ethereon.github.io/netscope/#%2Fgist%2Fdb945b393d40bfa26006

Function & Task

The scalability threshold was expanded upon the introduction of Resnet— it is apparent that we can indeed train a computer vision model that is 150+ layers deep with an incrementally decreasing amount of

[20] https://iq.opengenus.org/resnet50-architecture/

[21] https://iq.opengenus.org/floating-point-operations-per-second-flops-of-machine-learning-models/

error.

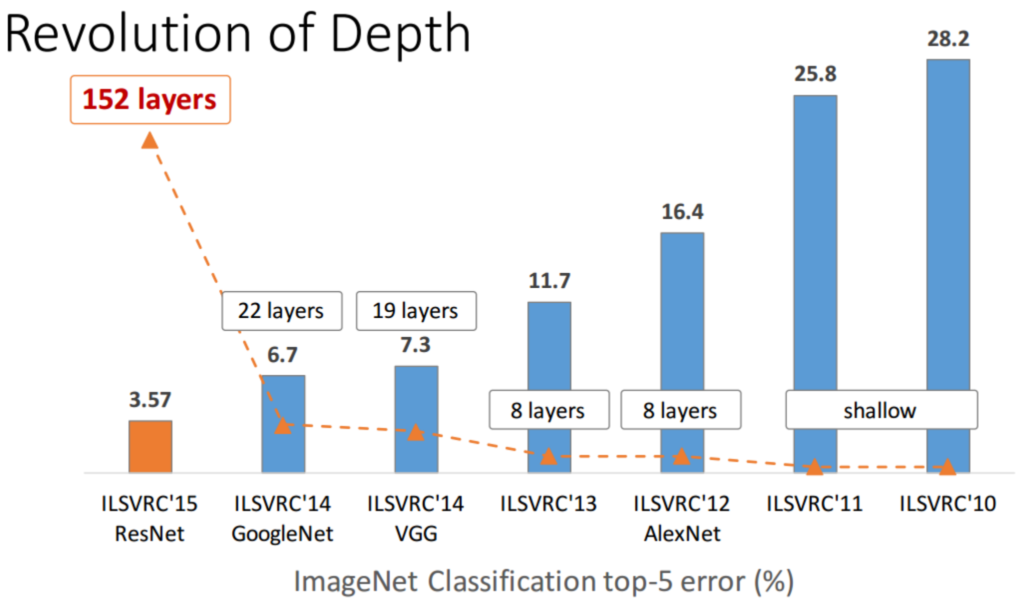


**Figure 1. A list of ILSVRC model submissions and their layers count**
For reference, the winner of ILSVRC 2015 was ResNet152, with 152 layers as opposed to the 2012 winner's 8 layers.

## Contrastive Language-Image Pre-training (CLIP)

### 2021

OpenAI launched CLIP in February 2021. It is a model that solves problems like insufficient vision datasets[22] and has the ability to "adapt" to a new task without additional training examples.

Furthermore, a previous publication[23] has identified that deep neural networks (DNN) sometimes have lower classification accuracy than humans when it comes to distorted images. Noisy and blurry images present a significantly harder task to the DNN than the human visual system, which suggests there is a different internal processing representation of these distorted images. This is a problem that CLIP was also able to solve.

As a zero-shot model, CLIP meets numerous benchmarks without exactly finetuning for "meeting those classification benchmarks". Zero-shot means that the model can infer a specialized task, in particular, one that it is not trained to perform. For example, one-shot means that the model is given one example before it is asked to perform its task. The method that CLIP uses is feeding text-image data pairs, which is then given a training task of identifying which of 32,768 sample text snippets a given image belongs to.

[22] Additional training datasets (text-image pairs) are costly to acquire
[23] A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions

CLIP's distinguishing feature is its ability to lower compute using a contrasting objective. A contrasting objective is essentially what the name suggests— we contrast image samples. Similar image samples are clustered in the embedding space after loss minimization.

1. Amount of parameters: 33M
2. GPUs: 256 GPUs for two weeks
3. Training data in bytes: not specified
4. Cost for training: not specified

Function & Task

CLIP is capable of conducting fine-grained object classification, or subordinate concept level classification. Fine-grained object classification means being able to categorize subobjects within a class, e.g., rather than having a single label 'bird' for all bird species, it can categorize types of birds. CLIP being able to geo-localize means it can know where an image was captured. It is also able to perform Optical Character Recognition (OCR) at high accuracy.

## Language Model Section

### Bidirectional Encoder Representations from Transformers (BERT)

2018

BERT[24] is a transformer language model developed by Google, with an architecture almost identical to that of the original transformer. There is subsequently a BERT Large model that has a parameter size of 340M, here we refer to BERT base.

A particular feature of BERT is next sentence prediction (NSP). Almost half of the model training is dedicated to this feature— therefore it is optimized for prediction.

1. Amount of parameters: 110M
2. Layers: 12 Transformer layers
3. TPU chips: 64
4. Training data in bytes: 3.3 billion words as training input data
   An estimated 2.5 billion text-paragraph words come from Wikipedia and estimated 800M words from BooksCorpus
5. Cost for training: estimated to be $6912, excluding R&D[25]
6. FLOPs: 22,500,000,000[26]
7. Training data in bytes: 16 GB
8. Time for training: 64 TPU chips trained over 4 days, or 16 Cloud TPUs over 4 days
   Estimated cost for 1 Cloud TPUv2 per hour is $4.5

[24] https://arxiv.org/abs/1810.04805
[25] https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/
[26] https://aclanthology.org/2020.findings-emnlp.372.pdf to find the BERT size chart

24 hours * 4 days * $4.5 TPU chip/hr * 16 Cloud TPUv2 = $6912 for pre-training

9. Source code: https://github.com/google-research/bert

### Function & Task

BERT is able to conduct sentiment analysis, which is to discern positive and negative connotations. It can conduct chatbot conversations, predict text by the never-before seen Masked Language Modeling (MLM)[27] method, generate text given keywords, understand context, and parse and summarize a long document. Another feature of BERT is the ability to perform Named Entity Recognition (NER), which is annotating words in a sentence with the entity it belongs to. For example, if a sentence is “Palpatine is eating some flan”, then Palpatine will be associated with the entity ‘person’ and flan will be associated with the entity ‘food’.

Currently, there are different domains as to where BERT is being used. This is referring to BioBERT[28], finBERT[29], and patentBERT[30] which are models adopted from BERT that are traversing the biomedical corpus, financial corpus, and patent corpus.

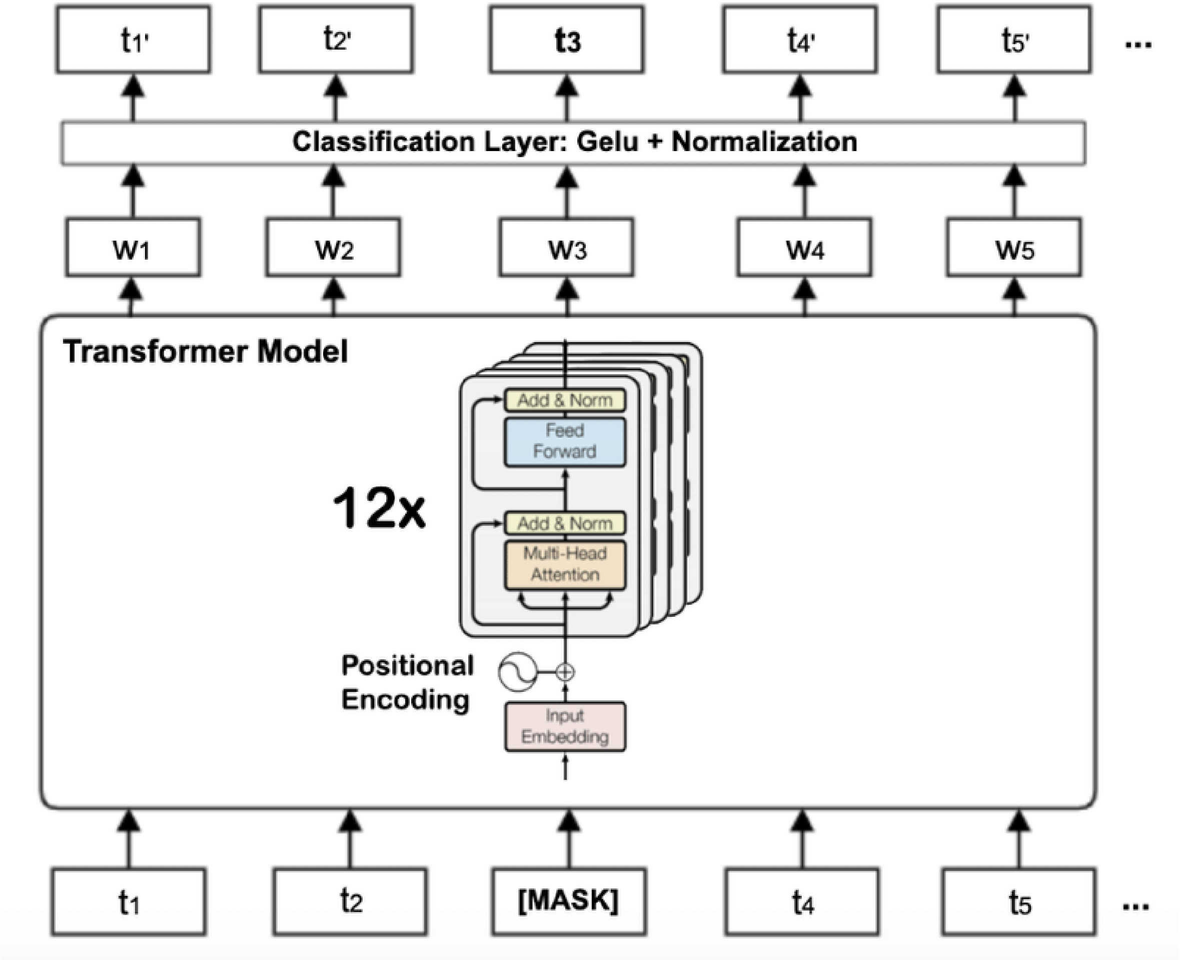
Now, almost every English Google search query is processed by BERT.

[27] Masked Language Modeling is hiding or “masking” some tokens in a sentence. To be able to predict the missing token, the model must have contextual understanding of the sentence.

[28] https://arxiv.org/abs/1901.08746

[29] https://arxiv.org/abs/1908.10063

[30] https://arxiv.org/abs/1906.02124



## Generative Pre-trained Transformer (GPT)

2018

Natural Language Processing (NLP) is a branch of AI where machine learning algorithms enable the interpretation and production of rule-based human language. Being a cross of linguistics and computer science, some NLP tasks include but are not limited to information retrieval, sentiment analysis, machine translation, and question answering.

Some examples of the first NLP models are: Bag of Words (BOW) model, TF-IDF model, Word2Vec. Open AI's first GPT[31] model was established on the basis of these precedents— evaluating given text data, weighing words by relevance, and clustering words by similarity in English. The paper published in 2018 detailed the model's pre-trained dataset called BookCorpus, which is a collection of 7000 unpublished books. The model is tasked with both text prediction and text classification.

A transformer-based encoder-decoder model is important for our discussion of GPT-1, GPT-2, and GPT-3. An encoder takes the input and make it into a more intelligible format such as a vector, a map, or a tensor.

[31] https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

The information is now extracted from the input and is often referred to as a feature matrix or feature vector. Then, the decoder receives the processed representation and changes it back to a sensical format, which is decoding.

The model has Nx (shown below in figure) amounts of encoder and decoder blocks, such that the transformer's task is to extract features from each encoder and feed them into the decoder. Positional encoding ensures that the tokens are in the right order and word-embedding/input-embedding associates similar words together. Normalization of the embedding vectors makes sure that the mean and standard deviation of the vector doesn't shift, which may in turn negatively affect training. With a multi-head attention step in each of the blocks, we control the amount of information mixing.

GPT-1 is a transformer-based decoder-only model.

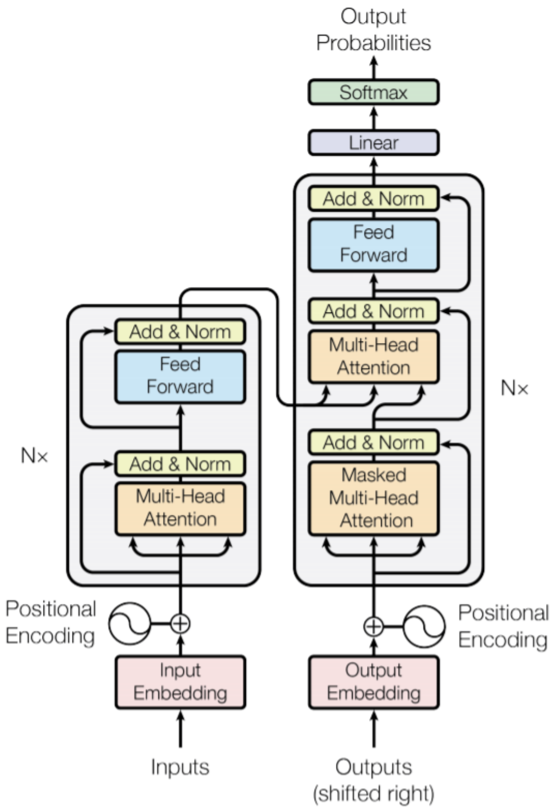


Figure. **The Transformer-based Encoder-Decoder model**

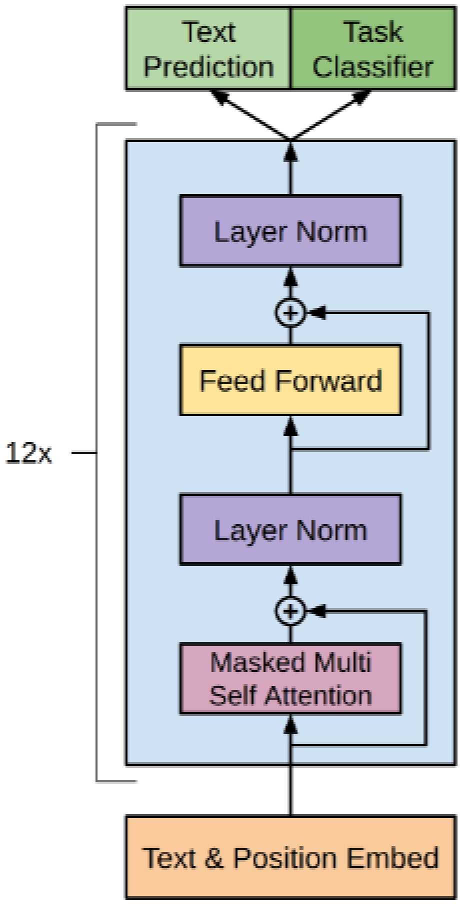
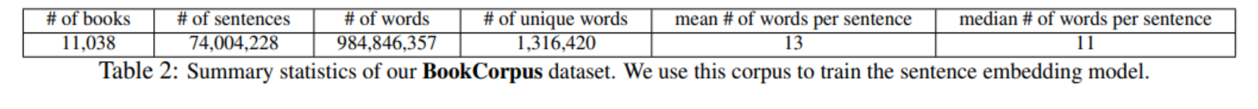


**Figure. GPT-1 model**. Transformer-based decoder-only model

Below we would like to add another variable to the language models: perplexity. Generally speaking, the lower the perplexity is, the better the language model is to predict unseen words in a test sentence. For reference, the perplexity of a unigram is 962, that of a bigram is 170, and that of a trigram is 109— where n is the number of words in a sequence of n-grams.

1. Amount of parameters: 117 million
2. Layers: 37 layers (12 decoder blocks)
3. Nodes: no information was given on the number of nodes
4. Hidden Layers: not specified
5. FLOPs to measure complexity of the model: 1.10E9[32]
6. Training data in bytes: 7000 unpublished books from BookCorpus, 1E9 bytes[33]
   Given that there is approximately 1E9 tokens from BookCorpus dataset, and assuming that 1 byte is 1 token, then 1E9 bytes.

[32] https://www.metaculus.com/questions/9519/flops-used-for-gpt-4-if-released/
[33] https://hellofuture.orange.com/en/the-gpt-3-language-model-revolution-or-evolution/



| # of books | # of sentences | # of words | # of unique words | mean # of words per sentence | median # of words per sentence |
|---|---|---|---|---|---|
| 11,038 | 74,004,228 | 984,846,357 | 1,316,420 | 13 | 11 |

Table 2: Summary statistics of our **BookCorpus** dataset. We use this corpus to train the sentence embedding model.

Here I was confused why the GPT-1 paper states that the training data was based on 7000 unpublished books instead of the total number of books from BookCorpus, which is clearly 11,038 books[34].

7. Cost for training: Not documented
8. Link to implement it yourself: https://github.com/openai/finetune-transformer-lm

Function & Task

There are many tasks that GPT-1 (or GPT) was able to do: tokenization, stopword removal, and word sense disambiguation.

Tokenization is the idea of splitting words in a sequence that represents a specific idea. For example, we can say that tokenization by blanks would be counting each word as a token in English. However, there are many other, more advanced ways to tokenize.

Stopword removal is the idea of processing the input data by filtering out words that are not semantically significant.

Word sense disambiguation is the idea of taking the word given in the context, especially in situations where it is knowledge-based and not syntactical.

## Generative Pre-trained Transformer 2 (GPT-2)

### 2019

GPT-2[35], the successor to GPT-1 (or GPT), saw some significant modifications. First, to encourage the diversity of content, the research group broadened the training dataset to 8 million human-curated web pages (some example pages are: Google, Archive, Blogspot, GitHub, NYTimes, Wordpress, Washington Post). Amongst the most important changes adopted by OpenAI on GPT-2 was research on potential model misuse and how to leverage greater social benefit.

Until now, previously we only discussed ReLU activation functions. Another significant change is that GPT-2 uses GeLU, the Gaussian Error Linear Unit. This cumulative distribution function uses the phi(x) function instead— x*phi(x) and not ReLU's sigmoid function, x*sigmoid(x). Like ReLU, activation functions filter out forward propagations in the neural network. However, the GeLU's advantage is that it weights inputs by value instead of by whether the input is positive or negative, so it prevents a "gating" or quick elimination of neurons.

Open AI addresses the fact that previously their published paper's parameter counts are inaccurate, so

[34] https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Zhu_Aligning_Books_and_ICCV_2015_paper.pdf

[35] https://www.google.com/url?q=https://openai.com/blog/better-language-models/&sa=D&source=docs&ust=1660950488491014&usg=AOvVaw3jNzGVtwyIQ7rxXwrYzMfc

small, medium models did not capture the current model's parameter count. For example, in February 2019, the small model's parameter count was 124M whereas the medium model's was 345M in May. The most recent model's parameter count is 1.5B.

1. Amount of parameters:
    - Small : 117M
    - Medium: 345M
    - Large: 762M
    - XL: 1542M
2. Layers:
    - Small: 12 decoder layers
    - Medium: 24 decoder layers
    - Large: 36 decoder layers
    - XL: 48 decoder layers
3. Nodes: no information given on the number of nodes
4. Hidden Layers:
    - Small: no information found
    - Medium: no information found
    - Large: no information found
5. FLOPs to measure complexity of the model: 2.49E21[36]
6. Training data in bytes: 40 gigabytes
7. Cost for training:  $256 per hour, total hours not specified by OpenAI
8. Perplexity in Penn Treebank dataset: 35.76
9. Source code: https://github.com/openai/gpt-2

Function & Task

GPT-2's general tasks are to answer factual and non-factual questions, write short novels given supplementary information prompts, and conduct calibrated data analysis. Machine translation, summarization are some examples of GPT's many tasks.

Based on the model card provided by OpenAI for GPT-2, the model is able to provide writing assistance such as grammar and autocompletion on normal prose or code; it is able to generate creative writing such as fictional text, poetry, and other categories of literary art; it is able to entertain the user with games, chat bots, and various other types when prompted. [37]

GPT-2 based models such as Tabnine[38] were launched in July 2019, and are able to complete written code.

## Generative Pre-trained Transformer 3 (GPT-3)

[36] https://www.metaculus.com/questions/9519/flops-used-for-gpt-4-if-released/

[37] https://github.com/openai/gpt-2/blob/master/model_card.md

[38] https://www.tabnine.com/

2020[39]

Open AI's most recent GPT model was launched in May 2020 and was in its beta-testing stage until July 2020. Some of the new features introduced in the improved model include the concept of Byte Pair Encoding (BPE)— a method to simplify or to "compress" a string of words. Adopted into GPT-3, BPE facilitates "breaking" long and rare words into more frequently used sub-words: it is highly valuable in a limited vocabulary memory size case.

When testing autoregressive language models, another important benchmark is in-context learning performance. Researchers tested GPT-3 using zero-shot, one-shot, and few-shot metrics on GPT-3. Zero-shot means that the model can infer a specialized task, in particular, one that it is not trained to perform. One shot means that the model is given one example, and a few-shot means it is given a few examples before asked to classify/perform.

Traditionally, the model is updated every time it takes in an example task— this is called fine-tuning. However, with n-shot learning, there are only n example task input and outputs. The difficulty of arriving at the appropriate output incrementally increases as the number of example tasks decreases.

The pre-training dataset takes in Common Crawl, WebText2, Books1, Book2, and Wikipedia dataset, boasting a 410 billion BPE tokens count. Up to date, GPT-3 has 175 billion parameters, which is the largest parameter counts for any language models to date.

1. Amount of parameters: 175 billion
2. Layers: 96 decoder layers
3. Nodes (neurons): 80-100 million
4. Hidden Layers: 96
5. FLOPs to measure complexity of the model: 3.14x10^23[40]
6. Training data in bytes: 45 Terabytes
7. Cost for training: 12 million USD
8. Perplexity in Penn Treebank dataset: 20.5[41] for the zero-shot GPT-3 model
9. Source code: currently, GPT-3 is not open-sourced. It is available through OpenAI's API. The GPT-3 beta API playground is available to everybody to test the model's functionalities: https://beta.openai.com/playground

Function & Task

The GPT-3 language model tasks include and are not limited to, when prompted: generation of news articles, translation of human descriptions of a javascript-based website into code, demonstration of enhanced multilingual text processing from its predecessor.

On a more creative note, the model is able to compose literature in the rhetorical styles of renowned writers such as Shakespeare, Edgar Allen Poe, Ernest Hemingway. Prose, poetry, parody, puns— the model redefines our expectations of what an AI can compose. The model is able to personify notable characters in history and literature, like Cleopatra and Steve Jobs.

[39] https://arxiv.org/pdf/2005.14165.pdf

[40] https://lambdalabs.com/blog/demystifying-gpt-3/

[41] https://paperswithcode.com/sota/language-modelling-on-penn-treebank-word

GPT-3 inspires a series of other softwares.[42] Copilot, an AI pair programmer tool, uses GPT-3 as a supporting model to complete lines of code followed by user-input. In 2021, the Chinese version of a large-scale autoregressive language model was found to closely resemble GPT-3 called PanGu-Alpha.

GPT-4 is currently being developed with a predicted release date in fall 2022.

## Game-Playing Section

The subsequent narrow AIs specialize in games— mastering a subset of rules and strategies that have proven to be effective against humans. The function and tasks of these models are to perfect the gameplay or to optimize the winning score.

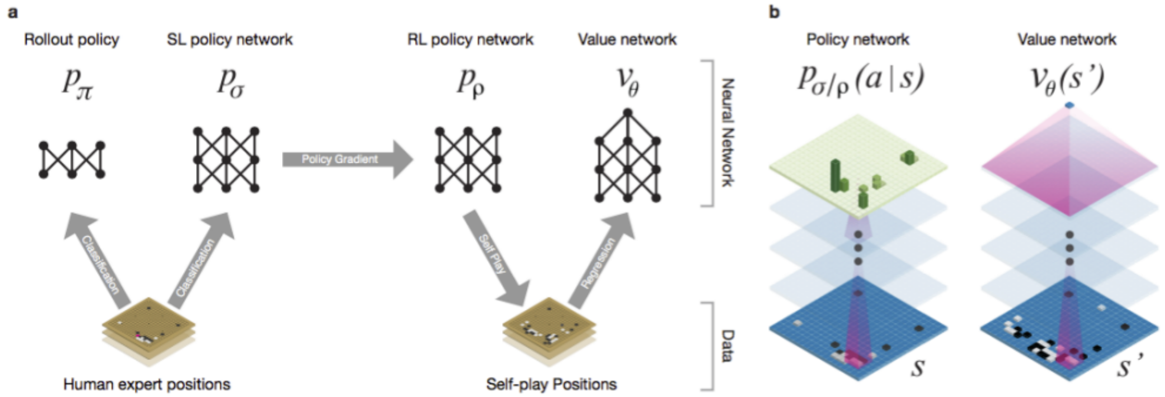
### AlphaGo

2016

Go is a two-player board game with the objective of enclosing the pieces of one's opponent; the game is won when either of player resigns or the board is saturated with pieces. AlphaGo[43], equipped with reinforcement learning and a large database of historical winning moves, was Deepmind's game-play model. In 2016, AlphaGo played against a world-class Go player, Lee Sedol and won 4 to 1 in the Google DeepMind Challenge Match.

AlphaGo has three neural networks, the supervised learning policy network, the reinforced learning policy network, and the value network. AlphaGo is driven by the Monte Carlo Tree Search algorithm in addition, which is a step-by-step process to maximize the expected value from each node/game state. This yields a "best" move, which is then played. Additionally, AlphaGo's policy network suggests the moves that the model will look at, let the value network discard the moves that will have a deterministic outcome of winning or losing.

In AlphaGo, the model is given human data, domain knowledge, and known rules.



[42] https://www.analyticsinsight.net/top-10-gpt-3-powered-applications-to-know-in-2022/

[43] https://www.nature.com/articles/nature16961

1. TPUs: 48
2. Estimated Cost to train: 35 million USD[44]

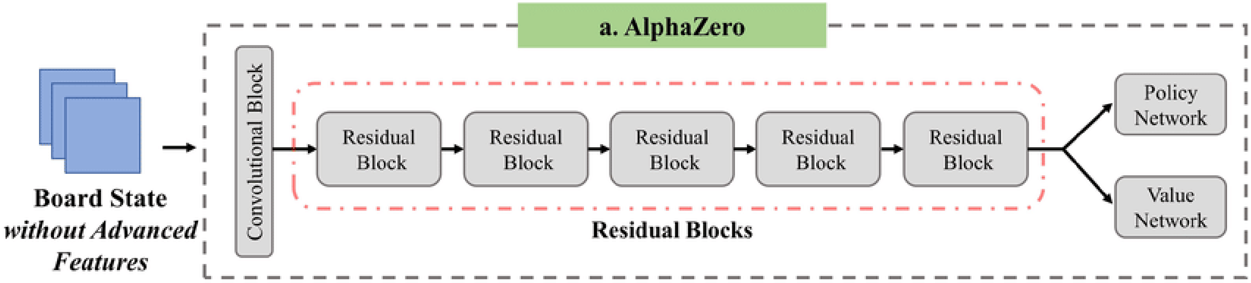
## AlphaZero

2017

Deepmind's AlphaZero expands the horizons of games that AI can master. This game-playing model was able to perfect chess, shogi, and go by self-playing using only one algorithm. Although there are in total 5064 TPUs, for the game of chess, AlphaZero only uses 4 for the game Go.[45]

Whereas the game of Go's board size is 19 by 19, a chess board has 64 spaces. These two games, when compared side-by-side, vary in complexity significantly. While chess has 10^120 possible game configurations, Go can have 10^174 possible configurations. Finally, Shogi has 10^224[46] game configurations. Shogi has a much higher gane configuration because in the game, each player has control of twenty pieces. However, once a player captures an opponent's piece, the player can claim and use the piece.

In AlphaZero, the model is given only known rules.



1. Residual blocks: 19
2. Steps: 700,000
3. TPUs: 5000 first-generation, 64 second-generation
4. Estimated cost to train:

## MuZero

2019

MuZero was a pioneering model in the RL frontier. Researchers at Deepmind launched MuZero[47] in 2019, introducing a model that adapts to a visually-challenging domain without any prior knowledge and achieves superhuman performance. It masters the games chess, shogi, go, and atari. The model, again,

[44] https://www.yuzeh.com/data/agz-cost.html
[45] https://www.chess.com/article/view/whats-inside-alphazeros-brain
[46] https://www.shogi.cz/en/about-shogi
[47] https://arxiv.org/abs/1911.08265?source=techstories.org

uses a learned model and tree-based search. Its performance was evaluated against all 57 Atari games, each resulting in an equal performance with its known-rules predecessor AlphaZero.

The most significant feature of MuZero was its use of implicit learning. Implicit learning is learning about the environment or the structure of the task without intention, or conscious awareness. The model was able to do this by transforming the board of Go or the screen of Atari into an input, and compute it as a hidden state. In each hidden state, the model finds the predicted move (policy network), the predicted winner (value network), and the points rewarded.

In MuZero, no previously known rules were given.

1. Residual blocks: 16
2. TPUs: 16 third-generation TPUs

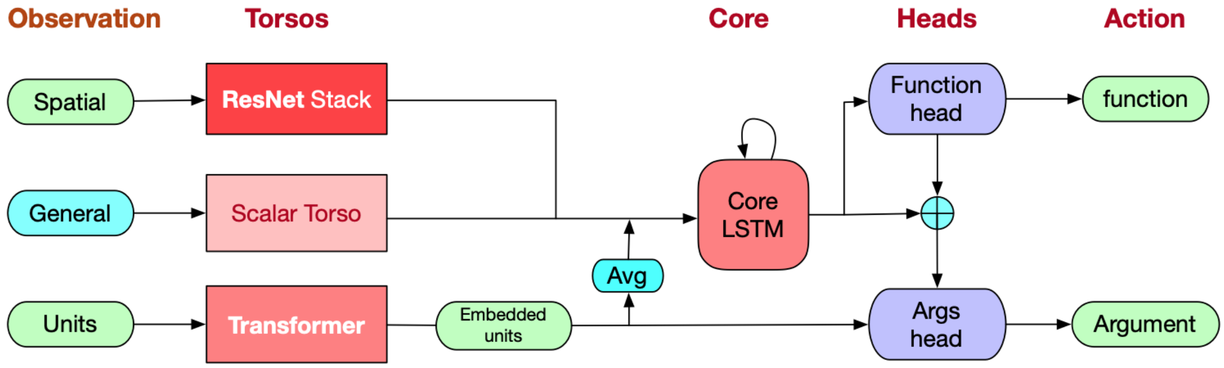
## AlphaStar

2019

Starcraft II is a strategy video game that gained popularity in the 2000s. The game is one of the most played in eSports and holds annual championships. Deepmind's expansion into a real-time and high-complexity game such as Starcraft II tests the performance of game play AI systems. Introducing AlphaStar[48]— a model tasked with perfecting Starcraft II. The game's difficulty comes from the fact that it requires macro-level maintenance of the working environment economy as well as the micro-level management of worker units, structures, and bases.

More specifically, AlphaStar has a large action space— there are 10 to 26 legal actions in every time step. For example, there are 16 building objects alone and one game-play usually takes an hour. The number of instances one can choose an object and build it is countless. So although it's a strategy game, Starcraft II doesn't have a single best strategy to be played.

MaNa, one of the strongest Starcraft II players, lost to AlphaStar 0 to 5 after the model had two weeks of training. Quoting the Deepmind page for AlphaStar, the model advantages are "superior decision-making, rather than superior click-rate, faster reaction times, or the raw interface."



[48] https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii

1. Residual blocks: 16
2. TPUs: 16 third-generation TPUs
3. Estimated cost for training: $6,488,064
   12 agents (three races in the game, three exploiter races, and six exploiter agents)
   Each agent was trained 44 days
   16 third-generation TPUs per agent
   Average cost of TPU per hour in 2018 was $32

   12 agents * 16 TPUs/agent * $32/TPU * 44 days * 24 hours/day = $6,488,064

## AlphaFold

2018

Protein structure prediction technology sits at the center of bioinformatics and computational biology. The protein's function is heavily determined by its structure. Knowing this, we can modify and control the protein to perform certain functions. For example, by predicting an amino acid chain's structure computationally, researchers have successfully produced marketed drugs[49]. Furthermore, it is generally a good idea to understand the proteins involved at the molecular level of complex diseases.

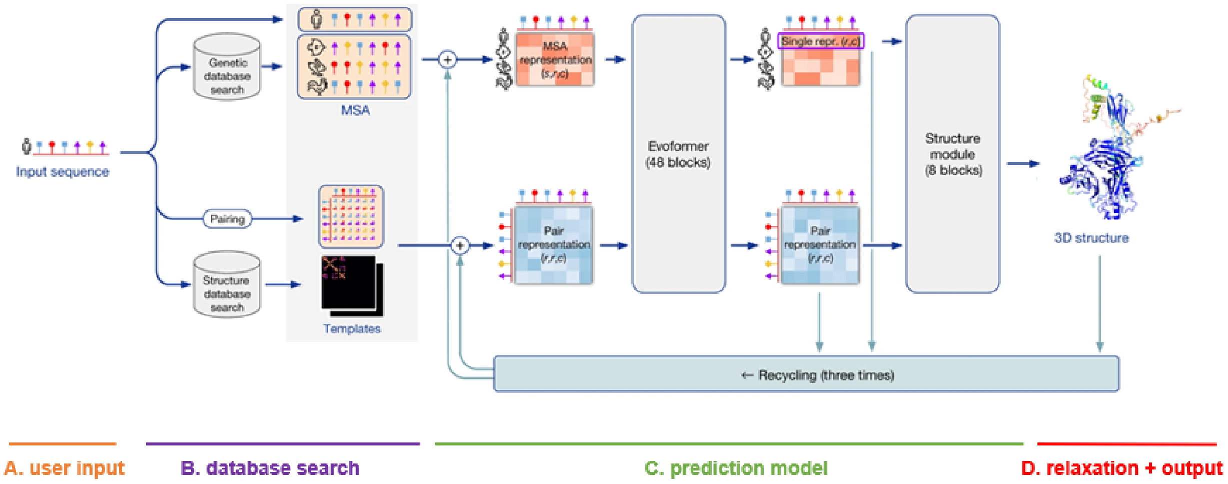
The protein database such as the Research Collaboratory for Structurally Bioinformatics (RCSB)[50] Protein database (PDB) provides 3D information about the protein, annotations on associated small molecules, sequence, and more. These databases are crucial in protein prediction software— like I-TASSER and RaptorX.

AlphaFold was launched by Deepmind in 2018, performing at much higher accuracy than previous models. In the Critical Assessment of Structure Prediction 2020, AlphaFold achieved a high enough accuracy to solve the problem of protein structure prediction. AlphaFold is currently open-sourced.

The AlphaFold model can predict the protein structure by taking in an amino acid sequence, looking at sequence alignment precedents, and then recycling through an Evoformer and a structure module. More specifically, the first step is to perform a multiple sequence alignment (MSA) to keep all the sequence matches. The Evoformer blocks then take the MSA representation and the 2D pair representation and deliver them to the Structure module, where the protein is folded. Recycling means that this process is being iterated 3 times until we arrive at our 3D structure.

[49] https://pubmed.ncbi.nlm.nih.gov/30242117/#:~:text=The%20first%20protein%20structures%20revealed,as%20substrates%20in%20enzymatic%20reactions.

[50] https://www.rcsb.org/



1. Number of parameters: 93M[51]
2. Residual blocks: 220
3. TPUs: 128 TPUv3 core
4. Estimated cost to train: $704,128
   Deepmind did not release the training cost to the public, so here is my approximation based on previously available model costs. According to the paper, the initial training stage takes approximately one week, and the fine-tuning stage takes approximately 4 more days. The paper states that they also requested four GPUs.

   Cloud TPUv3 core per hour costs $32
   128 TPUv3 core
   11 days * 24 hours = 168 hours
   Four ~$4,000 v100 NVIDIA GPUs

   $32 TPUv3 core/hr * 128 TPUv3 core * 11 days * 24 hr/day + (4 * $4000 v100 GPUs) = $704,128

5. To try yourself: https://github.com/deepmind/alphafold

Function & Task

AlphaFold allows for protein structure prediction free to public use at extremely high accuracy[52]. With this level of accuracy, Deepmind claims, AlphaFold and consequently AlphaFold 2 has successfully solved the protein folding problem[53].

[51] https://www.uvio.bio/alphafold-architecture/#inference

[52] https://predictioncenter.org/casp14/zscores_final.cgi?formula=gdt_ts documents AlphaFold 2's performance against other models in the 14th Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction

[53] https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

## Gato

2022

Deepmind introduced Gato in May 2022, a generalist agent that integrates language models capabilities, performs physical movements, annotates images, plays games like Atari and Go. Gato can perform a total of 604 trained tasks— however, some

Gato aims to test if a single agent that is versatile in multiple tasks and functions is possible, and what additional features must be implemented to make it possible. Researchers hypothesize that it is achievable once scaled to higher parameter count, a larger compute, and a bigger dataset.

Gato is a generalist agent— a model that is not specialized in one task. To truly appreciate Gato's functions, researchers believe that there is value in building a precedent for an agent capable of combining all kinds of models. It is leading the sentiment that, if more compute and memory are granted, it will mend Gato's shortcomings such as image annotation inaccuracies.

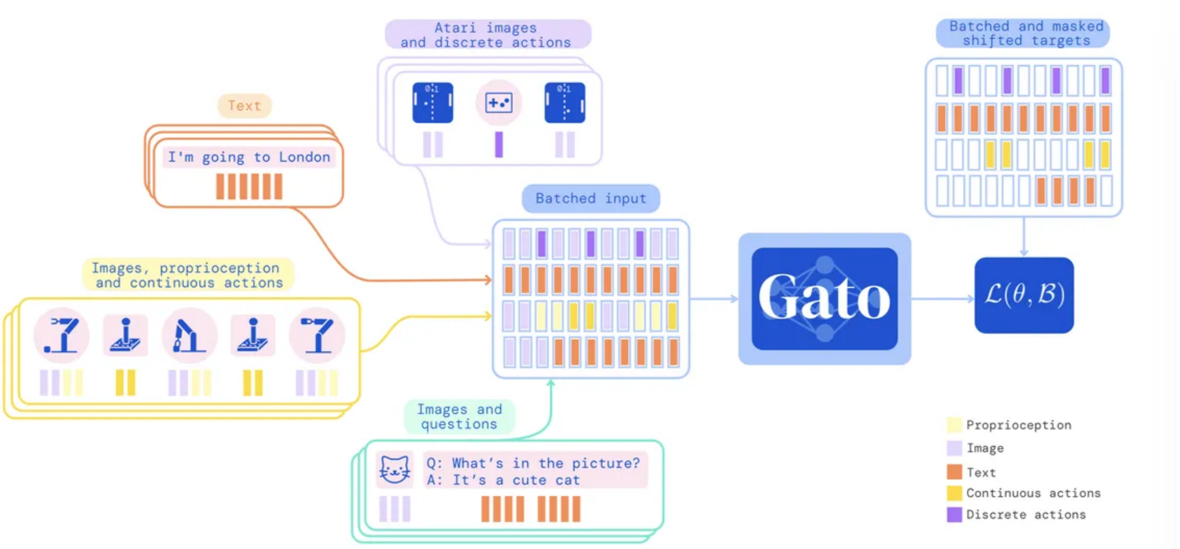

Atari images and discrete actions
Batched and masked shifted targets
Text
I'm going to London
Batched input
Gato
$\mathcal{L}(\theta, \mathcal{B})$
Images, proprioception and continuous actions
Images and questions
Q: What's in the picture?
A: It's a cute cat
Proprioception
Image
Text
Continuous actions
Discrete actions

Figure 2 | **Training phase of Gato.** Data from different tasks and modalities is serialized into a flat sequence of tokens, batched, and processed by a transformer neural network akin to a large language model. Masking is used such that the loss function is applied only to target outputs, i.e. text and various actions.

1. Number of parameters: 1.18B[54]
2. Layers: 24
3. Number of neurons: 49152
   Given layer width is 2048 and there is 24 layers
   2048 layer width (number of neurons in a single layer) * 24 layers = 49152 neurons
4. Cost for training: Not specified*

[54] https://arxiv.org/pdf/2205.06175.pdf

However, according to Lennart Heim[55] estimates the cost to train to be about $50k on Google Cloud

5. TPUs: one 16x16 TPU v3 slice

Function & Task

As a general purpose system, Gato is capable of performing chatbot conversations, conducting translations, finding an image to the text prompt or finding the corresponding text annotations to the image input, writing a short paragraph based on given background information, composing a poem, extracting factual information.

254 DM Lab[56] tasks, 51 ALE Atari tasks, and 46 BabyAI[57] tasks are some example control environment tasks that gave definite scores. A more comprehensive list of tasks is found here[58]. In particular, Gato scored much higher on DM Lab's 3D puzzle than the average human.

Outside of the virtual environment, it is also able to apply torque on robotic arms (joint torque), perform button presses on computer games, stack blocks by moving robotic arms.

## DALL-E

2021

When Open AI's image generator DALL-E[59] was launched in 2021, it carried high hopes of creating futuristic art alongside channeling the public's creativity. It is doing exactly that and more— it is an extension of GPT-3 model by transforming text into pictures.

Although Open AI has not published a paper on DALL-E specifically yet, we still are provided with its features. Its most surprising feature is the ability to perform zero-shot text-to-image generation. Zero-shot means that the model can infer a specialized task, in particular, one that it is not trained to perform.

DALL-E 2 was launched in 2022, with almost 1 million users on the product waitlist. Some newly added features include letting users import their own images to overlay variations upon a given prompt, saving generated images in the DALL-E platform, and giving users the right to commercialize the images produced by DALL-E.

1. Number of parameters: 12B
2. Layers: 64
3. Estimated cost for training: $131,604
   https://twitter.com/alexjc/status/1347458546636619778

[55] https://forum.effectivealtruism.org/posts/4m69jEBWxrqnjyuZp/deepmind-s-generalist-ai-gato-a-non-technical-explainer#:~:text=Scaling%20Laws,-Scaling%20laws%20are&text=On%20Twitter%2C%20Lennart%20Heim%20estimates,the%2Dart%20language%20model).

[56] Deepmind lab, according to its source code github page, functions "to provide a suite of challenging 3D navigation and puzzle-solving tasks for learning agents"

[57] BabyAI, according to its source code github page, is "a platform used to study the sample efficiency of grounded language acquisition"

[58] https://arxiv.org/pdf/2205.06175.pdf

[59] https://openai.com/blog/dall-e/

Following a Twitter thread, there seems to be some evidence on GPU use and inferred calculations of $131,604. OpenAI has not made the cost a part of public information.

4. GPUs: 256
5. FLOPs: 4.7E+22[60]
6. Source code: https://github.com/openai/dall-e

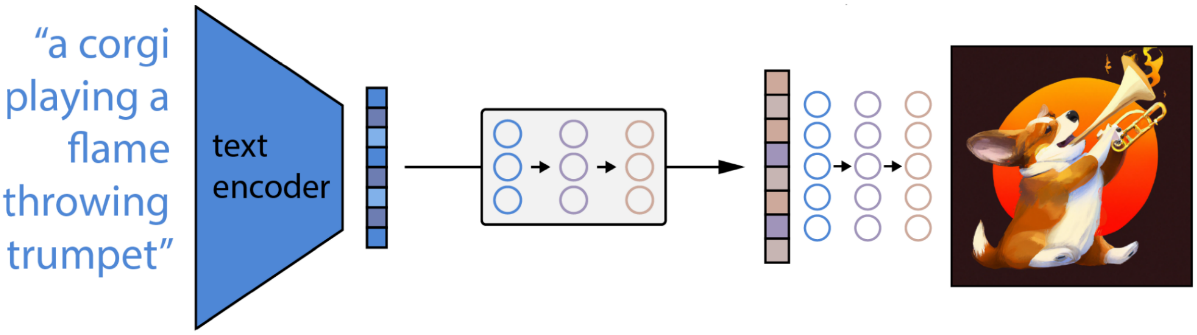
Function & Task

DALL-E, according to OpenAI's blog page, can create "anthropomorphized versions of animals and objects" in the generated pictures. Capable of transforming text that contains seemingly unrelated items within the same frame, DALL-E outputs several images with each run.

Specifically, DALL-E is able to anthropomorphize, or transfer human characteristics, onto inanimate objects. Additionally, there are certain key phrases that can prompt the model to design something in the form of another object: "in the form of" and "in the shape of" are good examples. Also, the model has the ability to adopt different fonts to various text prompts.

## DALL-E 2

2022[61]



## Pathways Language Model (PaLM)

2022

[60] https://muse.lighton.ai/home
[61] https://arxiv.org/pdf/2204.06125.pdf DALL-E 2 paper

Google research launched PaLM[62] to improve language models' performances by further scaling it. Most post-GPT models have similar improvement trends, according to the PaLM paper: scaling model depth and width, increasing training token input, intaking cleaner, and more diverse datasets all increase model capability without impinging upon computational cost.

However, PaLM takes a series of different approaches to enhance model performance. One of which is having a single model which generalizes across many domains. The pathways[63] system is the feature that will do this— the pathways system will support workloads anticipated for future ML research that is not present in state-of-the-art models.

1. Number of parameters: 540B
2. FLOPs: 2.56*10^24
3. TPUs: 6144 TPU v4
4. Cost for training: 17M via Google Cloud TPU

Function & Task

According to The Atlantic, PaLM was capable of performing "chain-of-thought prompting". This means that the model can break down the process in which it arrived at the answer if, for example, it were to be given a math problem. PaLM has an excellent performance on the multilingual domain. If the prompt was given in one language, it will answer the question in the prompter's language with an evaluation close to the benchmark for English. Furthermore, it is adept at source code generation either as text-to-code, code-to-code, or coding language translation.

## Concluding Thoughts

After looking at these models in respective functions & tasks, we find the state-of-the-art (SOTA) race is more or less based on raw compute price reduction, more efficient architecture (LSTM to transformers, from transformers to reformers, etc.), and optimized hardware (GPU to TPU). We therefore hypothesize a plausible correlation between the parameter count of ML models and their capabilities & complexity.

## References

https://openai.com/blog/ai-and-compute/
https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/
https://www.neuraldesigner.com/blog/perceptron-the-main-component-of-neural-networks#:~:text=Therefore%2C%20the%20total%20number%20of,the%20number%20of%20neurons'%20inputs.
https://medium.com/tebs-lab/how-to-classify-mnist-digits-with-different-neural-network-architectures-39c75a0f03e3
https://arxiv.org/pdf/2004.08900.pdf

[62] https://arxiv.org/pdf/2204.02311.pdf
[63] https://arxiv.org/pdf/2203.12533.pdf