CLIP, GPT, and Their Applications

Exploring Large-Scale Pre-Trained Models in Deep Learning

Nadav Kahlon — ID 213438575
Directed by Dr. Maya Herman

A seminar presented for the degree of
Bachelor of Science

Department of Mathematics and Computer Science
February 2023

Contents

1 Introduction

1.1 The World of AI

The concepts and innovations we'll discuss along this paper relate to the vast domain of *Artificial Intelligence* - commonly abbreviated as *AI*, so let us begin by first introducing this term and outlining key ideas surrounding it.

1.1.1 What is Artificial Intelligence?

All the way back in 1950 English mathematician Alan Turing raised the following question:

"Can machines think?"
(Turing and Haugeland [93])

A computer may be able to carry out any logic we may feed it with, but to many it is unknown if it can actually "think". What does it even mean? Turing proposed the famous *Turing test* to estimate the success of a "thinking machine". If a human interrogator is unable to differentiate between the behavior of a machine and that of a real person, then the machine is deemed to have successfully passed the test.

The term "Artificial Intelligence" was later coined by John McCarthy when he and his colleagues proposed to hold the first Artificial Intelligence conference in 1956, inspiring computer scientists and mathematicians to explore methods of enabling machines to solve the types of problems that have typically been considered the exclusive province of humans (McCarthy et al. [52]). They raised several directions that may be investigated, such as *self-improvement* and *language understanding*. They even suggested to study in what manner a set of theoretical *neurons* could be organized to develop concepts - an idea that evolved to form the astonishing tools described in this paper.

As vague as the question Turing raised may be, it is undeniable that research in the field of Artificial Intelligence has yielded significant results in the intervening years. Developments in search algorithms, machine learning techniques, and the integration of statistical analysis into comprehending the world have all considerably advanced the technological forefront, paving the way for numerous applications, as discussed by Borana [7].

1.1.2 Machine Learning

Let's focus our discussion on a certain field of research, known as *Machine Learning*. This term refers to a subfield of computer science and artificial intelligence that deals with algorithms capable of enhancing their proficiency through experience. In his work, Mitchell [57] established a formal definition of a computer program that learns from experience E within a given class of tasks T and evaluated by a performance measure P. This definition specifies that the program's performance on tasks in T, as evaluated by P, improves as a consequence of experience E.

Learning systems differ in their underlying methodology. They are usually divided into 3 paradigms, which have been extensively researched throughout the years:

1. **Supervised learning:** in which the task is to estimate a certain (maybe stochastic) function $f : \mathcal{X} \to \mathcal{Y}$ (usually defined over a set of continuous input features $\mathcal{X} = \mathbb{R}^n$), and the experience consists of known pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $y = f(x)$. Such pairs are often referred to as *labeled examples*. Classic examples of supervised learning tasks are:
    - *Regression:* in which the estimated function maps to real valued outputs.
    - *Classification* in which the estimated function maps to a finite set of classes $c_1, ..., c_m$.

2. **Unsupervised learning:** in which the task is to implicitly learn the underlying patterns and characteristics behind an unknown distribution $\mathcal{D}$ over a domain $\mathcal{X}$ from an experience set consisting of samples $x \in A$ drawn from it. Here, the samples are considered *unlabeled*, since no additional information is provided but the data points themselves. Classic examples of unsupervised learning tasks are:
    - *Distribution modeling:* in which we wish to approximate the probability distribution function of the underlying (unknown) distribution.
    - *Clustering:* in which we wish to recognize distinct groups of similar regions in the distribution.
    - *Dimensionality reduction:* in which we wish to identify and utilize correlations between the internal features of the samples and construct a function from $\mathcal{X}$ to a "lighter" domain $\mathcal{X}'$, which is as reversible and representative as possible over samples drawn from $\mathcal{D}$.
3. **Reinforcement learning:** in which the task is to learn a policy π mapping a set of states $\mathcal{S}$ to a set of actions $\mathcal{A}$, such that an agent operating in the corresponding environment of states following π receives maximal cumulative reward over its lifetime, based on some given reward metric $r : \mathcal{S} \to \mathbb{R}$ (Kaelbling et al. [36]). In this seminar we will be less concerned with this type of learning.

1.1.3 Computer Vision

One popular field of research in the wide domain of AI and machine learning in particular is *Computer Vision*. Szeliski [87] describes Computer Vision as the ability of a computer to perceive and interpret imagery the way a human may do. It's a non-trivial inverse task - in which we wish to describe and understand the contents of the world based solely on images sampled from it.

Due to the diverse characteristics of the physical world and the applicability of automatically understanding them via visual representations - there exists a wide variety of Computer Vision tasks, which have been thoroughly researched throughout the years. Several examples of such include:

- **Image classification:** just like the classic classification task described above, in image classification we wish to identify the class to which the objects in the images belong.
- **Semantic segmentation:** the task of identifying objects in an image (e.g. road, sky, car, building) and their exact location in the image by classifying *pixels* to the objects they cover.
- **Image generation / synthesis:** the task of approximating the distribution from which a given set of images were drawn (e.g. the distribution of all images of cats) and generating real-looking novel images synthetically drawn from it, or altering an existing image without deviating from the ("convincing") distribution.

1.1.4 Natural language Processing

Another domain that will constitute an integral part of our discussions is *Natural Language Processing*, commonly abbreviated as NLP. Chowdhary [12] defined NLP as a field of research and application that investigates how computers can be utilized to comprehend and process natural language (text or speech) to perform practical tasks, such as:

- Machine Translation: converting text from one language to another.
- Language Modeling: approximating the distribution of natural language input.
- Natural Language Inference: determining the logical relationship between a couple of sentences.

Language may be easy for humans to understand, but interpreting it using digital logic is far from a trivial task. It consists of discrete units (characters) combined together in variable-length sequences with a meaningful order based on conventional rules developed by humans for humans across millennia to express almost anything. Several tools have been developed to deal with language in program, as discussed in section 3.2.

1.2 Transfer Learning

1.2.1 Introduction

As its title suggests, this seminar surrounds *pre-trained models*. Let's explore the idea of *pre-training* through the introduction of *transfer learning*.

Ventura and Warnick [96] referred to "transfer learning" as an algorithmic process by which structure of knowledge derived from one learning problem, is used to enhance learning on a (different but) related problem. Lin and Jung [49] further formulated the idea by considering *tasks and domains*, as follows.

A domain $\mathcal{D}$ is defined as a pair, containing a feature space $\mathcal{X}$ and a probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$. A task is defined using an additional label space $\mathcal{Y}$ and an unknown mapping $f : \mathcal{X} \to \mathcal{Y}$. The task, denoted by $\mathcal{T} : \mathcal{X} \to \mathcal{Y}$, is learned based on a training dataset of ground truth samples - $\{x_i, y_i\}$ where $x_i \in \mathcal{X}$ and $y_i = f(x_i) \in \mathcal{Y}$.

Consider a pair of tasks - a source task $\mathcal{T}_S$ defined over a source domain $\mathcal{D} - S$, and a target task $\mathcal{T}_T$, defined over a target domain $\mathcal{D}_T$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. Transfer learning is defined as a method to help improve the learning of the mapping $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge collected in $\mathcal{D}_S$ and $\mathcal{T}_S$.

1.2.2 Transfer Through Fine-Tuning

In practice, there are several ways to apply transfer learning. One of the more well known method is known as *fine-tuning*.

All learning algorithms, each in its own way, solve *optimization problems*, of the form:

$$\min_{\mathbf{a}} \mathcal{L}(\mathbf{a}), \quad \text{s.t. } \mathbf{a} \text{ satisfies some predicate } P.$$

For example - $\mathbf{a}$ may denote the coefficient of a polynomial that approximates the unknown mapping f, while $\mathcal{L}$ denotes a measure for the distance between both functions on the known (x, y) samples. Minimizing this distance, is equivalent to approximating f.

Solving optimization problems is usually carried out by *iterative procedures*, which start from some feasible $\mathbf{a}$ (that may be randomly initialized), and slowly perform iterative updates of $\mathbf{a}$ to bring $\mathcal{L}(\mathbf{a})$ to its minimum further down the line. We'll see examples for such iterative optimization procedures in Section 2.2.

Transfer through fine-tuning is carried out by initializing this optimization procedure for the target task $\mathcal{T}_T$, with the solution $\mathbf{a}$ for the source task $\mathcal{T}_S$. This follows the hypothesis that if both tasks are related, the solution for the source task may already be close to the optimum of the target task, requiring less iterative updates to get there (than a randomly initialized process). This type of transfer methodology is referred to as fine-tuning, as we make fine iterative updates to the solution of $\mathcal{T}_S$ to make it a better fit for $\mathcal{T}_T$. In practice, only parts of the (probably composite) optimized variable may be transferred, while additional parts may be also added (and randomly initialized).

If the optimized variable $\mathbf{a}$ denotes the parameter set Θ of some hypothesis function template h_Θ that we wish to use to approximate the unknown mapping f (by finding Θ for which h_Θ is the closest to f), then we can think of transferring the solution Θ_S of the source task, as utilizing the computations carried out by the learned h_{Θ_S} to solve the original source task $\mathcal{T}_S$. These computations may be relevant and aid the solution of $\mathcal{T}_T$ as well, as both tasks are related.

1.2.3 Transfer Through Feature Extraction

Another methodology for application of transfer learning is through *feature extraction*. Learning algorithms may implicitly apply dimensionality reduction to their inputs in order to solve the corresponding task. This is carried out by learning a mapping $d : \mathcal{X} \to \mathcal{X}'$ from the source domain to a lighter domain $\mathcal{X}'$, known as a *feature space*, which is easier to interpret and process. d generates "condense" representations of the inputs, which usually encapsulate their higher-level components.

The extracted representations can be utilized by a learning algorithm to solve a target task $\mathcal{T}_T$ by fitting a lighter hypothesis $h' : \mathcal{X}' \to \mathcal{Y}$ on top of the features extracted by the solution for a related source task $\mathcal{T}_S$. As both tasks are related, the target task may also benefit from those high-level representative encodings of the inputs, which aided the solution of the source task. Since the encodings space, $\mathcal{X}'$, is lighter - fitting an hypothesis h' on top of it can be much less expensive.

It's worth noting that when specifically dealing with the task of classification for continuous inputs, the most straight-forward and popular method to utilize a pre-trained feature extractor d is by fitting a *linear probe* on the extracted features of the training examples - that is, fitting a linear classifier (see section 2.1) on top of these lighter feature vectors. Such classifier can easily be trained, with very marginal expenses.

1.2.4 Additional Terms

Transfer learning is not only useful for reducing training time and computational resources by utilizing knowledge obtained while fitting for a source task $\mathcal{T}_S$, but it also reduces data requirements. When the source task is general enough and the model fit to it is powerful enough, little to no additional training may be necessary to solve downstream target tasks. A powerful general model may have already implicitly learned most of the required functionality. This observation resulted in 3 popular transfer paradigms, know for the ease of their applicability:

- *Few-Shot Learning*, in which only a small set of known training examples are used for downstream training.
- *One-Shot Learning*, in which a single training instance (maybe per class) is used for downstream training.
- And the most astonishing *Zero-Shot Learning* - in which no additional training examples are provided for the target task and the functionality provided by the trained model is intelligently harnessed to solve the downstream task.

Pre-training is the process of fitting a learning model for a certain task, for the sole purpose of later utilizing it for relevant downstream tasks through transfer learning. Pre-training is usually done in large scale - using a very powerful model, large amount of resources (both time and computational), on large-scale datasets, and for a rather general task - for the exact goal of gaining powerful and general transfer capabilities using little to no task-specific training.

1.3 What's Ahead of Us

So what this is all about?

In this seminar we'll explore the idea of pre-training *deep learning* models for large scale tasks by investigating a pair of powerful models developed at OpenAI: **CLIP** (Radford et al. [73]) and **GPT** (Radford et al. [71, 72]; Brown et al. [9]). We'll dive into the methodology behind these systems, and thoroughly study their transfer capabilities by delving into selected applications that uniquely harness the functionality they provide.

We'll kick things off in chapter 2 by introducing the field of *deep learning* for the unfamiliar reader. We'll present the architectures and techniques used by the models discussed in this paper.

In chapter 3 we'll dive into the GPT family of language models. The discussion begins with a quick background check on the relevant technologies used in the field of Natural Language Processing, and continue with a thorough discussion of the 3 main models in the GPT family - highlighting key ideas regarding the innovative approaches behind them and analyzing their transfer capabilities.

In chapter 4 we explore CLIP - a powerful model that connects imagery and natural language developed at OpenAI. We'll study the unique approach of the authors behind CLIP and the wide range of transfer methodologies it enables. We'll also take a quick tour exploring the beautiful meaning behind some of CLIP's learned neurons.

In chapter 5 we'll further investigate the transfer capabilities of these powerful pre-trained models by delving into selected applications that harness the functionality they provide in a creative way, solving unexpected target tasks.

In chapter 6 we'll discuss the broader impacts that large-scale pretrained models such as CLIP and GPT may have - the social biases they may inherit from the datasets they are trained on, the malicious applications they may enable, and more.

Finally, in chapter 7 we'll conclude the seminar and discuss the future of the field.

2 Deep Learning in Depth

2.1 Introduction

As mentioned in the first chapter, in this seminar we'll focus on a certain branch of machine learning called *deep learning*. Deep learning is a subfield of machine learning surrounding computational models that are inspired by the structure and function of the brain, known as *artificial neural networks*.

The concept of artificial neural networks has been around since the 1940s - through Hebb's Hebbian learning theory (Hebb [27]) and Rosenblatt's perceptron (Rosenblatt [77]), but it wasn't until the advent of big data and powerful computing resources in the 21st century that deep learning began to take off. In 2012, a deep learning algorithm called AlexNet, proposed by Krizhevsky et al. [41], won the ImageNet Large Scale Visual Recognition Challenge (a famous image classification task we'll mention quite a lot in this paper, introduced by Deng et al. [16]), outperforming all previous computer vision algorithms by a significant margin. This breakthrough marked the beginning of a new era in artificial intelligence, as deep learning algorithms have since been applied to a wide range of problems, from image and speech recognition to natural language processing and beyond.

An artificial neural network is in fact - a function, defined over the source domain $\mathcal{X} = \mathbb{R}^n$. The term usually refers to the composition of (possibly-parameterised) functional blocks - also known as *layers*:

$$h = L_{output} \circ L_{hidden,N} \circ \cdots \circ L_{hidden,1} \circ L_{input}$$

The network (i.e. the function) receives its input through its *input layer*, processes it in its *hidden layers*, and finally supplies the output through the *output layer*. An overview of the process is given in Figure 1. The term *forward pass* refers to the process of feeding an input through these blocks and obtaining the corresponding output at the network's end. The scalar components of the inputs and outputs of the layers in the model are the ones known as the *"artificial neurons"* of the network. The values they posses may sometimes be referred to as *neuron activations*.

The most simple layer used in neural networks is a linear operator:

$$L : \mathbb{R}^n \to \mathbb{R}^m, \quad L(x) = Wx + b,$$

parameterized by:

- The *weights matrix* $W \in \mathbb{R}^{m \times n}$, connecting input vector components ("input neurons"), to output vector components ("output neurons").
- And the *biases vector* $b \in \mathbb{R}^m$ which enables unconditional shift of the output neurons

This is also known as a *fully connected layer* - as each input neuron has a direct impact on each output neuron (defined by the corresponding entry in the weights matrix).

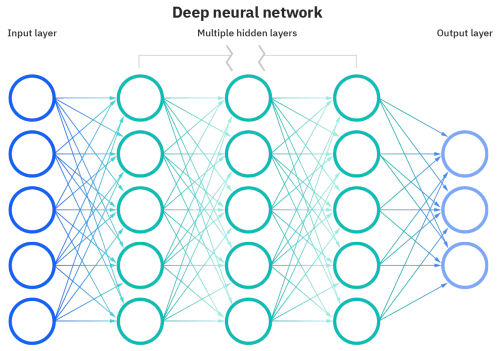


Figure 1: A schematic overview of a feed-forward deep neural network. Source: *IBM Cloud Learn Hub*.

Recall that we defined a neural network as a concatenation of such layers - but one might notice that a concatenation of linear operators is equivalent to a single linear operator! To enable stronger representational capabilities, *non-linearities* are added in-between the layers, that is - non-linear functions, known as *activation functions*. Examples of popular activation functions are:

- The sigmoid function:

$$\sigma : \mathbb{R} \to (0, 1), \quad \sigma(x) = \frac{1}{1 + e^{-x}}$$

- A rectified linear-unit:

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases}$$

A network consisting of only fully-connected layers and non-linearities in between them is known as a *fully-connected network*, a *multi layer perceptron (MLP)*, or a simple *feed-forward network*.

One activation function that's worth noting is the *softmax activation*, defined over a set of neurons $x \in \mathbb{R}^n$ as follows:

$$Softmax : \mathbb{R}^n \to (0,1)^n, \quad Softmax_i(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

The softmax activation produces a probability distribution, as all elements in it are between 0 and 1, and they all sum up to 1. It is used in the output layer of classification networks to produce a probabilistic estimate of the input's class. The runtime values of the inputs to a softmax layer are sometimes referred to as *logits*, and they quantify the un-normalized "belief" of the network in each of the classes. A network with a single linear input layer, no hidden layers, and a single softmax output layer, is usually called a *softmax classifier*, *a linear classifier*, or *logistic regression classifier* (although the last two may also refer to other types of classifier models).

2.2 Optimization

2.2.1 Loss

Consider the task of estimating a certain function $f : \mathbb{R}^n \to \mathbb{R}^m$ based on a set of examples $\{x_i, y_i = f(x_i)\}_{i=1}^N$. Now that we've established the basics of deep learning, we may wish to approximate f using a neural network - which we now know is just a long function parameterised by some set of parameters Θ defining an hypothesis function $h_\Theta : \mathbb{R}^n \to \mathbb{R}^m$. In order to ensure h_Θ approximates f, we can require its output on the known examples to be as close as possible to the known evaluations of f:

$$h_\Theta(x_i) \approx y_i = f(x_i)$$

But how can we formally state this requirement? One way is through a *loss function* - defining the distance between the predicted output and the ground-truth for a given input:

$$\bar{\mathcal{L}} : \mathbb{R}^m \times \mathbb{R}^m \to [0, \infty)$$

The task can now be formulated as an optimization problem, in which we wish to find the combination of parameters Θ that minimizes the average loss of the hypothesis on all known samples:

$$\min_\Theta \mathcal{L}, \quad \mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \bar{\mathcal{L}}(h_\Theta(x_i), y_i) \tag{1}$$

To conclude the subsection, let's list a couple of popular loss functions used for different machine learning tasks:

- Mean-Squared Error loss (MSE), used for regression tasks:

$$\bar{\mathcal{L}}_{MSE}(h_\Theta(x), y) = ||h_\Theta(x) - y||^2$$

- Cross-entropy loss (CE), used for classification tasks (the output of the model is assumed to be a probability distribution of classes, and the ground truth vector is assumed to be a distribution with 100% probability for the input's actual class):

$$\bar{\mathcal{L}}_{CE}(h_\Theta(x), y) = -\sum_{j=1}^{m} y_j \log(h_{\Theta,j}(x))$$

  Sometimes, the exponentiation of the cross-entropy loss is used as a measure of performance - which is referred to as *perplexity*.

2.2.2 Gradient-Based Optimization

So we wish to solve (1) - find the combinations of parameters Θ that makes h_Θ as close as possible to the ground truth f on the known samples (on average). In the world of deep learning, this is usually accomplished by *gradient based optimization algorithms*. Assuming that the network and the loss measure are differentiable, we can find the gradient of the error with respect to the network's parameters. This is the direction of steepest ascent of the loss, and the opposite is the direction of steepest descent of the loss. Since we wish to minimize the loss, we can construct an algorithm that iteratively steps Θ in that direction:

$$\Theta \leftarrow \Theta - \eta \frac{\partial \mathcal{L}}{\partial \Theta}$$

The multiplier η is called the *learning rate*, and it controls the magnitude of the steps - the algorithm's speed and ability to perform finer updates. This iterative scheme is known as the basic *gradient descent* optimization algorithm.

The gradient themselves are evaluated using a process known as *backpropagation* - which iteratively applies the chain rule of derivation through the atomic (differentiable) calculations carried out by the various layers in the network:

$$\frac{\partial \mathcal{L}}{\partial L_i} = \frac{\partial L_{i+1}}{\partial L_i} \cdot \frac{\partial \mathcal{L}}{\partial L_{i+1}}$$

As the gradients are calculated from the end of the network (where the loss is directly evaluated) and propagate backwards to its beginning via the chain rule, this process is sometime referred to as a *backward pass* (in contrast to the forward pass done when evaluating the network's output).

However, when the dataset is large, evaluating all samples in it and calculating the gradients with respects to them just to perform a single update step is way too expensive. Thus, the update steps are usually carried out using a subset of the samples each time:

$$B \subset \{1, ..., N\}, \quad \mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} \bar{\mathcal{L}}(h_\Theta(x_i), y_i)$$

$$\Theta \leftarrow \Theta - \eta \frac{\partial \mathcal{L}_B}{\partial \Theta}$$

These subsets of instances are usually referred to as *minibatches* or just *batches*. The resulting algorithm is known as the *Stochastic Gradient-Descent* algorithm (SGD), as in expectation the gradient of the loss with respect to a minibatch is the same as the actual gradient of the loss with respect to the entire dataset (assuming all examples are drawn from the same source distribution). More complicated update scheme are usually Incorporated to enhance optimization - such as the ones used by the Adam optimizer (Kingma and Ba [39]) or the momentum mechanism for SGD (Qian [70]).

2.2.3 Supporting Optimization

Optimization by its nature is strongly affected by the behavior of the gradients. However, these tend to have an exponential behavior as feeding inputs or backpropagating gradients through linear layers has the affect of multiplication by weight matrices. These may stack up to form an unstable repeated multiplications, causing the phenomena of *exploding gradients* - where gradients become exponentially large (and gradient-based optimization diverges), or *vanishing gradients* - where gradients become exponentially small (and gradient-based optimization gets stuck).

Several methods to aid optimization and stabilize gradient flow have been developed since the recent popularization of deep neural networks. One simple approach to tackle this is to use more intelligent random initialization schemes of the model's parameters. Xavier initialization scheme (Glorot and Bengio [23]), for instance, draws the initial weights from a specific distribution, that in theory should ensure balanced gradient flow across the network.

More powerful tools are *normalization layers* - which normalize the output of a network's layer to have certain (learned) mean and standard deviation, ensuring stable behavior of neuron activations. A couple of popular normalization schemes are *batch normalization* (Ioffe and Szegedy [32]) - which normalizes the same neuron in the layer over a batch of training samples, and *layer normalization* (Ba et al. [3]) - which normalizes all neurons in the layer over the same training sample.

Another very powerful technique that was first introduced by He et al. [26] in their successful research *"Deep residual learning for image recognition"* (2016) is based on *residual connections*. Formally, a residual connection applied on a layer L adds the layer's input to its output before continuing to the next layer:

$$L'(x) = x + L(x)$$

By enabling the direct propagation of information from the earlier layers to the later layers, residual connections allow the gradients to flow more efficiently through the network. This has been shown to significantly improve the overall training of neural networks, enabling stable networks with deeper architectures.

2.3 Convolutional Networks

Convolutional layers are the fundamental building block of most neural networks used for image classification and other computer vision tasks. In these layers, a set of filters or kernels are used to perform convolution operations on the input data, which essentially extracts *local features* from the input image (in a linear manner). The output of a convolutional layer is a feature map that highlights important information from the input, which is then passed to the next layer for further processing. Convolutional layers play a crucial role in capturing and extracting spatial hierarchies of features from an image, allowing *Convolutional Neural Networks* (ConvNets) that incorporate them to learn complex patterns and make better predictions about the contents of an image.

Formally, the operation of a convolutional layer on an input image $I \in \mathbb{R}^{H \times W \times C}$ ($H \times W$ is the dimension of the image and C is the number of channels in it) with a learned kernel $K \in \mathbb{R}^{H' \times W' \times C' \times C}$ and a learned bias vector $b \in \mathbb{R}^{C'}$ is the C'-channels image defined as follows:

$$(I * K)_{i,j,c'} = \sum_{i'=0}^{H'-1} \sum_{j'=0}^{W'-1} \sum_{c=0}^{C-1} K_{i',j',c',c} \cdot I_{i+i',j+j',c} + b_{c'}$$

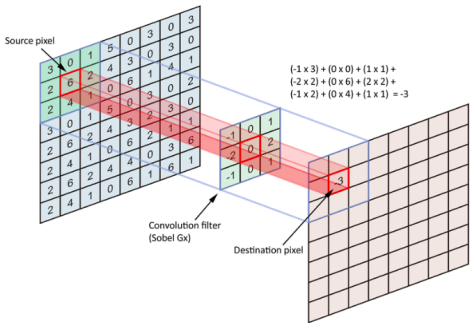


Figure 2: An illustration of the convolution operation. Source: *Towards Data Science*.

The process is visualized in Figure 2. This is different from a fully connected layer, in that only the $H' \times W'$ input neurons in the neighborhood of an output neuron has an effect on him, and the linear kernel weights are reused for all output neurons. This reflects the locality of the convolutional layer - locality that characterizes visual data.

A key feature of convolutional layer is padding, which allows the network to preserve the spatial resolution of the input data by adding zeros around the edge of the input. Stride is another feature of ConvNets, which controls the step size of the filter as it slides over the input. A larger stride results in a smaller output size. Finally, down-sampling pooling operations are often used in ConvNets to reduce the spatial resolution of the data. Max-pooling, for instance takes the maximum activation in a neighborhood of pixels, which make the network invariant to small translations in the input.

We'll finish our discussion by mentioning the *ResNet family*. ResNet is a series of deep convolutional network architectures with integrated residual connection and increasing capacity, denoted by the number of convolutional layers they employ: ResNet-18, ResNet-34, ResNet-50 (the most popular

of the five), ResNet-101 and ResNet-152. It was first presented in the same paper that originally proposed residual connections for deep networks: "Deep Residual Learning for Image Recognition" (He et al. [26]). The models in the family significantly advanced the state of the art on the ImageNet classification task and proved useful for many other applications. Even today, 8 years since they were first proposed, ResNets are still popular among the computer vision community and serve as a powerful baseline architecture for many applications.

2.4 Transformers

2.4.1 The Classic Transformer

The Transformer architecture is a type of neural network that was introduced in 2017 by Vaswani et al. [94] in their groundbreaking paper "Attention is All You Need". The Transformer architecture is particularly well-suited for processing sequential data, such as natural language, and has advanced the state-of-the-art on several NLP tasks. The Transformer architecture uses *self-attention* mechanisms to weight the importance of different elements in the sequence and aggregate information from different parts of it to make predictions. This architecture has proven to be highly effective, allowing models to process long sequences efficiently, while maintaining the ability to capture long-range dependencies.

In this section, we will focus our discussion specifically on the Transformer's encoder unit, as it captures all key features behind the Transformer, and is relevant for the models later discussed in this paper. An overview of its operation is given in Figure 3.

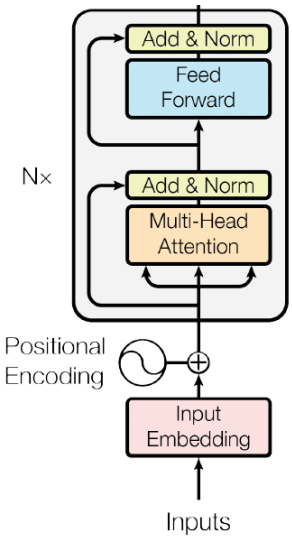


Figure 3: A schematic overview of the Transformer encoder unit architecture. Source: Vaswani et al. [94].

At the core of the Transformer is the *self-attention mechanism*. Consider a sequence of n elements $x_1, ..., x_n \in \mathbb{R}^m$. Each element x_i is used to generate a *key* $k_i \in \mathbb{R}^{d_k}$, a *query* $q_i \in \mathbb{R}^{d_k}$, and a *value* $v_i \in \mathbb{R}^{d_v}$. Element i attends to the value associated with element j (v_j) based on the similarity between the query it generated (q_i) and the key associated with the target element (k_j). Softmax activation (with the logits normalized) is used to weight the n value vectors element i may attend to, resulting in single vector capturing the attended knowledge:

$$Attention_i(x_1, ..., x_n) = Softmax(\frac{q_i^T k_1}{\sqrt{d_k}}, ..., \frac{q_i^T k_n}{\sqrt{d_k}})^T \cdot V$$

The layer as a whole can then be represented in matrix form as follows (where X, K, Q, V are the matrices whose rows are x_i, k_i, q_i, v_i, respectively):

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

In practice, Q, K, V are generated using simple learned linear mappings. *Multiheaded self-attention* layer consists of several different instances of such attention units, whose output is concatenated together and then linearly mapped back to a lower-dimensional space (using a shared learned projection matrix), resulting in a new sequence that is passed forward to the rest of the network. Multiheaded attention is depicted in Figure 4.

It's worth noting that for some applications, *masked self-attention* must instead by employed - a mechanism in which each element can only attend to the elements that precede it in the sequence. This is carried out by discarding the key-query pairs associated with illegal combinations (where the key follows the query).

As depicted in Figure 3, the Transformer encoder is a concatenation of identical layers, each composed of 2 sub-layers:

- Multiheaded self attention - as previously described.
- A fully connected network, which is applied to each element separately and identically. This consists of two simple linear transformations with a *ReLU* activation in between.

Residual connections and layer-normalization are employed in-between the sub-layers to assist optimization, as discussed in subsection 2.2.3.

You may have noticed that the Transformer block does not consider the *order* of the elements in the sequence. To incorporate this information into the network's representation of the data, *positional encodings* are added to the embedded representations of each element in the input sequence and helps the model understand the relative position of each element with respect to the others. One common technique for positional encoding is to add sine and cosine functions of different frequencies to the vector representations of the input elements. These functions have a unique pattern for each position in the sequence, allowing the model to differentiate between elements based on their position. By incorporating positional encoding, Transformer models proved to effectively handle sequences of varying lengths and capture the consecutive relationships between elements in a sequence.

Finally, it's worth mentioning that the dimension of the in-between vector representation of the elements in the sequence is usually kept constant across the network. This is usually referred to as the *width* (or "hidden size") of the Transformer, while the number of Transformer blocks is referred to as its *depth*.

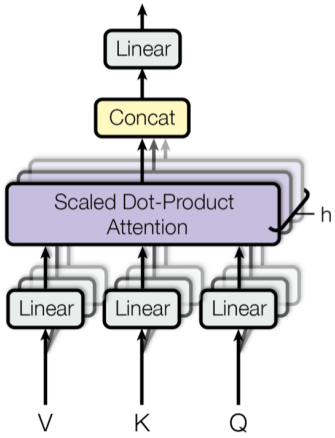


Figure 4: A schematic overview of multiheaded attention, as part of the Transformer architecture. Source: Vaswani et al. [94].

2.4.2 Vision Transformer

Since the introduction of the Transformer in 2017, it was proved to be a powerful and scalable architecture for many natural language processing tasks. Inspired by its success, Dosovitskiy et al. [19] experimented with applying a standard Transformer directly to images with the fewest possible modifications, hoping to achieve similar success in computer vision. Their idea is to split the image into equally-sized patches, flatten and linearly map them onto a d-dimensional space using a shared (learned) projection matrix, and feed them to a Transformer as if they were just a long sequence of elements - or "visual words". An additional learned d-dimensional element is added, which will carry the global representation of the image by the end of the Transformer (in a similar manner to the BERT architecture proposed by Devlin et al. [18]). They named the architecture *Vision Transformer* (ViT).

Dosovitskiy et al. [19] experimented with various methods to adapt the positional embedding mechanism to utilize the 2-dimensional nature of the ordering of the "elements", but came to the conclusion that when enough data is present, any sophisticated mechanism is no better than a simple mechanism based on *learned positional embeddings* associated with the uniformly extracted patches in the (equally sized) training images. A Vision Transformer can thus handle images of varying resolutions, by interpolating between these learned embeddings in the 2D plane - as a varying number of patches may now be extracted.

The authors also provide a suite of baseline ViT architectures of varying sizes (similar to the ResNet family proposed by He et al. [26]), including:

- ViT-Base: having hidden size of 768 components, 12 layers, and 12 attention heads per layer.

- ViT-Large: having hidden size of 1024 components, 24 layers, and 16 attention heads per layer.
- ViT-Huge: having hidden size of 1280 components, 32 layers, and 16 attention heads per layer.

ViT architectures are denoted by the capacity of the corresponding Transformer, and the size of the equally-extracted patches they are fed with. For example, a ViT-L/16 refers to a ViT-Large Transformer with 16×16 input patches.

3 GPT

3.1 Introduction

The first model we'll overview in this seminar, which is actually a series of models, is the **GPT** family. GPT (Generative Pre-trained Transformer) entered our lives at 2018, with the publication of the famous paper *"Improving Language Understanding by Generative Pre-Training"*, by Radford et al. [71]. The paper introduced the model informally known as *GPT-1* (which was not actually named yet and was simply referred to as "Transformer LM"), that initiated a continuously-evolving fruitful research surrounding the capabilities of large-scale pre-trained Transformer-based *language models* - a concept we'll thoroughly discuss in this chapter.

The GPT family, consisting of several powerful language models, is now known as one of the greatest Natural Language Processing systems humanity has ever possessed. As Inamdar [31] discusses, the astonishing abilities associated with the models in the series have revolutionized the world of Artificial Intelligence.

In this chapter we'll dive into the main 3 models in the GPT family - study their structure and discuss their capabilities. GPT-1 (Radford et al. [71]) - which we'll discuss in section 3.3 - is in fact the least known model in the family. The more popular GPT-2 (Radford et al. [72]) was later published in 2019 and widely adopted by the research community as the team released the complete version of the the model for free by the end of 2019. We'll discuss GPT-2 in section 3.4. Finally, in section 3.5 we'll overview the extensive research surrounding GPT-3 (Brown et al. [9]) - the largest and most powerful model in the series, published in 2020.

But first, let us begin by gathering the relevant background in Natural Language Processing and Language Modeling, so we can better understand what GPT is about.

3.2 NLP: Technological Background

3.2.1 Tokenization

Tokenization is a crucial step in Natural Language Processing, that involves dividing unstructured textual data into smaller meaningful units. Following Webster and Kit [101], these units are named *tokens*. They are the building block of structured natural language input sequences, and are treated as individual elements in NLP tasks. They can be phrases, words, subwords, characters, or even sub-character bytes. The process of tokenization involves several important decisions, such as whether to split words based on whitespaces or punctuations, and whether to consider numbers and special characters as separate tokens. Tokenization has been extensively studied and several algorithms have been proposed for it (Mielke et al. [54]).

Let's overview the practical process of tokenization, following Pai [64]. Initially, a large corpus of text is scanned and meaningful units - tokens - are extracted from it based on the *tokenization algorithm*, and collected into a *vocabulary*. When new text input comes - it is divided into the tokens it is composed of using the tokenization algorithm, and transformed into a mere sequence of references into the vocabulary. This structured sequence is then passed along as the input for the rest of the system.

Usually, special tokens are added to the vocabulary, such as the `<SOS>` (start-of-sequence) and `<EOS>` (end-of-sequence) tokens marking the boundaries of the input, the `<UNK>` token to which words not present in the vocabulary at test time may be mapped, and many others - depending on the application and tokenization algorithm.

Nowdays, central tokenization techniques usually involve subword segmentation algorithms, as discussed by Mielke et al. [54]. The breakthrough for subword tokenization was the use of *Byte-Pair-Encoding* (BPE; Gage [21]) by Sennrich et al. [82] for machine translation. To construct a vocabulary from a given text corpus, BPE starts with a poor vocabulary containing only the possible characters.

It then iteratively replaces the most frequent successive pair of tokens with a new symbol representing that pair, until the vocabulary reaches a certain size. At test time, input text can be segmented in a similar manner, based on the order in which the tokens were added during training. BPE bridges the gap between character and word level language modeling: it provides the "high-level" token representation of whole words and phrases, while preventing the need for the `<UNK>` token which models unknown lingual units ambiguously - as all possible characters are present in the vocabulary, enabling the indirect modeling of every character string.

3.2.2 Word Embedding

After tokenization, natural language input is provided as a sequence of tokens - which are seemingly meaningless discrete units. *Word embedding* is a technique that maps a continuous vector $e(t) \in \mathbb{R}^n$ to each token t in the vocabulary, "embedding" the token in a continuous vector space, as described by Almeida and Xexéo [2].

These latent embeddings encapsulate the tokens' lingual meanings in a representative way that's easier for a learning model to interpret. The mapping from tokens to embeddings defines the first layer of NLP networks - the *word embedding layer*, whose parameters are the embeddings themselves. These are usually learned via similar gradient-based methods in conjunction with the network itself. Word embeddings can also be transferred from open-source pre-trained systems - such as GLoVe (Pennington et al. [69]) or Word2Vec (Mikolov et al. [55]).

3.2.3 Language Modeling

As briefly mentioned in subsection 1.1.4, *language modeling* is the task of assessing the likelihood of a natural language input. The goal of language modeling is to build a statistical model that can assign probabilities to sequences of tokens, such that the probability of a given sequence is a measure of the likelihood of that sequence appearing in natural language text. Language models are widely used in many NLP applications, such as speech recognition, machine translation, and text generation.

A language model's quality is determined by its ability to accurately predict the likelihood of sequences of words in the target language, and is usually achieved by training a learning algorithm on a (very) large corpus of text data to predict the conditional probability of a token appearing, given the history of tokens that precede it:

$$P(t_n|t_1, ..., t_{n-1}) \tag{2}$$

A sequence's probability can then be evaluated using the chain rule of probabilities:

$$P(t_1, ..., t_n) = \prod_{i=1}^{n} P(t_i|t_1, ..., t_{i-1})$$

(2) can be modeled with a probabilistic classifier, classifying sequences (such as $t_1, ..., t_{n-1}$) based on the token that may follow them (t_n). It's worth noting that in most practical applications, due to computational constraints, a token's history is clipped to a fixed size c, which is usually referred to as the model's *context size*. Practically, the following approximation is made:

$$P(t_n|t_1, ..., t_{n-1}) \approx P(t_n|t_{n-c}, ..., t_{n-1})$$

Probably the most important application for which language models can be utilized is *text generation*. Language models can generate a viable sequence of tokens one by one based on the probabilistic distribution of the next token collected from the output of the model. It is for that reason that language models are sometimes referred to as *autoregressive models*, as they regress text from the same distribution of their input. There several methodologies to do so:

- A straightforward approach, known as *greedy text generation*, iteratively appends the token that has the highest probability to follow the current generated sequence. However, this approach is prone to repetitive and unvaried outputs as it only considers a single token at each step
- *Beam search* is a more sophisticated approach where the model maintains a set of the top k possibilities of sequential continuations at each step, thereby generating more accurate and diverse continuations. This approach helps in avoiding repetition and improving overall quality of the generated text.
- *Top-k random sampling* is yet another approach, where the model randomly samples the top k possibilities at each step, instead of only considering the highest probability outputs. This approach helps in introducing more randomness and creativity in the generated text.

The ending of a generated sequence can be indicated by the model when it generates the special token `<EOS>`.

3.2.4 Classic NLP Tasks

To finish our introductory background on NLP, let us overview several NLP tasks and datasets we will encounter in the remainder of the chapter.

Natural language inference (sometimes referred to as *NLI* or "recognizing text entailment"), is the task of judging the logical relationship between a given pair of sentences. It's a classification task, mapping their relationship to one of three possible categories: `entailment`, `contradiction`, or `neutral`. For example:

```
["At the other end of Pennsylvania Avenue, people began to line up for a White House
  tour.", "People formed a line at the end of Pennsylvania Avenue."] -> entailment
```

Reading comprehension (sometimes referred to as "question answering") is a task that tests a model's ability to capture long range-dependencies and extract relevant information from large amounts of data. A sample of that task consists of 3 elements: a rather long document, a question asking about a specific thing mentioned in that document, and a set of possible answers. It's a classification task, that requires a model to assign the correct answer. *RACE* (Lai et al. [43]) is a popular dataset for this task, containing passages and associated questions collected from middle and high school exams. Another popular dataset is *CoQA* (Reddy et al. [75]), which integrated follow-up questions in a conversation-like manner, assessing the model's ability to rely on the historical context of a chat.

Commonsense reasoning is another task that demands both single and multi-sentence reasoning. It's a classification task, that usually requires the model to select the correct choice for an omitted part of a short story. Popular datasets are: Story Cloze Test (Mostafazadeh et al. [60]) - which requires the model to pick an ending to a story from 2 possible options, the Children's Book Test (CBT; Hill et al. [29]) - where the objective is to identify the correct choice among 10 possible options for a word that has been omitted, and the LAMBADA dataset (Paperno et al. [65]) in which the task is to anticipate the concluding word of sentences that demand a context of at least 50 tokens for a human to make an accurate prediction.

The Winograd Schemas Challenge is another popular commonsense reasoning dataset, introduced by Levesque et al. [45] in 2012, which evaluates a system's proficiency in common-sense reasoning by gauging its capacity to *resolve ambiguities* in text by selecting the correct missing word in a sentence:

```
"The firemen arrived after the police because they were coming from so far away.
              Who came from far away?  [The firemen / The police]"
                          Answer:  "The firemen".
```

The harder *WinoGrande* challenge (Sakaguchi et al. [80]) consists of adversarially picked instances of the task - which are intentionally easier for humans than they are to machines.

Semantic similarity, also known as paraphrase detection, is a classification task that entails predicting whether two sentences have the same meaning or not. This task presents difficulties in identifying the rewording of concepts, comprehending negation, and managing syntactic ambiguity. An example is:

```
"Obama speaks to the media in Illinois" <-> "The president greets the press in
                                Chicago"
```

The *CoLA (Corpus of Linguistic Acceptability, Warstadt et al. [100]) dataset* evaluates whether a sentence is grammatically correct or not and examines the intrinsic linguistic predisposition of trained models:

```
                    "She voted for herself." -> Correct
"Books were sent to each other by the students." -> Wrong
```

The *SST-2 (Stanford Sentiment Treebank)* dataset (Socher et al. [84]) is a binary classification task, predicting the sentiment of a sentence (positive / negative):

```
"Yet the act is still charming here" -> Positive
"contains no wit, only labored gags" -> Negative
```

The *CNN and Daily Mail* dataset tests a generative model's ability to extract the relevant information from a large text, and construct sufficient summaries. It consists of large articles, and highlight paragraphs containing the most important information extracted from them.

To evaluate the amount of *knowledge* embedded in the network's parameters, *factoid-style* questions may be used, also referred to as *closed book question answering*. The task usually requires the model to answer short knowledge questions such as:

```
"Who wrote the book the origin of species?  " -> "Charles Darwin"
```

when provided with no context whatsoever. Various datasets have been devised for this task, including the Natural Questions dataset (Kwiatkowski et al. [42]), TriviaQA dataset (Joshi et al. [34]), and WebQuestions (WebQs; Berant et al. [5]).

Finally, the classic task of *translation* from one language to another is also one of the central task in NLP, so much that a special term for this algorithmic task was coined - *Machine Translation (MT)*. Success of machine translation is usually measured by the *BLEU* score (bilingual evaluation understudy), described by Papineni et al. [66]. Several datasets are used for this task, measuring translation capabilities between a variety of languages.

3.3 GPT-1: Language Modeling for Fine-Tuning

3.3.1 Approach

In this section we'll overview the first model in the GPT series, informally referred to as *GPT-1* (Generative Pretrained Transformer). It was introduced in 2018 in the paper *"Improving Language Understanding by Generative Pre-Training"* by Radford et al. [71].

The authors wished to diminish the need for large annotated datasets to solve NLP tasks using deep learning techniques. Such datasets are difficult to gather, both financially and logistically. The idea was to train a system that's able to extract linguistic and semantic information *directly from large-scaled unlabeled text corpora*, that can be easily transferred to various down-stream tasks with minimal (supervised) adaptation.

Even when abundant supervision is accessible, unsupervised learning of powerful representations can result in substantial performance enhancements to any application, when considering the generality they offer by being trained on a diverse large-scale dataset. The widespread deployment of pretrained word embeddings in many NLP systems is a good example of that. While these methods mainly transfer word-level information, Radford et al. aimed to capture higher-level semantics of natural language input.

However, leveraging more than just word-level information from unannotated text raises a couple of challenges. First, what unsupervised optimization objective should we use during pretraining? And once pretraining is complete, what's the best way to transfer the learned representations to the target task?

As for the former challenge - *language modeling objective* was chosen to learn the initial parameters of the neural network model. To fit a language model in the manner described in subsection 3.2.3, we are only concerned with what token may follow what history of tokens. This kind of information is readily available in any text corpora, making language modeling an appealing unsupervised pretraining objective. And as for the latter challenge - once a language model is trained, its parameters can be *fine-tuned* (see section 1.2) to the specific target task using the relevant supervised objective.

Radford et al. employed a multi-layer Transformer for the language model, due to the excellent performance it has demonstrated in the past on numerous NLP tasks. As they state, the Transformer is ideal for sequences of varying lengths, and the attention mechanism enables a more organized memory mechanism to manage long-term dependencies in text.

The Transformer language model works as follows: natural language input from the known corpus is fed into the Transformer. The i'th element of the last layer's output is then used in a probabilistic softmax classifier that compares it (by dot-product) to the original token embeddings, to predict the $i+1$'th token (see eq. (2)) . Masked attention (described in section 2.4) is employed to ensure none of the tokens "cheat" by attending to its successors. Cross-entropy loss is used as the optimization objective for the language model:

$$\mathcal{L}_{LM} = \sum_i \log P(t_i | t_{i-c}, ..., t_{i-1})$$

The 'Generative" part in **G**PT marks the generative nature of this approach, previously discussed in subsection 3.2.3.

For training the language model, the BooksCorpus dataset (Zhu et al. [105]) was utilized. This dataset consists of over 7,000 distinctive unpublished books from various genres, including Adventure, Fantasy, and Romance. Importantly, it comprises of extensive sequences of continuous text, enabling the generative model to learn to capture long-range dependencies.

Fine tuning the model for down-stream classification tasks is in fact not trivial. For tasks such as question answering or textual entailment (later discussed), the inputs may be structured in a specific way. For example, a triplet `[document, question, answer]` is a possible structure for the task of classifying whether an answer to a question is correct given a certain document in context.

Since the pretrained model was trained on continuous sequences of text, modifications are necessary to apply it to these structured inputs. Radford et al. employed a traversal-style approach (Rocktäschel et al. [76]), where the structured inputs are converted into an ordered sequence delimited by *special new tokens* (whose embeddings are randomly initialized), so it could be processed by the sequential model. These input conversions allow us to minimize changes to the architecture across tasks. Figure 5 illustrates the process.

To classify a given sequence $x = (x_1, ..., x_m)$ to a certain class y, the last element in the Transformer's final layer's output $h_m^{(l)}$ - associated with a new special token `<Extract>` - is then fed into a learned linear classifier:

$$P(y|x_1, ..., x_m) = Softmax(W_y \cdot h_m^{(l)})$$

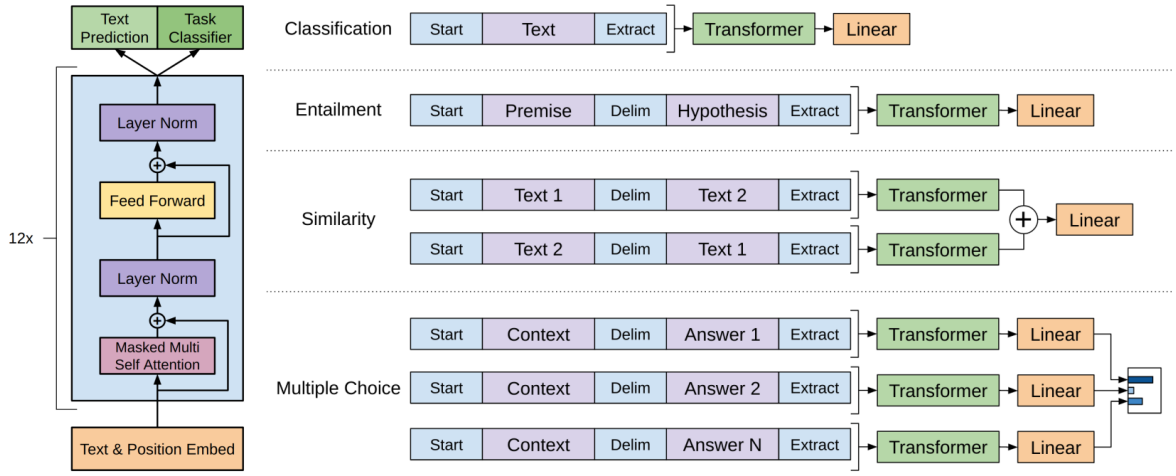


Figure 5: Architecture and training objectives used for GPT-1 (left), and the input transformations employed for fine-tuning on different tasks (right). All inputs are converted into token sequences. Source: Radford et al. [71].

and a cross entropy loss is optimized:

$$\mathcal{L}_T = \sum_{(x,y)} \log P(y|x_1, ..., x_m)$$

In practice, the authors discovered that incorporating language modeling as an additional objective during fine-tuning aided in learning by enhancing the generalization of the supervised model and speeding up convergence, resulting in the hybrid optimization target:

$$\mathcal{L}_H = \mathcal{L}_{LM} + \lambda \cdot \mathcal{L}_T$$

On the more technical note, it's worth mentioning some of the details of training GPT-1. A 12-layer Transformer with 12 attention heads per layer and hidden size of 768 dimensions was employed. The inner fully connected layers were 3072-dimensions wide. In total, GPT-1 had around 117M trainable parameters. For pretraining, the Adam optimizer was utilized, training for 100 epochs with batches of 64 randomly sampled sequences containing 512 tokens (and a context window of the same size). Byte Pair Encoding (BPE) was used for tokenization, on a vocabulary of around 40,000 tokens. Fine tuning was quick, and 3 epochs were mostly enough.

3.3.2 Performance

The research team tested the transfer capabilities of the Transformer language model they trained on a variety of classic supervised NLP tasks. In this subsection we'll overview the way the different experiments were carried out, and the performance of GPT-1 when fine-tuned for such down-stream tasks, relative to the state-of-the-art.

Natural language inference is tested and carried out in the sequential model by concatenating both the premise and the hypothesis, with a new special delimiter token in between. GPT-1 surpassed the current state-of-the-art baselines on four of the five datasets tested, achieving improvements of up to 5.8% accuracy (on the QNLI dataset, introduced by Wang et al. [97]). This showcases the model's capacity to reason more effectively over multiple sentences and handle linguistic ambiguity.

For question answering, the context document and the question and concatenated to each of the possible answers. The model processes each of the resulting sequences individually and then utilizes a softmax layer to normalize the outputs, producing an output distribution over the feasible answers. Using this scheme, GPT-1 managed to outperform the previous state state-of-the-art on the RACE dataset (Lai et al. [43]) by 5.7% accuracy.

To test commonsense reasoning, a similar multi-choice framework was applied. Here, GPT-1 significantly advanced the state-of-the-art on the Story Cloze dataset Mostafazadeh et al. [60] by 8.9% accuracy.

Finally, for some classic classification tasks: GPT-1 managed to achieve substantial improvement of 10.4% accuracy on the CoLA dataset (Warstadt et al. [99]), and was competitive with previous best-performing systems on SST-2 (Socher et al. [84]).

3.3.3 Additional Analysis

In addition to evaluating GPT-1 on typical NLP challenges, the team analyzed generative pretrained Transformers from various other perspectives, to gain a deeper comprehension of their features, advantages, and limitations.

To test whether the model's later layers may overfit to the task of language modeling during pre-training, making them sub-optimal for down-stream tasks, Radford et al. also studied the effect of transferring a varying number of layers from unsupervised pre-training to the supervised target task. The performance of the method on the MultiNLI dataset (Williams et al. [102]) and RACE (Lai et al. [43]) in relation to the number of layers transferred is presented in Figure 6 (left). As expected, transferring embeddings enhances performance, and each Transformer layer offers additional benefits, resulting in an improvement of up to 9% on MultiNLI when all layers are transferred. This finding suggests that each pre-trained model layer contains general valuable features for addressing any target task.

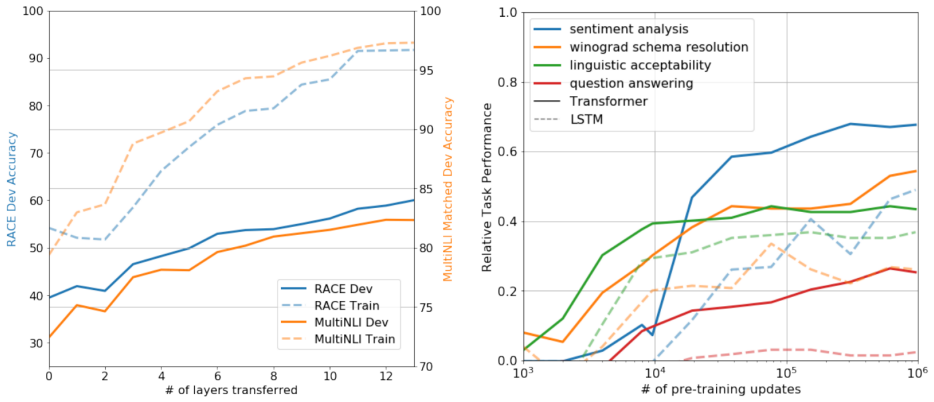


Figure 6: (left) Performance on RACE (Lai et al. [43]) and MULTINLP (Williams et al. [102]), as a function of the number of transferred GPT-1 layers. (right) GPT-1 zero-shot performance improves through unsupervised training. Source: Radford et al. [71].

They also wished to understand the effect that the Transformer has on these astonishing results. They compared it with a powerful 2048-unit, single-layer LSTM (which is another neural network architecture we've not discussed, introduced by Hochreiter and Schmidhuber [30]). The results indicated an average score decrease of 5.6% when the LSTM is used in place of the Transformer, hinting that the Transformer is a powerful architecture for these settings.

To analyze the effectiveness of pre-training on the large scale corpora prior to fitting for down-stream tasks, the team also assessed the performance of the same Transformer architecture when trained only on the relevant supervised target task, excluding pre-training. The findings demonstrate that the absence of pre-training adversely affects performance across all tasks, causing a 14.8% average decline in accuracy in comparison to the complete model.

Foreshadowing OpenAI's next big research on language models, which will be shortly discussed, the authors also quickly tested the capabilities of GPT-1 when transferred to down-stream tasks with *no additional supervised training* using a sequence of heuristic approaches that utilize the fundamental generative model to accomplish the tasks. Results are presented in Figure 6 (right). It's worth noticing how the performance in these *zero-shot* settings progressively improves during pre-training, indicating that language modeling facilitates the acquisition of diverse task-related features. The authors hypothesized that the underlying generative model learns to execute many of the assessed tasks to assist it with the global goal of language modeling.

3.4 GPT-2: Language Modeling for Zero-Shot Transfer

3.4.1 Approach

The second model in the GPT family, now formally referred to as *GPT-2*, was published in 2019 with the successful paper *"Language Models are Unsupervised Multitask Learners"* by Radford et al. [72]. This research continued the line of work presented in the previous section - using a pre-trained Transformer-based language model for a diverse variety of down-stream tasks, while taking a different transferring approach. The model was very popular among the research community, as it is the most powerful language model OpenAI ever released for free.

The authors' main hypothesis when developing GPT-2 was that single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Although the traditional method of training systems has been effective in making progress on narrow experts, the often erratic behavior of these models on the diversity and variety of possible inputs highlights some of the shortcomings of this approach. They mention that multitask learning (Caruana [10]) may be a promising framework for improving general performance, but with current approaches, it may need as much training data as for the traditional method. They conclude that exploring additional setups for performing multitask learning is necessary to improve general performance.

The authors follow their previous research surrounding GPT-1, which proved that language models are able perform specific tasks when there is no supervised data available (Figure 6 (right)). To create a general system that can perform many tasks, the system should not only condition on the input (i.e. model $P(y|x)$), but also on the task to be performed (i.e. model $P(y|x, \mathcal{T})$). McCann et al. [51] demonstrate that language can be used to express tasks ($\mathcal{T}$), inputs (x), and outputs (y) altogether as sequences of symbols in a flexible manner. Language modeling has the potential to learn these tasks without explicit supervision, as the supervised objective is the same as the unsupervised objective, when only considering these subset of sequences that correspond to the relevant tasks. Thus, we may roughly state that the global minimum of the unsupervised objective is also the global minimum of the supervised objective. From this, we can draw the hypothesis that a language model with enough capacity will start to learn how to solve specific tasks presented in naturally-occurring language sequences to better predict them, even without task-specific supervision.

Their strategy involved gathering a vast and diverse dataset to acquire a wide range of natural language demonstrations of tasks across different domains and contexts. However, after experimentation, they opted to prioritize data quality. They achieved that by using a new web scraping method that prioritized curated and filtered web pages. To avoid high manual filtering costs, they focused on web pages that were curated by humans and had at least three karma on the social media platform *Reddit* - meaning that enough users found them interesting. The resulting dataset, they named *WebText*,

contains the text subset of 45 million links. For training their new model, the team only used a preliminary version of WebText, that excludes links created after December 2017. The dataset contains over 8 million documents, equivalent to 40 GB of text (after de-duplication and heuristic-based cleaning).

The dataset is tokenized using Byte Pair Encoding, following its advantages mentioned in subsection 3.2.1. It is restricted from merging across character categories for any token sequence (with the exception of the space character) to avoid many versions of the same word (for example - `"happy"`, `"happy?"`, and `"happy!"`).

Following its success for GPT-1, Radford et al. [72] continued to base their GPT-2 architectures on the Transformer. This time, the team trained *a series* of 4 exponentially-increasing language models:

1. With 117M parameters: a 12-layers 768-dimensions-wide Transformer (copy of GPT-1).
2. With 345M parameters: a 24-layers 1024-dimensions-wide Transformer.
3. With 762M parameters: a 36-layers 1280-dimensions-wide Transformer.
4. With 1.54B parameters: a 48-layers 1600-dimensions-wide Transformer.

With the latter usually referred to as *GPT-2*.

Most technical details regarding the architectures and their training follow from the original GPT-1, with a few changes. Several slight modifications to the Transformer itself were applied, and a larger vocabulary of around 50,000 tokens is used, as opposed to the 40,000 tokens vocabulary of GPT-1. The context window was also expanded from 512 tokens to 1024, and training batches were enlarged from 64 samples to 512 samples each.

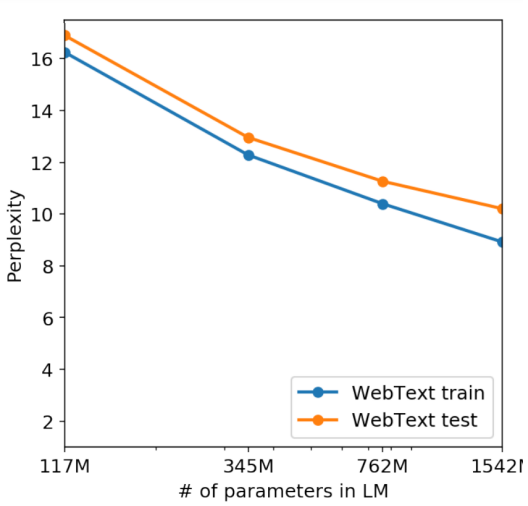


Figure 7: Model size effect on GPT-2 pretraining performance. Source: Radford et al. [72].

It's worth noting that as the training dataset of WebText was automatically extracted from the internet, some of the datasets on which it is evaluated down the line may accidentally overlap with it - as they are also publicly available on the web. Radford et al. were concerned with this "data leakage", and performed an extensive analysis of the effect it may have on the model's reported performance. The analysis showed that having some shared data between the WebText training data and certain evaluation datasets may lead to slightly better reported results. However, in most cases, the overlap between standard training and test sets was already similar to the overlap between the partitions of most evaluation datasets. Figure 7 illustrates that performance on both the training and test sets of WebText improves together as model size increases, indicating that GPT-2 is still far from overfitting the training set itself.

3.4.2 Performance

To test their hypothesis regarding the task-solving capabilities of mere language models, Radford et al. developed a suite of mechanisms to represent the wide variety of specific NLP tasks directly in natural language, enabling them to harness the functionality provided by the language model to develop systems that solve these tasks in a pure zero-shot manner. In this subsection we'll review the performance GPT-2 achieved in these settings on a variety of tasks.

First, Radford et al. examined how well the GPT-2 models performed at zero-shot domain transfer in their primary task of language modeling. WebText LMs may face significant out-of-distribution testing for many of these datasets, as these have been aggressively pre-processed and standardized, imposing artifacts of many kinds. The results showed that WebText LMs transfer effectively to these new domains, outperforming state-of-the-art on 7 out of 8 datasets in a zero-shot setting, especially on small datasets as well as datasets that test for long-term dependencies.

It is worth mentioning that in the LAMBADA language modeling dataset, the model is required to choose the best single word ending for a sentence. Investigations of GPT-2's errors revealed that while most of its predictions were valid continuations of the sentence, they were not valid final words. This indicates that the language model is not utilizing the useful constraint that the chosen word must be the final one. By adding a stop-word filter as an approximation of this constraint, the accuracy of the model increased from 52.66% to 63.24%, resulting in a 4% improvement in the overall state of the art on this task.

With an improvement of 7% on the Winograd Schemas Challenge, GPT-2 has achieved a new state-of-the-art accuracy of 70.70%, which proves its ability to tackle commonsense-reasoning tasks. Moreover, GPT-2's performance is comparable to or surpasses that of three out of four baseline systems for the CoQA dataset, even without utilizing the 127,000+ question-answer pairs manually collected for training those baselines. It is impressive that GPT-2 achieves this performance without any supervised training. However, upon inspecting its answers and errors, it appears that GPT-2 frequently employs rudimentary retrieval-based heuristics, such as selecting a name from the document as the answer to a "who" question.

To evaluate GPT-2's summarization capability, Radford et al. also conducted tests on the CNN and Daily Mail datasets. They prompted GPT-2 to summarize the article by appending the text "TL;DR:" and generating 100 tokens using Top-k random sampling with k = 2 to encourage more abstractive summaries and reduce repetition. They considered the first three sentences from these 100 generated tokens as the summary. Although the generated text qualitatively resembled a summary, the authors state that it often centered on recent content from the article and may include specific details. Removing the task hint reduces GPT-2's performance by 6.4 points on the aggregate metric, indicating the model's ability to *exhibit task-specific behavior through natural language guidance.*

The researchers also tested if the language model has learned to translate between languages. To encourage GPT-2 to perform translation, they prompted the generative model for continuations of sequences of the form:

```
"French: 'Je m'appelle Nadav'. English: "
```

Hoping for the correct continuation:

```
"'My name is Nadav'"
```

On the WMT-14 English-French test set (Bojar et al. [6]), GPT-2 achieved a BLEU score slightly worse than a word-by-word substitution. However, on the WMT-14 French-English test set, GPT-2 performed significantly better by leveraging its strong generative English model. The results surprised the researchers, as they had intentionally removed non-English webpages from the training data.

Finally, to evaluate the amount of knowledge embedded in the language model's learned parameters, the model was tested on the Natural Questions dataset (Kwiatkowski et al. [42]). Examples of question-answer pairs were included in the generative model's context to help it recognize the short-answer style of the dataset. However, even with the added guidance, the smallest model falls short of a basic baseline that returns the most frequent answer for each question type (such as `"who"`, `"what"`, and `"where"` questions). The largest LM of GPT-2 does answer 5.3 times more questions correctly, which suggests that neural system performance on this type of task is directly associated with model capacity. All in

all, GPT-2 still lags behind open domain question answering systems, which use a hybrid approach of information retrieval and extractive document question answering to achieve 30-50% accuracy.

As a conclusion, Radford et al. state that their findings may offer a potential explanation for why pre-training techniques have been successful in downstream NLP tasks, as we observed in the previous section when fine-tuning GPT-1. It appears that, in certain cases, these pre-training techniques are capable of learning how to perform tasks directly without requiring additional supervised adaptation or modification.

They still mention that despite GPT-2's ability to compete with supervised baselines, its zero-shot performance is impractical for most tasks. Even on commonly evaluated tasks, such as question answering and translation, their language models only perform better than trivial baselines when having enough capacity. Additionally, they state that there are likely many practical tasks where GPT-2's performance is still no better than random. The possibility of fine-tuning GPT-2 to down-stream tasks (similar to GPT-1) is mentioned, but left as future research direction.

3.5 GPT-3: Learning at Test Time

3.5.1 Approach

The last model in the GPT family we'll discuss is the famous *GPT-3*. GPT-3 made a huge buzz in social media, due to the outstanding capabilities this enormous model presented - hinting, for the first time ever, the possibility for true *Artificial General Intelligence* (Goertzel [24]). The research surrounding the development of the model was published in 2020, under the title "Language Models are Few-Shot Learners", carried out by the research team of Brown et al. [9] at OpenAI. The model itself was never released to the public, and the trained parameters are considered proprietary for OpenAI.

Continuing GPT's line of work, the authors wished to take the idea of pretrained language models for down-stream task to the next level. The authors neglected the fine-tuning approach, arguing that fine tuning for very limited tasks distributions may lead to poor generalization. They also stated that humans - marking a possible level of achievable intelligence - can learn most language tasks without the need for huge supervised datasets, which enables them to easily switch between various tasks and skills, such as performing a math operation during a long conversation. They wished for NLP systems to also exhibit this same level of fluidity and generality.

In their study, they investigated the potential of using *meta-learning* to tackle these challenges. Meta-learning for language models involves the acquisition of a diverse range of skills and pattern recognition abilities during the language modeling pretraining phase, which it can then leverage during the inference phase to quickly recognize and adapt to a specific task. This creates 2 phases of learning: the unsupervised pre-training step - in which gradient updates are performed to optimize the network for language modeling, and the novel *in-context learning* of the fixed generative model as it reads examples of the down-stream task at test time. For example, we may quickly "teach" the model analogies by feeding it with the following context:

```
Find word analogies:
hammer : nail :: comb : => hair
white : black :: up : => down
mansion : shack :: yacht : => dinghy
```

And append the following line:

```
light : heavy :: short : =>
```

hoping it would *learn* from the first three examples that it needs to generate the correct continuation: `long`. The novelty is this approach lies in the fact that *no gradient updates* are performed during in-context learning - as Figure 8 depicts. Inference must directly be carried out by the parameters

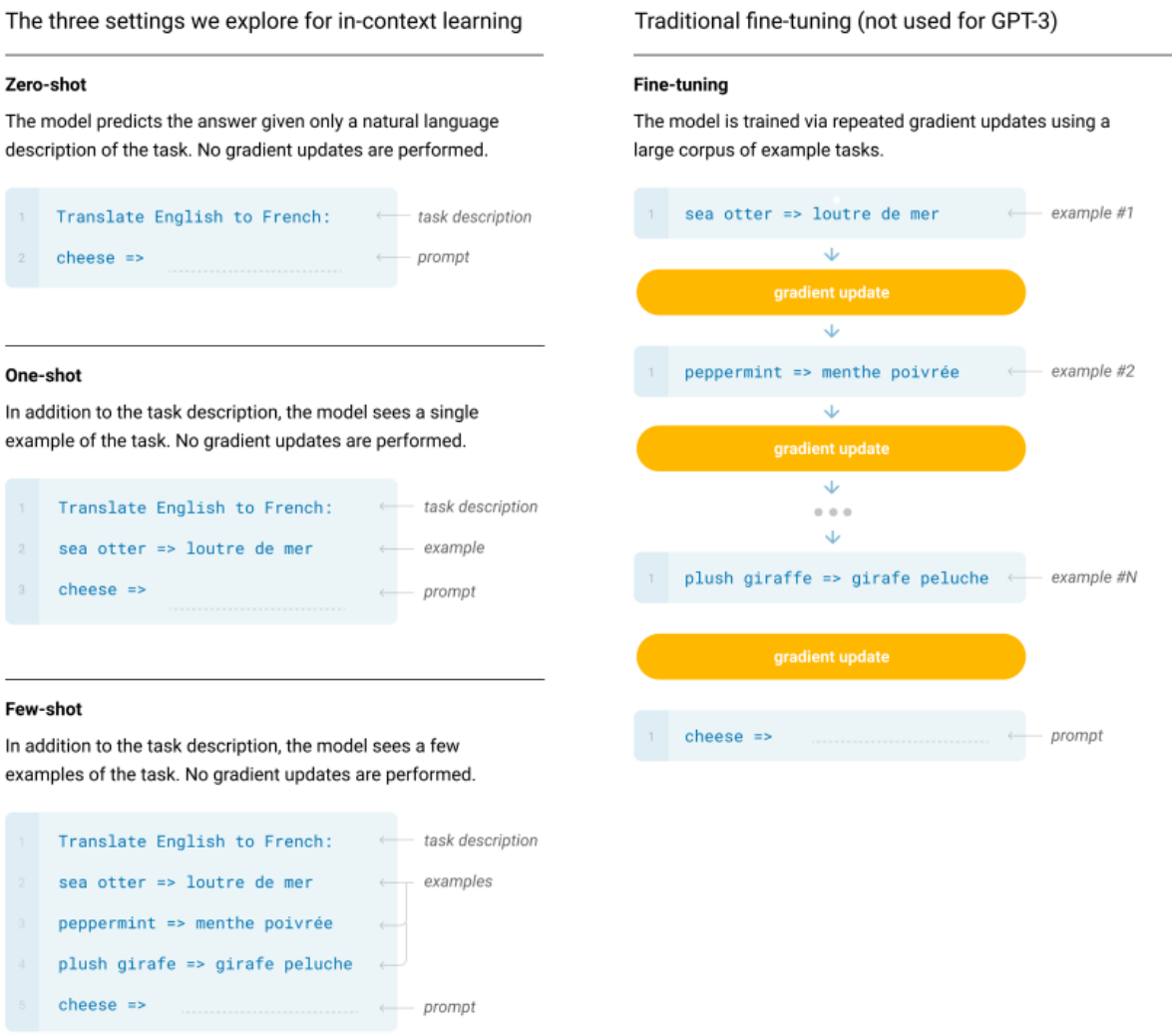


Figure 8: (left) In-context learning: the transfer methodologies behind GPT-3. (right) Traditional transfer through fine-tuning. Source: Brown et al. [9].

of the network. A pretrained model that's able to learn at test time is the authors' "vision" behind GPT-3.

As the series of GPT-2 architectures proved (Figure 7), increasing the model's size leads to a smooth improvement in log loss, which is a good indicator for downstream tasks. As in-context learning encompasses many skills and tasks within the model's parameters, it's possible that it too will improve with larger models. Brown et al. takes this hypothesis to the extreme, training a series of 8 models with increasing size, the largest of which reaching the unbelievable magnitude of 175.0B parameters. Table 1 summarizes the GPT-3 series of Transformer architectures (also note the increased batch size used for training, as opposed to the 512-sized batch of GPT-2 or the 64-sized batch of GPT-1).

They evaluated GPT-3 for each task under three conditions:

- "Few-shot learning" or in-context learning, where the model is given a few task demonstrations at inference time as conditioning, but no weight updates are allowed.

- "One-shot learning" where the model is allowed only one demonstration in addition to natural language task description, which closely matches how some tasks are communicated to humans.
- "Zero-shot learning" where no demonstrations are allowed, and the model is only given a natural language instruction describing the task. This is the most challenging setting and is closest to how humans perform tasks in most cases.

To evaluate few-shot learning, the authors use a process that involves randomly selecting K examples from a task's training set as conditioning, separated by 1 or 2 newlines (depending on the task). They mention that the value of K was usually better when larger, and was selected by testing the model on validation sets. For tasks that require selecting one correct completion from multiple options, they provide K examples of context plus correct completion, followed by one example of context only. Then they compare the likelihood of each completion using the language model probability estimations. For tasks that involve binary classification, they decided to assign more semantically meaningful names, such as `"True"` or `"False"`, instead of `0` or `1`, and treat the task as a multiple-choice question. For tasks with free-form completion, they utilize beam search for text generation.

Model Name	$n_{\texttt{params}}$	$n_{\texttt{layers}}$	$d_{\texttt{model}}$	$n_{\texttt{heads}}$	Batch Size
GPT-3 Small	125M	12	768	12	0.5M
GPT-3 Medium	350M	24	1024	16	0.5M
GPT-3 Large	760M	24	1536	16	0.5M
GPT-3 XL	1.3B	24	2048	24	1M
GPT-3 2.7B	2.7B	32	2560	32	1M
GPT-3 6.7B	6.7B	32	4096	32	2M
GPT-3 13B	13.0B	40	5140	40	2M
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	3.2M

Table 1: Specifications of the models in the GPT-3 series. Source: Brown et al. [9].

The technical methodologies for training GPT-3 are similar to GPT-2, with increased model sizes, longer training, larger dataset (described in the next subsection), and slight modifications to the underlying Transformer architecture. The same BPE tokenizer used for GPT-2 is reused for GPT-3.

3.5.2 Dataset

Brown et al. based their training dataset for GPT-3 on the *Common Crawl archive*. Common Crawl is an extensive web data repository that provides free and open access to web pages and associated metadata. It employs a web crawler to collect the data, which starts with a seed set of URLs and recursively follows links to discover new web pages. The crawler collects every website it encounters. Once collected, the data is processed, cleaned, compressed and added to the Common Crawl archive.

The training dataset included some of Common Crawl's partitions, covering 2016 to 2019, which amounted to 45TB of compressed plain text before filtering and 570GB after filtering, or approximately 400 billion tokens. Brown et al. found that Common Crawl's unfiltered or lightly filtered versions had lower quality compared to more curated datasets. To improve the average quality of the data, they included additional high-quality reference corpora in the training mix. During training, those datasets were sampled more frequently, accepting a small amount of overfitting in exchange for better quality of data.

3.5.3 Performance

Like GPT-2, GPT-3 was first tested on the task of language modeling. In the zero shot settings it achieved 76% accuracy on LAMBADA (Paperno et al. [65]) - around %8 better than the previous

SOTA, while reaching new heights with 86.4% when utilizing few-shot-in-context learning. GPT-3 also performs at 83.2% in the zero-shot setting and 87.7% in the few-shot setting (using 70 examples in context) when tested on the StoryCloze dataset (Mostafazadeh et al. [60]). Despite being 4.1% lower than state-of-the-art systems that have been fine-tuned, these results still represent an improvement of approximately 10% compared to previous zero-shot outcomes.

To test the factual knowledge the model absorbed in its parameters during pre-training, GPT-3 is also tested for closed-book question answering. The one-shot performance of GPT-3 on TriviaQA (Joshi et al. [34]) matches that of a SOTA open-domain QA system, which employs both fine-tuning and a learned retrieval mechanism (Lewis et al. [46]). It achieves another 3.2% in few-shot. On both WebQs (Berant et al. [5]) and Natural Questions (Kwiatkowski et al. [42]) there is a significant improvement from zero-shot to few-shot. This suggests that the format of the samples in these sets might be different from what GPT-3 is used to, although it successfully adapts to it in few-shot. On WebQs the few-shot system approaches SOTA, while it is less competitive for the Natural Questions fine-grained knowledge dataset. There is also a smooth trend of improvement with model capacity - suggesting that the size of the model is directly associated with absorbed knowledge.

Brown et al. [9] mention that although GPT-3's zero-shot Machine Translation capabilities were impressive, it still underperforms some of the more recent unsupervised systems. They also note that there is a noticeable imbalance in translation ability between different language directions. Specifically, GPT-3 performs much better when translating into English than when translating from English.

To test its common-sense reasoning skills, GPT-3's performance on the Winograd Schema Challenge (Levesque et al. [45]) was measured. Its zero-shot, one-shot, and few-shot settings achieved strong results of 88.3%, 89.7%, and 88.6%, respectively, without clear evidence of in-context learning. Although these results are slightly below state-of-the-art, they are still impressive. On the more challenging WinoGrande dataset (Sakaguchi et al. [80]), GPT-3's in-context learning led to some improvement, but it underperformed relative to the state-of-the-art.

GPT-3's reading comprehension ability was evaluated on 5 different datasets that use abstractive, multiple choice, and span-based answer formats in both dialog and single question settings. The results show a wide variation in GPT-3's performance across these datasets, which suggests that it has varying capabilities with different answer formats. Overall, the authors state that GPT-3's performance is comparable to initial baselines and early results for this NLP task.

GPT-3's natural language inference ability was also evaluated using the RTE dataset (Dagan et al. [14]) and the adversarially mined ANLI dataset (Nie et al. [61]). Only the largest version of GPT-3 performed better than random on both datasets, but still insufficiently. Overall, the authors mention that GPT-3 seems to struggle in tasks involving the comparison of two sentences or snippets.

GPT-3 was evaluated in the few-shot or zero- and one-shot setting using various tasks to test its ability to perform on-the-fly computational reasoning, recognize novel patterns, and adapt to unusual tasks. It performed well on simple arithmetic problems in natural language, with performance decreasing as the number of digits increases. GPT-3 was also able to learn novel symbolic manipulations from a few examples and can recognize and use nonexistent words it's taught in its context. Additionally, it managed to generate synthetic news articles that were difficult for humans to distinguish from human-written ones. However, it still struggled with correcting English grammar and understanding the contextual nuances of language.

3.5.4 Additional remarks

In this final subsection we cover a few more remarks and observations Brown et al. raised during their extensive research developing the powerful GPT-3 model.

It's worth comparing the different settings to each other, to learn more about the significance of the different parts in play. First, through all tasks, performance tends to smoothly improve with the increasing size of the model. This is summarized in Figure 9. Moreover, it also seems like the number of examples provided in context for few-shot learning has a direct effect on performance too, as Figure 10 depicts.

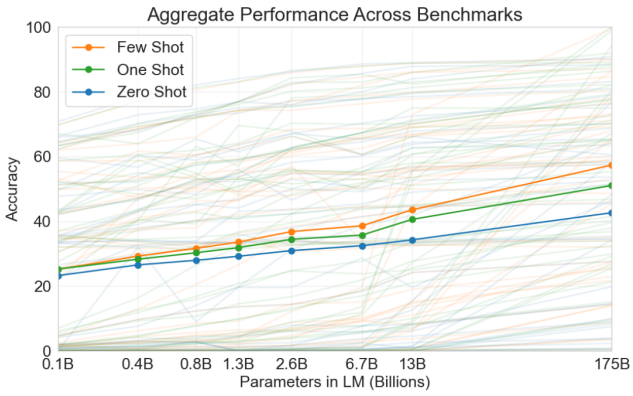


Figure 9: Effect of model size and in-context transfer methodology on GPT-3 average performance. Lighter lines represent performance on specific benchmarks, which tends to be rather noisy. Source: Brown et al. [9].

Although it has made remarkable progress in the field of natural language processing, Brown et al. mention that GPT-3 still faces several limitations. One of the main drawbacks is related to text synthesis, where the overall quality is high but GPT-3 samples still exhibit semantic repetition at the document level, loss of coherence over long passages, contradictions, and non-sequitur sentences or paragraphs. Additionally, GPT-3 has difficulty with "common sense physics", performs poorly on some "comparison" tasks, and struggles with reading comprehension in some cases.

Another limitation of GPT-3 the authors bring to our attention is that the current objective weights every token equally, lacking a notion of what is more important to predict. Moreover, large pretrained language models are not grounded in other domains of experience, such as video or real-world physical interaction, and therefore lack a lot of context about the world. They also mention that the model sees more text during pre-training than a human sees in their lifetime, which raises ambiguity about whether few-shot learning actually learns new tasks "from scratch" at inference time - like humans - or simply recognizes and identifies tasks it has learned during training.

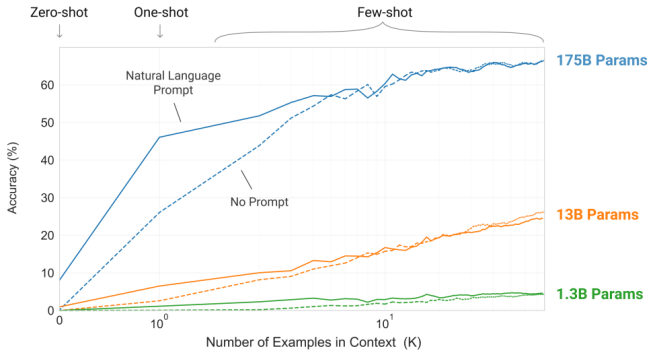


Figure 10: GPT-3 performance improves with the number of examples in its context: the model learns at test time. Source: Brown et al. [9].

Finally, the authors also emphasize that as with most deep learning systems, the decisions GPT-3 makes are not easily interpretable, not well-calibrated in their predictions, and retain biases from the data they have been trained on.

As a final remark, one promising direction mentioned towards the ending the paper is learning the objective function directly from human instructors through *reinforcement learning techniques*. By doing so, it may be possible to address some of the limitations of GPT-3 and improve its overall performance. A follow-up paper by Ouyang et al. [63] investigates this approach, and introduces the *InstructGPT* model - which forms the basis for the famous chat platform of OpenAI, known as *ChatGPT*. Unfortunately, these are beyond the scope of this seminar.

4 CLIP

4.1 Introduction

The second model we'll overview in this seminar is **CLIP** (Contrastive Language-Image Pre-training). CLIP was introduced in February 2021 by a group of researchers at OpenAI, in their highly quoted paper *Learning Transferable Visual Models From Natural Language Supervision*, by Radford et al. [73]. As we will see, CLIP is an extremely powerful and flexible tool, connecting visual concepts with natural language. The trained models were publicly published by the team - providing researchers and practitioners all around the world with the ability to develop their own applications on top of it.

In section 4.2 we'll study the unique suite of concepts upon which the development of CLIP was based. In section 4.3 we'll overview the models that surround CLIP, and the process of their training. In section 4.4 we'll explore CLIP's strengths and weaknesses through transfer to a wide variety of classification tasks. Finally, in section 4.5 we'll gain a deeper understanding of CLIP by looking at astonishing patterns that appear when examining the activations of the neurons in the trained model.

4.2 Approach

Radford et al. [73] were encouraged by the success of pre-training models directly from raw text for NLP tasks using large collections of web text portrayed by the GPT family, and sought to explore how scalable pre-training methods that learn directly from web text could lead to a similar breakthrough in the field of computer vision.

They considered the possibility of using natural language to connect imagery with a broad range of visual concepts. They emphasized the value of natural language as a training signal and its ability to express an unlimited set of visual concepts through its generality. They also appreciated the scalability of this approach - as it enables models to learn passively from the vast amounts of text available on the internet. Furthermore, they point out that using natural language supervision not only enables the learning of general image representations, but also connects those representations to language - a powerful tool for representing almost every possible concept, which may enable flexible language-based zero-shot transfer methodologies.further down the line.

The concept of connecting imagery and language has been around for ac couple of decades, as seen in earlier papers such as Mori et al. [59]. However, the performance of these approaches was limited. They often relied on primitive representations such as bags-of-words (Joulin et al. [35]) or n-grams (Li et al. [47]). Others were based on varying-quality datasets (such as the YFCC100M dataset by Thomee et al. [91], were too application specific (such as the work of Zhang et al. [104] for medical imaging), or used small datasets in comparison to modern computer vision datasets (such as the MS COCO dataset by Lin et al. [48]). In contrast, successful approaches in NLP relied on very large scale training. The authors aim to close this performance gap by exploring a new approach to learn visual representations directly from large-scale internet-based datasets.

In order to achieve their objective, the researchers created a new dataset, named WIT (WebImageText), consisting of 400 million pairs of images and text. WIT was constructed by collecting a large number of image and text pairs from various publicly available sources on the internet using web-crawling techniques. The dataset was balanced across multiple variables, such as object types and image quality, and covers a broad range of visual concepts. The images were obtained from public image-sharing platforms, such as Flickr and Instagram, and were filtered to remove duplicates and low-quality images. The text was obtained from associated metadata, such as image captions and tags, as well as surrounding textual content, such as article headlines and social media posts. The WIT dataset was designed to be diverse and representative of the full range of visual concepts, with unsupervised and supervised methods used for data collection and filtering.

The authors of the CLIP model found that predicting captions, as previously done in the VirTex architecture (Desai and Johnson [17]), was suboptimal. Instead, they improved the model's performance by exploring the use of a *contrastive objective*, as described in previous work by Zhang et al. [104] in the field of medical imaging. Specifically, Given a batch of N $(image, text)$ pairs:

$$(I_1, T_1), ..., (I_n, T_n) \in \mathcal{I} \times \mathcal{T}$$

CLIP is trained to predict which (image, text) pairs actually occurred by jointly training an image encoder and text encoder:

$$ImageEncoder : \mathcal{I} \to \mathbb{R}^d, \quad TestEncoder : \mathcal{T} \to \mathbb{R}^d$$

to maximize the cosine similarity of the embeddings of the n real pairs in the batch while minimizing the cosine similarity of the embeddings of the $n^2 - n$ incorrect pairings:

$$logits = e^{\tau} \times CosineSimilarity(ImageEncoder(I_1, ..., I_n), TextEncoder(T_1, ..., T_n))$$
$$(logits \in \mathbb{R}^{n \times n})$$

(with a learned temperature parameter τ), and optimize a symmetric cross-entropy loss (Wang et al. [98]) over these similarity scores:

$$\mathcal{L}_{SCE} = \mathcal{L}_{CE}(Softmax(logits)) + \mathcal{L}_{CE}(Softmax(logits^T))$$

4.3 Models and Training

Two different architectures are utilized for the vision encoders in the different CLIP models:

- A set of ResNets with varying scale, including ResNet-50 and ResNet-101, as well as three additional models denoted as RN50x4, RN50x16, and RN50x64, respectively. These models follow the EfficientNet-style scaling approach (Tan and Le [89]) and use about 4x, 16x, and 64x the compute of ResNet-50, with a few slight modifications to the original ResNet architecture.
- Vision Transformers, which include a ViT-B/32, a ViT-B/16, and a ViT-L/14. To further enhance the performance of the ViT-L/14 model, it is also pre-trained at a higher image resolution of 336 pixels (and subsequently denoted ViT-L/14@336px), similar to the approach of FixRes (Touvron et al. [92]). The latter model is the one usually associated with the name *CLIP* (unless otherwise stated), which the authors found to perform best.

The text encoder used in CLIP is a variant of the GPT-2 Transformer architecture (Radford et al. [72]). The base model is a 12-layer, 63M-parameter model with a width of 512 and 8 attention heads. The text is tokenized using byte pair encoding (BPE) with a vocabulary size of 49,152 (Sennrich et al., 2015), and is capped at a maximum sequence length of 76 for computational efficiency. The width of the text encoder is scaled proportionally to the increase in width of the ResNet, but the depth is not scaled, as the researchers found that CLIP's performance was not very sensitive to the capacity of the text encoder. The input sequence is prepended with the start-of-sequence `[SOS]` token and appended with the end-of-sequence `[EOS]` token, and the output vector of the highest layer of the Transformer at the `[EOS]` token is used as the feature representation vector of the text.

Finally, the representation vectors extracted from both vision and text encoders are linearly mapped using learned projection matrices to a joint embedding space - where they are compared against one another.

Quickly delving into the technical details, we mention that CLIP is trained from scratch using a minibatch size of 32,768 and the Adam optimizer (Kingma and Ba [39]). Additionally, as previously mentioned, the temperature parameter which controls the range of the logits in the softmax, τ, is

directly optimized as well. Various other techniques were utilized to support the training process. The largest ResNet model, RN50x64, required 18 days of training on 592 V100 GPUs, while the largest Vision Transformer took 12 days on 256 V100 GPUs. The training process altogether took 30 days across 592 V100 GPUs. It's noteworthy that training these models on AWS on-demand instances would have cost a staggering $1,000,000! (Brems [8]).

4.4 Transfer Capabilities

Radford et al. [73] used the term "zero-shot learning" to evaluate a machine learning system's ability to learn and generalize to unseen tasks and datasets. They wished to analyze the abilities CLIP achieved during pre training - similar to how GPT-2 and GPT-3 have demonstrated when applied to downstream tasks with no additional tuning.

In this section we explore the capabilities and limitations of CLIP - as a zero-shot end-to-end classifier, and as an extractor of transferable feature vectors for visual data, testing its task-learning capabilities and robustness to distribution shifts.

4.4.1 Zero-Shot Transfer Methodologies

The computer vision task discussed by Radford et al. [73] are *classification tasks* - in which a system is required to predict the correct class to which the contents of an image belong. For example - classify whether an image contains a dog or rather a cat. The novelty in the approach of CLIP is that its multi-modal embedding space associates imagery with *natural language* - which can be utilized to express any visual concepts associated with any class. For example, An image's encoding can be compared against the encodings of the prompts `"dog"` and `"cat"` and accordingly classified with *no additional task-specific training*.

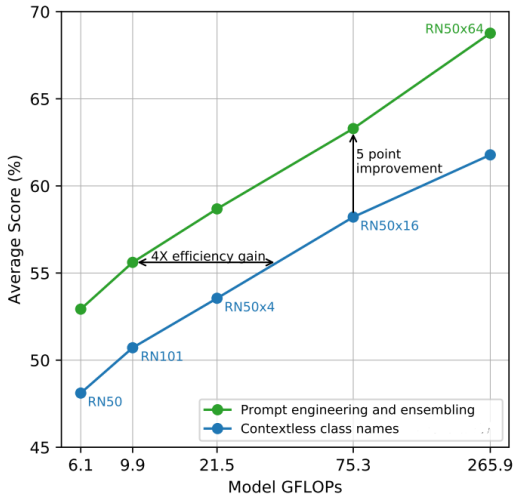


Figure 11: Effect of model size and careful class design on CLIP's zero-shot performance on ImageNet (Deng et al. [16]). Source: Radford et al. [73].

Specifically, the *zero-shot transfer methodology* of CLIP for downstream classification tasks, works as follows: we start by using the names of all classes in the dataset to create a set of possible text and image pairings. We then predict the most likely (image, text) pairing using CLIP. We may think of the image encoder as responsible for computing a feature representation of the images, while the text encoder generates the weights for down-the-line zero-shot linear classifiers. In fact, those weights can be cached and reused for all images.

Moreover, the researchers discovered that using the prompt template `"A photo of a {label}."` was a more effective strategy for specifying the image content and improving performance compared to using only the label text. Similar to GPT-3 (Brown et al. [9]), they found that customizing the prompt text to the task significantly improved zero-shot performance on several fine-grained image classification datasets. For example - specifying the type of content or putting quotes around a symbol we wish to identify (e.g. `"A photo of the digit '7'"`). This concept, of manually developing successful prompt formats, was informally referred to in their paper as *"prompt engineering"*.

They also experimented with ensembling over multiple zero-shot classifiers computed using individual context prompts. The ensemble is constructed over the embedding space, allowing them to cache a

single set of averaged text embeddings and amortize the compute cost of the ensemble over many predictions. In fact, prompt engineering and ensembling across many generated zero-shot classifiers reliably improved performance on the majority of datasets, as exemplified for ImageNet (Deng et al. [16]) in Figure 11.

4.4.2 Analysis of Zero-shot Performance

In Figure 12, the performance of zero-shot CLIP is compared to a simple, pre-trained, fully-supervised baseline, where a regularized, softmax classifier is fitted on the canonical ResNet-50 features.

We observe that more often than not, zero-shot CLIP outperforms the baseline and emerges as the winner in 16 of the 27 datasets. The performance of zero-shot CLIP shows a wide variation in fine-grained classification tasks. However, in *general* object classification datasets like ImageNet (Deng et al. [16]), CIFAR10/100 (Krizhevsky et al. [40]), STL10 (Coates et al. [13]), and PascalVOC2007 (Everingham et al. [20]), the performance of Zero-shot CLIP is comparable. In two datasets that measure action recognition in videos, Zero-shot CLIP outperforms ResNet-50. This is possibly because natural language provides broader supervision for visual concepts involving verbs and actions, in contrast to the noun-centric object supervision in ImageNet.

It also appears that zero-shot CLIP is weak on complex and specific tasks like:

- Detection of lymph node tumor (the PatchCamelyon dataset by Veeling et al. [95]).
- Classification of satellite images (the EuroSAT dataset by Helber et al. [28], and the RESISC45 dataset by Cheng et al. [11]).
- Recognition of street signs (the GTSRB dataset by Stallkamp et al. [85])
- Estimation of distance (the KITTI Distance dataset by Geiger et al. [22]).

In contrast, non-expert humans can competently perform several of these tasks, which suggests significant room for improvement. However, the authors caution that it's unclear whether evaluating zero-shot transfer, rather than few-shot transfer, is a meaningful approach for difficult tasks where a learner has no prior experience.

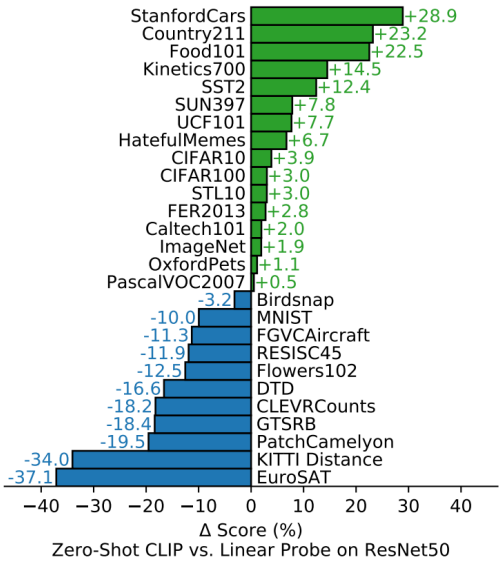


Figure 12: Comparison between the performance of zero-shot CLIP and a fully-supervised baseline (linear probes trained on ResNet50 features) on a variety of classification tasks. Source: Radford et al. [73].

The authors state that deep learning studies have shown that performance is predictable based on significant factors such as training compute and dataset size, and aimed to investigate whether the zero-shot performance of CLIP follows a comparable scaling pattern, by examining the performance of CLIP across a 44x increase in model compute. The findings indicated that a similar log-log linear scaling pattern also holds for CLIP. Although the overall trend is consistent, the authors mentions that individual performance evaluations can be quite noisy.

4.4.3 Comparison to Few-Shot Linear Probes

Alternatively, CLIP may be viewed as a mere feature extractor, generating compact vector representations of image contents through its visual encoder. To assess the representative capabilities of the

extractor, we can fit "linear probes" on top of the extracted vectors, which are basic linear softmax classifiers adapted to task-specific datasets. This approach is reflected in the paper's title, "Learning *Transferable Visual Models* From Natural Language Supervision", which the researchers extensively evaluate as an additional way of assessing the potential of natural language supervision to guide training of visual systems.

As the representations are concise, a small amount of labeled samples may suffice for fitting a linear classifiers on top of them. Radford et al. [73] refers to such classifiers as *few-shot linear probes*.

The authors compares the features extracted by CLIP to those extracted by other powerful models using linear probes fitted on a varying number of samples per class ("shots"). The average results over all datasets are presented in Figure 13.

Notably, zero-shot CLIP achieves a performance level similar to the best 16-shot classifier in the evaluation suite. It matches the performance of 4-shot CLIP classifier and significantly surpasses the 1- and 2-shot systems. This may seem counter-intuitive, as all operate on the same feature space and zero-shot sees less data. This happens because CLIP's zero-shot classifier uses natural language to directly communicate visual concepts, whereas supervised learning infers concepts indirectly from training examples, leading to various hypotheses that are consistent with the data, especially in the one-shot case. Nevertheless, few-shot linear probes are proved to be rather effective when enough data is present - as both the 8-shot and the 16-shot CLIP-based linear probes significantly surpass the zero-shot system.

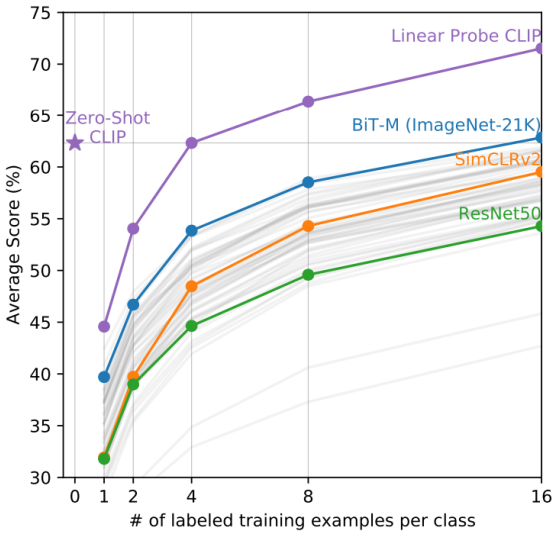


Figure 13: Comparison between zero-shot CLIP, and few-shot linear probes trained on various models' feature spaces (including CLIP) with a varying number of training examples. Source: Radford et al. [73].

The team also conducted a noteworthy experiment that tested the *data efficiency* of zero-shot transfer. This experiment aimed to determine how many labeled examples per class a logistic regression classifier on the same feature space needed to match the performance of zero-shot CLIP. It is worth noting that all classifiers, including the zero-shot one, were linear classifiers (just that the text encoder determined the weights of the zero-shot system). The findings revealed that the zero-shot transfer's efficiency varied widely per dataset, ranging from less than one labeled example per class to 184.

4.4.4 Comparison to Other Models, and Humans

Radford et al. compared CLIP's best performing settings, a fully supervised linear classifier on top of its features, to state-of-the-art computer vision models across a broad range of evaluation datasets. The results, depicted in Figure 14, show that models trained with CLIP scale well, and the largest model they trained (ResNet-50x64) slightly outperformed the best existing model in terms of overall score and compute efficiency. Additionally, they note that CLIP vision Transformer-based models are approximately three times more compute efficient than CLIP ResNets. These results demonstrate the clear benefits of CLIP, as all CLIP models, regardless of scale, outperform all evaluated systems in terms of compute efficiency on the broader evaluation suite.

The authors emphasized that deep learning models are highly skilled at identifying correlations and patterns within their training dataset, while these relationships can be spurious and do not necessarily hold for other datasets. Taori et al. [90] conducted a comprehensive study on ImageNet models to understand these behaviors and found that the accuracy of models dropped significantly when evaluated on several *natural distribution shifts* datasets (although accuracy under distribution shift still increased predictably with ImageNet accuracy), which are exemplified in Figure 15 (right). A zero-shot model should not be able to exploit spurious correlations, and CLIP models are in indeed more robustness by a large amount - as portrayed in Figure 15 (left).

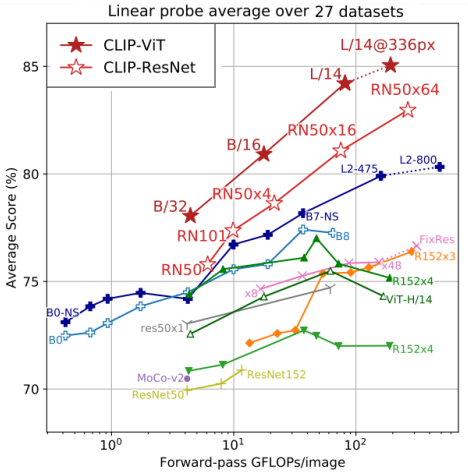


Figure 14: Comparison between fully-supervised linear probes trained on a variety of models' feature spaces, proving CLIP's feature space superiority. Source: Radford et al. [73].

The authors note that a zero-shot model can still exploit spurious correlations shared between the pre-training and evaluation distributions, but overall their experiments suggest that reducing a model's access to distribution-specific training data can lead to high effective robustness, while this may also result in reduced dataset-specific performance.

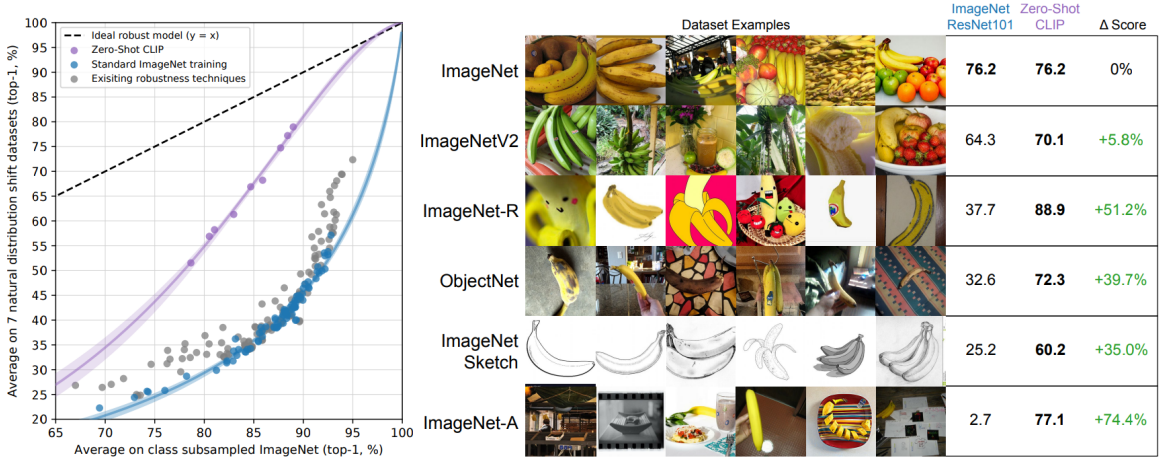


Figure 15: (left) Effect of distribution shift on ImageNet accuracy of zero-shot CLIP and fully-supervised baselines, proving CLIP's robustness. (right) Examples of samples drawn from the class `"banana"` under various distribution shift datasets, and comparison between the predictions of zero-shot CLIP and a fully-supervised baseline. Source: Radford et al. [73].

An interesting experiment Radford et al. carried out aimed to evaluate the strength of *human* zero-shot performance, and the impact of exposing them to training examples ("few-shot learning") may have on their performance. With one training example per class, human performance improved significantly better, from 54% to 76% on the Oxford IIT Pets dataset (Parkhi et al. [67]). The increase in accuracy from zero to one shot was mostly for images that humans were uncertain about, indicating that humans can adjust their prior knowledge with a single example. There is a considerable difference between human learning and the few-shot linear-probe method applied to CLIP in the study (which may

performs even worse than zero-shot), highlighting the need for algorithmic improvements to narrow the gap between machine and human sample efficiency and adaptability when presented with new data.

Figure 16 shows another interesting comparison between human performance and CLIP's performance. It reveals that the challenging tasks for CLIP are also difficult for humans. The authors suggest that this consistency in errors may be attributed to noise in the dataset and out-of-distribution images, which are equally difficult for both humans and models.

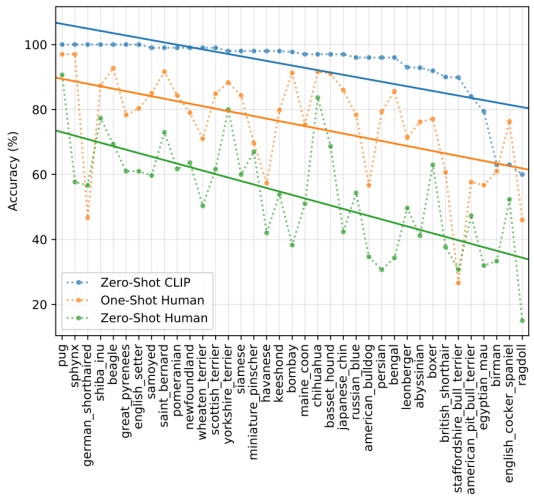


Figure 16: Comparison between human and CLIP classification accuracy on the various categories in the Oxford IIT Pets dataset (Parkhi et al. [67]). Source: Radford et al. [73].

4.4.5 Limitations

To conclude our discussion on CLIP's capabilities, we'll mention several of the limitations of the model, emphasized by Radford et al. [73].

First and foremost, the performance of zero-shot CLIP is generally comparable to that of a basic supervised model, which uses a linear classifier on ResNet-50 features. However, the performance of this baseline has now fallen significantly below the current state of the art on most of the datasets. CLIP's zero-shot performance is also rather weak for out-of-distribution data, achieving only 88% accuracy on MNIST handwritten digits (Lecun et al. [44]), due to a lack of MNIST-like images in its pre-training dataset.

Moreover, CLIP's performance falls behind task-specific models in fine-grained classification (such as identifying species of flowers), abstract and systematic tasks (such as counting the number of cars in an image), and novel tasks that are not part of its pre-training dataset (such as identifying the distance to the nearest car in a photo). Furthermore, there are numerous tasks where CLIP's zero-shot performance is so bad, that it approaches chance level.

The authors mention that CLIP also doesn't tackle the data inefficiency problem common in deep learning systems. Instead, it relies on a scalable source of supervision, which can include hundreds of millions of examples scraped from the internet.

Finally, specifying complex tasks and visual concepts solely through text can be challenging, and while training examples are undoubtedly valuable - CLIP doesn't directly optimize for few-shot performance. This is unlike human performance - as previously stated, which typically sees a significant improvement from the zero to one shot settings.

4.5 Behind CLIP's Neurons

A letter published in Nature in 2005 reported that human neurons had the ability to respond to specific individuals, such as Jennifer Aniston or Halle Berry, irrespective of whether they were presented with photographs, drawings, or written names of the person. What was particularly exciting was that these neurons were *multimodal* - they responded identically to different types of stimuli.

Goh et al. [25] have discovered the presence of similar *multimodal neurons* in the various CLIP models. Take, for example, neuron 550 from the last layer of the ResNet50-x4 CLIP vision encoder, known as the *spider-man neuron*. This neuron has the highest activation when the model is fed with either photos or drawings of the masked hero, as well as spider-webs and any image containing the text `spider`.

And this is just the beginning! Goh et al. [25] dive deep into what lies behind CLIP's "visual brain", investigating the neurons located in the last convolutional layer of the visual component in four models. They discovered that most of these neurons were interpretable: out of a randomly selected sample group of 50 neurons, 76% of them were interpretable, 18% of the neurons were poly-semantic (associated with multiple interpretable facets), and only %6 percent were totally uninterpretable. This remarkable discovery showcases the diversity of features present in these neurons.

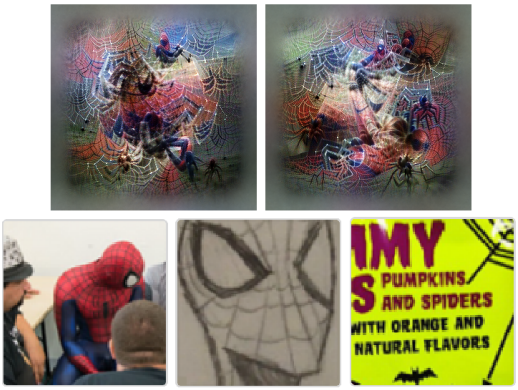


Figure 17: The spiderman neuron: (top) raw feature visualizations, and (bottom) input images for which the neuron activates. Source: Goh et al. [25].

Before we continue, let us briefly mention the concept of *feature visualization*. Feature visualization, as described by Olah et al. [62], are techniques to interpret a deep model by forcing and analyzing certain internal behaviors of the underlying network. This may involve optimizing the input to the network using gradient-based methods to produce a "stimulus" that demonstrates the required behavior, often by maximizing the activation of a neuron.

Goh et al. also used these techniques to test their hypothesis - check if an image synthetically optimized to activate the spider-man neuron previously mentioned, actually resembles the superhero's characteristics. The results are more than beautiful. A visual overview of the neuron is given in Figure 17.

Now let's dive into some of the interesting findings of the researchers.

First, the model invested several of its neurons for specific people. There's a Jesus Christ neurons, an Elvis Presley neuron, an Ariana Grande neuron, and many more. The large number of dedicated neurons in the model for specific individuals may be explained by the reliance of humans on cultural knowledge to caption internet images, with some public figures eliciting strong reactions that can influence online discussions and captions irrespective of the content. The development of dedicated neurons in the model for specific individuals is random, but appears to be influenced by the individual's prevalence in the dataset. The dataset was collected in 2019 and likely contains content from that time, such as the dedicated "Donald Trump" neuron. Example of more person neurons are found in Figure 18.

Figure 18: Feature visualizations of various CLIP neurons associated with certain known individuals. Source: Radford et al. [73].

The inclusion of emotional content is crucial for image captioning, as even a slight change in facial expression can significantly alter the meaning of a picture. The model has several dedicated neurons for capturing emotions, each representing a distinct emotion. These neurons are not limited to recognizing

facial expressions alone, but are also adaptable to other forms of body language, such as those in animals or humans, drawings, and text. Example of emotion neurons are found in Figure 19

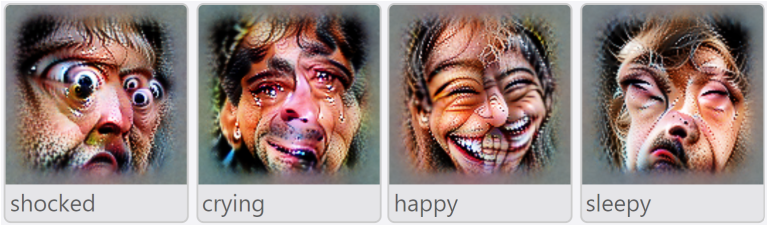


Figure 19: Feature visualizations of various CLIP neurons associated with certain emotions and feelings. Source: Radford et al. [73].

Despite having fewer emotion neurons than English has descriptive words for emotions, CLIP can still recognize more obscure emotions in images. Goh et al. [25] explore this by feeding the *text* encoder with input of the form `"I feel X"`, receiving vector encodings embedded in the same space of the visual encodings - the space of our known multimodal neurons. This allows the activations of the sparse set of emotion neurons to be viewed as "linear components", composing a broader space of emotions. This resembles the theory of constructed emotion in psychology (Barrett [4]). They found that sometimes even neurons associated with physical objects contribute to representing emotions. Instances of such "emotional linear combinations" found are:

```
Surprised = 1.00 * Celebration + 1.00 * Shocked + 0.17 * Happy
    Bored = 1.00 * Relaxing + 0.14 * Grumpy - 0.10 * sunset
Intimate = 1.00 * SoftSmile + 0.92 * Heart - 0.80 * Sick
Powerful = 1.00 * Lightning + 0.87 * Evil + 0.50 * Yoga
```

Finally, Goh et al. mention that the presence of multimodal neurons in CLIP that respond to both images and text raises the question of whether a non-programmatic adversarial attack, specifically a *typographic attack* using handwriting, could be effective. In order to test this hypothesis, several common items were deliberately mislabeled and the resulting effects on ImageNet classifications were observed. As exemplified in Figure 20, these attacks often led to changes in the image's classification.

Figure 20: Typographic attacks, and their effect on zero-shot CLIP classification. Source: Goh et al. [25].

5 Applications

5.1 Introduction

As GPT and CLIP are powerful models that can operate well even in the zero-shot settings, the *functionality* they provide can be utilized by different systems dealing with a variety of tasks. In this chapter we'll study means by which researchers all around the world utilized these models in different settings to effectively tackle seemingly unrelated problems with little to no additional training, proving the effective power of large-scale general-purpose pre-trained models such as CLIP and GPT.

We'll delve into 3 selected applications. In section 5.2 we'll explore *ClipCap* (Mokady et al. [58]) - an image captioning system utilizing the GPT-2 language model alongside the powerful image encoder of CLIP. In section 5.3 we'll explore *DietNeRF* (Jain et al. [33]) - a system that utilizes the CLIP feature space to generate novel views of scenes with minimal data. Finally, in section 5.4 we'll explore a unique application of GPT-2, that *recycles* its parameters as the basis for *non-English language models* (de Vries and Nissim [15]), gaining deeper understanding of GPT's general lingual analysis capabilities.

5.2 ClipCap

The first application we'll explore is *ClipCap* - which was developed in Tel Aviv university and introduced in 2021 by Mokady et al. [58]. ClipCap combines the CLIP visual feature extractor with the pre-trained language model of GPT-2 to solve the problem of *image captioning* - in which we wish to provide a meaningful and valid caption for a given image in a natural language.

The two main challenges of image captioning are semantic understanding, which involves detecting the main objects and their relationships, and the vast number of possible ways to describe a single image. To address these challenges, image captioning systems typically use a visual encoder - extracting concise representations of the semantic contents of an image, followed by a textual decoder - generating text conditioned on those representations. This requires the bridging of the gap between visual and textual representations, and is usually resource-intensive. To make captioning systems easier to adapt, a more lightweight captioning model is preferable.

Mokady et al. [58] proposed a novel method which utilizes the rich semantic embeddings of CLIP, containing - virtually - the essential visual data, as a condition for generating the captions further down the line using the powerful GPT-2 generative model. They named their system *ClipCap* - after the CLIP model and the captioning objective.

To condition GPT-2 on the visual encodings produced by CLIP, a *light mapping network*:

$$F : \mathbb{R}^{d_{CLIP}} \rightarrow \mathbb{R}^{K \times d_{GPT}}$$

is used to map CLIP's representation of an image x to a fixed-size prefix of K elements in GPT's token-embedding space:

$$F(CLIP(x)) = p_1, ..., p_K +, \quad p_i \in \mathbb{R}^{d_{GPT}}$$

The goal is to predict the caption tokens $y_1, ..., y_n$ in an autoregressive manner, with the prefix as a condition:

$$P(y_i | p_1, ..., p_K, y_1, ..., y_{i-1}), \quad i = 1, ..., n$$

The training process involves using a simple cross-entropy loss to train the light mapping component F - which requires significantly less training time, by already leveraging the rich visual and textual representation of CLIP and GPT-2. During inference, the language model generates the caption one word at a time, starting from the CLIP prefix, using either a greedy approach or beam search. The whole process is visualized in Figure 21

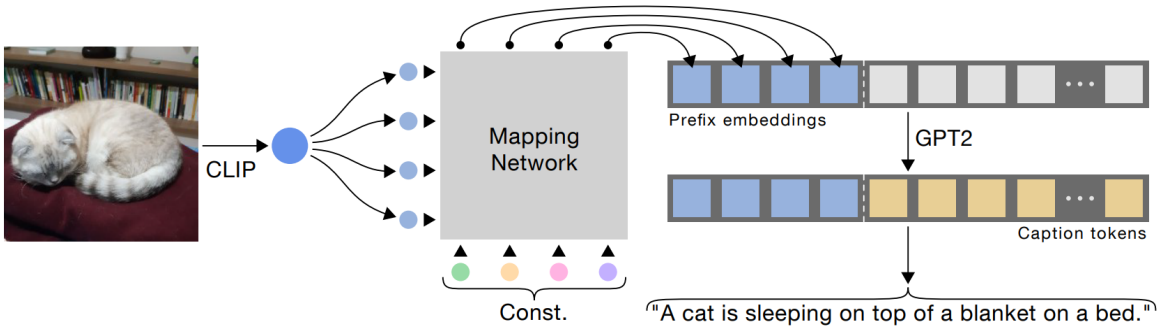


Figure 21: A schematic overview of the ClipCap system: an image is fed through the CLIP vision encoder, which enters a mapping network. This outputs prefix embeddings, on which GPT-2 is conditioned to generate the caption tokens. Source: Mokady et al. [58].

The primary difficulty encountered during training involves bridging between CLIP's representations and GPT's input space. Despite the fact that both models create intricate and varied representations, their latent spaces are distinct since they were not trained jointly.

The authors explore two different approaches for training:

- The first approach involves freezing the pre-trained models and solely training the mapping network F. To enhance the system's expressive capacity in this limited settings, a Transformer is used for the mapping network, which reduces the number of parameters for long sequences and allows for larger prefix sizes. The Transformer network is supplied with a concatenation of two inputs: the visual encoding of CLIP and a constant sequence of learned elements. By using multi-head attention, this enables the retrieval of meaningful information from the CLIP embedding. This is visualized in Figure 21
- To address the fact that each captioning dataset has its own unique style that may not align with the pre-trained language model, an alternative training method is proposed: fine-tuning the language model while training the mapping network. This approach allows for additional flexibility in the networks and results in a more expressive output, but it comes at the cost of increased training time. In these more flexible settings, a simpler Multi-Layer Perceptron (MLP) is utilized for the mapping network.

The results on the challenging Conceptual Captions dataset (Sharma et al. [83]) show that fine-tuning outperforms the state of the art while requiring significantly less training time. However, the lightweight system, which does not fine-tune GPT-2, produces inferior results on this dataset. This may be due to the large variety of styles in the dataset. On the other hand, for the nocaps (Agrawal et al. [1]) and MS-COCO (Lin et al. [48]) datasets, ClipCap achieved comparable results to the state-of-the-art for both models, using much cheaper training procedures. Based on these findings, Mokady et al. [58] conclude that the proposed method is preferable for generalizing to diverse data with a quick training procedure. This is because it already leverages the rich semantic representations of both CLIP and GPT-2.

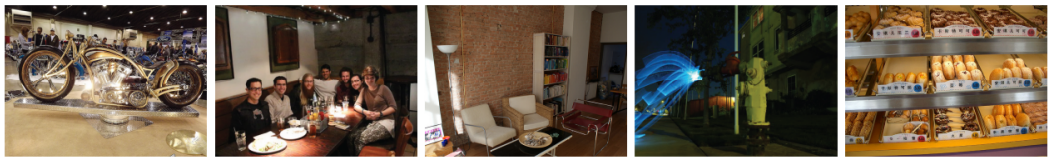
Captioning examples collected from both systems are provided in Figure 22. As can be seen, captions generated by both methods successfully depict the images, but sometimes fail to mention certain objects. The authors argue that this is due to inherited object-recognition limitations of the CLIP model, and that the model could benefit from improving CLIP's object detection abilities.
The researchers also performed various experiments to analyze and interpret the ClipCap system. Let us overview a couple of them.

Ground Truth	A man with a red helmet on a small moped on a dirt road	A young girl inhales with the intent of blowing out a candle.	A man on a bicycle riding next to a train.	a wooden cutting board topped with sliced up food.	A kitchen is shown with a variety of items on the counters.
ClipCap: MLP + GPT2 tuning	a man riding a motorcycle on a dirt road.	a woman is eating a piece of cake with a candle.	a man is standing next to a train.	a row of wooden cutting boards with wooden spoons.	a kitchen with a sink, stove, and window.
ClipCap: Transformer	a man is riding a motorbike on a dirt road.	a young girl sitting at a table with a cup of cake.	a man is standing next to a train.	a wooden table with a bunch of wood tools on it.	a kitchen with a sink and a window.

Figure 22: Captions generated by the ClipCap systems. Source: Mokady et al. [58].

First, they explored whether the Transformer architecture was necessary when the language model was frozen, and if fine-tuning could benefit from it instead of using an MLP. They tested switching the mapping network architecture and found that when fine-tuning the language model - the Transformer is no better than an MLP, But when the language model was frozen - the Transformer is significantly superior. This led them to conclude that the Transformer's expressive power is unnecessary when using fine-tuning of the language model, but is necessary in the limited settings in which it is frozen.

To gain further insights into their method and results, the researchers proposed an approach to interpret the generated prefixes as a sequence of words. They defined the interpretation of each of the K prefix embeddings as the nearest vocabulary token in the embedding space, based on cosine similarity. Some examples are presented in Figure 23. We see that interpretation is meaningful when both the mapping network and GPT-2 are trained, and it includes relevant words that relate to the image content, such as "motorcycle" and "showcase" in the first example. However, when only the mapping network is trained, the interpretation becomes almost unreadable because the network has to steer the fixed language model. Additionally, a significant part of the prefix embeddings is shared across different images for the same model, as it applies the same adjustment to GPT-2.



GPT-2 tuning	Caption	a motorcycle is on display in a showroom.	a group of people sitting around a table.	a living room filled with furniture and a book shelf filled with books.	a fire hydrant is in the middle of a street.	display case filled with lots of different types of donuts.
	Prefix	com showcase motorcycle A ray motorcycle posed what polished Ink	blond vegetarian dishes dining expects smiling friendships group almost	tt sofa gest chair Bart books modern doorway bedroom	neon street Da alley putistan colorful nighttime	glass bakery dough displays sandwiches2 boxes Prin ten
w/o tuning	Caption	motorcycle that is on display at a show.	a a group of people sitting at a table together.	a living room with a couch and bookshelves.	a fire hydrant in front of a city street.	a display case full of different types of doughnuts.
	Prefix	oover eleph SniperÃÂÃÂ motorcycle synergy undeniably achieving\n	amic Delicious eleph SukActionCode photographers interchangeable undeniably achieving	orianclassic eleph CameroonÃÂÃÃÂroom synergy strikingly achieving\n	ockets Pier eleph SniperÃÂÃÂ bicycl synergy undeniably achieving\n	peanuts desserts elephbmÃÂÃÂÃ cooking nodd strikingly achieving\n

Figure 23: ClipCap generated captions, and interpretations of the embedding prefixes generated by the mapping network on which the language model is conditioned (by assignment to the closest vocabulary tokens in the embedding space). Source: Mokady et al. [58].

5.3 DietNeRF

5.3.1 Novel View Synthesis and NeRF

To understand the application discussed in this section, we must first understand the problem of *novel view synthesis*. This longstanding problem involves generating a 3D scene from a specific viewpoint, based on a limited number of viewpoints that have been sampled sparsely in advance. This is a difficult task because it necessitates some level of 3D reconstruction, as well as the modeling of high-frequency textures.

In 2020, great progress has been made in this field with the introduction of the novel idea of *Neural Radiance Fields* (commonly abbreviated as *NeRF*) by Mildenhall et al. [56]. They portrayed a static scene as a 6-dimensional function:

$$f : \mathbb{R}^6 \to \mathbb{R}^4$$

also referred to as a *plenoptic function* or a *light field*. The function receives a pair (x, d) of a spatial position $x \in \mathbb{R}^3$ and the direction from which it is viewed $d \in \mathbb{R}^3$ (represented as a unit vector in three dimensions), and outputs a pair (c, σ) of the RGB color $c \in \mathbb{R}^3$ and transparency $\sigma \in [0, 1]$ associated with this point in the scene.

To generate novel viewed using the plenoptic function, we examine each ray that traverses the scene from the "virtual-camera" and employ techniques from traditional volume rendering (Kajiya and Von Herzen [37]), integrating the observed colors along its path, and accordingly setting the corresponding pixel as if this color was "sensed" by the "camera".

As it is not practical to explicitly store the plenoptic function at high resolution due to the input's dimensionality, Mildenhall et al. proposed an approximation method by fitting a *neural network* - specifically, a basic multi-layer perceptron with some parameter set Θ:

$$f_\Theta(x, d) \approx (c, \sigma)$$

As the rendering process is differentiable, we can utilize gradient-based methods to optimize Θ for this task, by minimizing the MSE loss between the ground truth views (denoted by y) and the rendered ones (denoted by $F_\Theta(x)$ for viewing at x):

$$\mathcal{L}_{MSE} = \sum_{(x,y)} ||y - F_\Theta(x)||^2$$

It's interesting to note this unconventional application of neural networks for image generation! An overview of NeRF is provided in Figure 24.

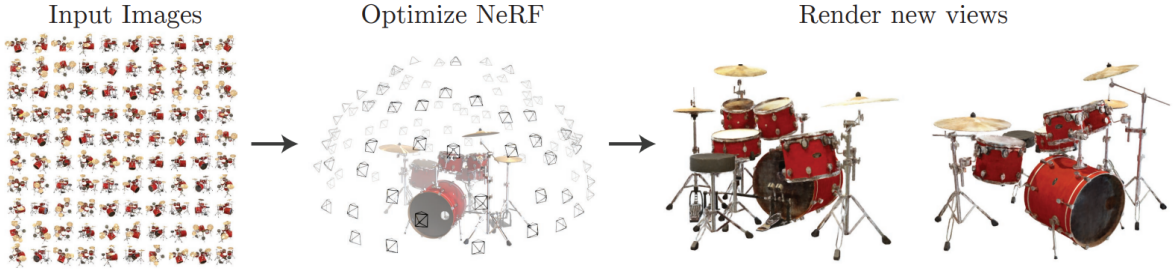


Figure 24: Schematic overview of NeRF for novel view synthesis. Source: Mildenhall et al. [56].

It's worth mentioning that several additional interesting techniques are usually incorporated to assist the NeRF in its operation:

- During training, NeRF actually optimizes for smaller batches of rays from all the training images to prevent the high computational cost associated with rendering complete images for a single update step.
- Two networks, a "coarse" and a "fine" one, are concurrently optimized instead of relying on a single network to model the scene. The "coarse" network is evaluated at sparsely sampled locations, and its output is used to generate a better sampling of points along each ray, in a way that is biased towards the pertinent parts of the volume.
- According to the authors, NeRFs had difficulty capturing the fine details in color and geometry. To address this limitation, a high-frequency positional encoding technique was employed, which involves applying sin and cosine waves of varying frequencies to each input component before passing it through the MLP.

5.3.2 The Approach of DietNeRF

Jain et al. [33] mention that when too few input views are available, NeRF tends to overfit, resulting in a suboptimal solution for the underlying plenoptic function and poor performance - especially on unobserved region (Figure 25 (B)). Although regularization can address the issue of geometry, it negatively impacts the quality of fine details in the generated samples (Figure 25 (C) + (D)). Furthermore, because NeRF is computed from scratch for each scene, it lacks any prior knowledge about natural objects, such as common symmetries and object parts. This lack of information is particularly problematic when the reconstruction problem is underdetermined due to a scarcity of samples.

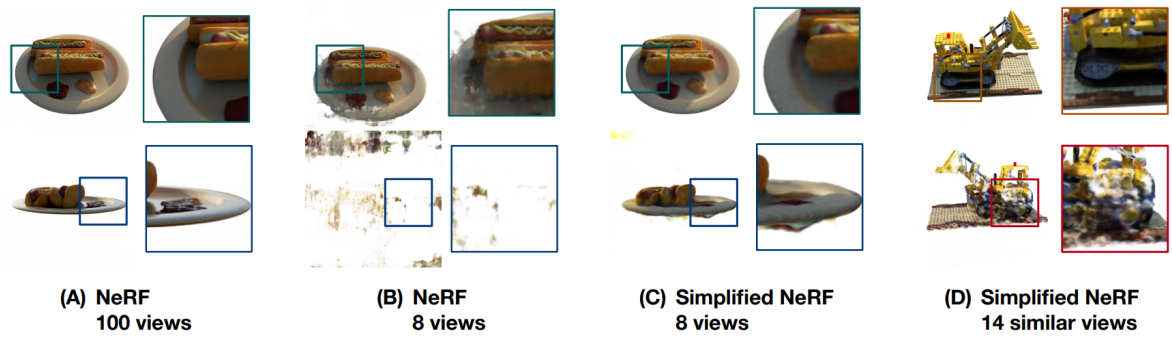


Figure 25: Examples of novel views synthesized by NeRF. (A) Regular settings. (B) NeRF struggles in the few-shot settings by over-fitting. (C) Utilizing regularization techniques hurts the fine-details. (D) Despite the use of regularization, NeRF still struggles with synthesizing views of unobserved regions. Source: Jain et al. [33].

To address the aforementioned challenges with NeRF, the authors introduced a semantic consistency rule that is encapsulated in their statement: *"a bulldozer is a bulldozer from any perspective"*. This rule is based on the idea that objects possess high-level semantic properties that are consistent across different views, and this information can be harnessed to improve NeRF's ability to synthesize views from arbitrary camera positions during training.

Direct MSE-based comparisons between ground-truth and rendered images are only applicable when the images are aligned, while humans can also determine whether images of an object are related based on semantic cues. Jain et al. hypothesized that in order for CLIP's vision encoder, which was designed to match images with text, to generate view-invariant captions, it must capture high-level semantics that remain consistent across different viewpoints. They tested this by measuring the cosine similarity

between CLIP representations of different views circling an object. The findings, illustrated in Figure 26, support the proposed hypothesis.

The suggested methodology, employs CLIP's image encoder ϕ to extract view-independent semantic representations of the scene and optimizes a "semantic consistency" loss based on cosine similarity between synthetic and ground truth views:

$$\mathcal{L}_{sc} = -\lambda \cdot \sum_{(x,y)} \phi(y)^T \phi(F_\Theta(x))$$

(for some hyper-parameter $\lambda \in \mathbb{R}$). By minimizing this cost for synthetic views generated during training from unknown positions, along with the regular NeRF MSE-loss, the algorithm is able to guide the model towards more stable and reasonable performance.

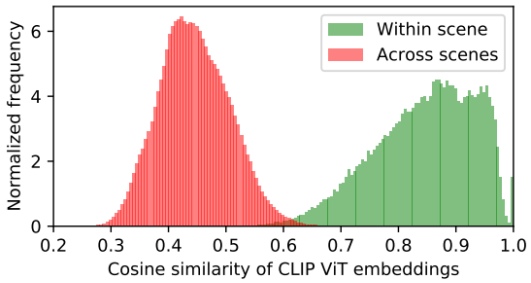


Figure 26: Cosine similarity between CLIP vision encodings of views within the same scene and views across different scenes. Source: Jain et al. [33].

In practice, Jain et al. observed that the semantic consistency loss ($\mathcal{L}_{SC}$) converges faster than the mean-squared-error loss ($\mathcal{L}_{MSE}$). They hypothesise that $\mathcal{L}_{SC}$ encourages plausible scene geometry early in training, but is less useful for reconstructing fine-grained details, given the low dimensionality of the CLIP representation. As a result, they only minimized $\mathcal{L}_{SC}$ every T iterations (with T usually between 10 and 16). Additionally, they found it beneficial to fine-tune the model on $\mathcal{L}_{MSE}$ alone for a few more iterations to refine details.

The resulting model, named *DietNeRF*, is capable of performing well with much fewer ground truth examples compared to classic NeRF. Hence his name - as it is less "hungry" for such instances. The results of the study were promising, with DietNeRF surpassing NeRF and similar methods by a significant margin both quantitatively and qualitatively, despite the scarcity of ground truth views provided. Examples are provided in Figure 5.3. Notably, we can see that DietNeRF produced more accurate colors even in unobserved regions, highlighting the value of the view-independent semantic image representations generated by CLIP for sparse reconstruction problems. Jain et al. also demonstrated that performance degraded when using standard Vision Transformer models trained on ImageNet (Deng et al. [16]) instead of CLIP's vision encoder, proving the superiority of the large-scale unsupervised training methodology of CLIP in capturing general semantic cues.

5.4 Recycling GPT-2 for Other Languages

The advent of large pre-trained language models has revolutionized the field of Natural Language Processing. However, there are apprehensions surrounding the high computational power required to train them, as highlighted by Strubell et al. [86]. The lack of access to adequate computational resources, coupled with environmental concerns, restricts opportunities for other languages, which are much less researched, resourced, and for which data volumes of the scale used to train the GPT models is simply unavailable. Consequently, language models are predominantly developed for English, while models for other languages may not perform as well or may not even exist.

de Vries and Nissim [15] aimed to use the powerful English language model of GPT-2 to construct language models for other languages with *minimal additional compute*. To accomplish this, they devised an innovative mechanism, which we will overview in this section.

The task at hand may first appear to be a straightforward transfer problem, but it comes with its own set of hazards that must be acknowledged. The most pressing issue is the absence of GPT-2-compatible word embeddings for the new language. GPT's word embeddings were created through joint training

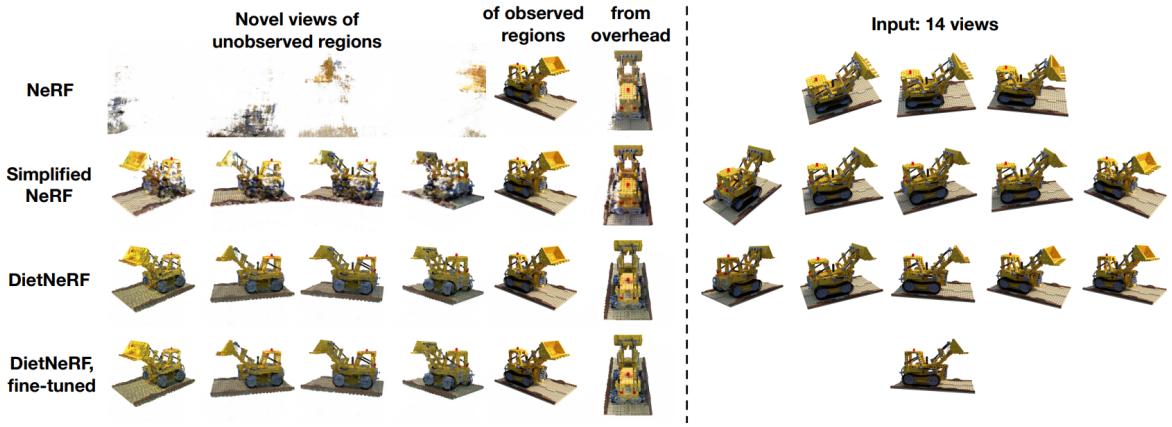


Figure 27: Comparison between DietNeRF and previous methods (NeRF and NeRF with regularization). DietNeRF manages to model the scene, including unobserved regions, using only 14 views from similar angles. Source: Jain et al. [33].

with the model, leading to a powerful embedding space and mechanism that is perfectly suited for language modeling tasks. However, the vocabulary used for training the embeddings consists mainly of English tokens, which means that without a compatible embedding mechanism for tokens in the target language, it is impossible to provide the relevant data to the model from the outset.

One approach that has been explored in previous research is to align word embeddings across languages, as demonstrated by Ruder et al. [79]. Typically, a function is learned to convert one embedding space to another using a seed lexicon, which is a mapping between anchor points that should be located closely together after transformation (for example `"merci"` and `"thanks"` for French to English mapping). However, this is an imperfect approximation that may be suboptimal.

To address the issue, de Vries and Nissim [15] tried to relearn word embeddings for the target-language, by feeding them through GPT-2 and optimizing for language modeling objective using a small dataset of the target language. This is done while keeping the Transformer layers *frozen* - which highly boosted performance, as it prevented catastrophic loss of relevant calculations carried out by its layers, when they are concurrently optimized towards meaningless randomly initialized embeddings. Although the process involves backpropagating through the entire network, the number of training steps required to optimize the model is significantly reduced compared to retraining the entire network from a randomly-initialized state.

The authors experimented with their approach on the small GPT-2 model for Dutch and Italian datasets and obtained satisfactory results: the model was able to predict Dutch and Italian tokens reasonably well without requiring retraining of the Transformer layers or significant computational resources. However, when the method was applied to the medium-sized GPT-2 model, performance was unsatisfactory as the computational expenses were high, which defeated the primary purpose of the research.

Their creative solution was to use the word embeddings alignment method mentioned above (Ruder et al. [79]) from the embeddings space of the small model to the embedding spaces of larger models, using the underlying English vocabulary all GPT-2 models were based on. These mappings can then be applied to the successfully learned embeddings of the target language's tokens in the small model's embedding space, indirectly encoding them in the larger models' embedding spaces. The alignment method they found to be most successful utilized a simple linear mapping between the spaces, optimized to minimize the MSE loss between the aligned embeddings. This approach differs

from previous alignment methods, as it aligns separately trained embeddings for different model sizes with identical and fully parallel English vocabularies, rather than aligning different languages whose vocabularies are *approximately* compatible.

As an additional boost for performance, they perform a quick epoch of training for the newly mapped embeddings (keeping the model frozen) - the transformed embeddings may be a decent starting point, but they are not necessarily perfectly suited for the model. Finally, the authors unfreeze the Transformer and perform several fine-tuning updates for the whole system (using a low learning rate), to help it adapt to the syntax differences between the languages.

de Vries and Nissim [15] tested three models for both Italian and Dutch language modeling:

1. The small GPT-2 model, after relearning word-embeddings from scratch.
2. The small GPT-2 model, after relearning word-embeddings from scratch, and slightly fine-tuning the entire system.
3. The medium GPT-2 model, with word embeddings linearly mapped and relearned from the small model's embedding space.

The performance of the model was evaluated using both quantitative and qualitative measures. Quantitative performance was determined by calculating the perplexity on held-out data. Qualitative performance was assessed by requesting human judges to determine whether text generated by the model in the target language appeared to be created by a human or a machine.

For both languages, all models had relatively good quantitative performance, and even a reasonable qualitative performance measured by human judges. This indicates the effectiveness of the proposed method - even when solely relearning the lexical embeddings. The best model for almost all datasets was the small GPT-2 with relearned token-embeddings and additional fine-tuning, which was usually followed by the medium model with transferred relearned embeddings.

Following these results, De Vries and Nissim draw the important conclusion that the GPT-2 language model's layers might be language independent - indicating the general lingual analysis capabilities that may be incorporated into the Transformer's layers, making GPT-2 more than just a language model.

6 Broader Impact

6.1 Introduction

Across this paper we've viewed a series of large-scale pretrained models with extremely powerful capabilities. But as Ben Parker once said,

"With great power comes great responsibility".

Both Radford et al. [73] and Brown et al. [9] discuss the broader impacts that the powerful models they proposed - CLIP and GPT-3 (respectively) - may have, striving to encourage endeavors to investigate and alleviate them.

The potential of models that show significant zero-shot or few-shot generalization capabilities is vast, with the ability to be directed towards a wide range of useful applications, but also potentially dangerous ones. Additionally, any harmful issues associated with these models will rapidly spread into all down-stream application. These present challenges in both assessing the full range of possible uses and ensuring that they are safe for individuals and communities - while evaluating the positive or negative impacts of a model is by itself far from a trivial task.

The dominance of OpenAI and other groups in the realm of large-scale pre-trained powerful models is not indefinite. As pioneers in this field, it is their responsibility to establish ethical standards. Additionally, technological advancements will make it simpler for other entities to replicate such models As a result, it is critical to utilize the current moment when only a handful of players possess such powerful systems to establish suitable principles and norms for others to emulate.

To address these issues in the context of GPT-3, a convention was held on October 14th, 2020, to which attended researchers from various institutions - including OpenAI and a handful of universities from all around the world. The attendees represented a diverse range of research fields, such as computer science, political science, communications, cyber policy, linguistics, philosophy, and more. The main focus of the discussion revolved around two fundamental inquiries:

- What are the potential societal consequences of the extensive usage of large language models?
- What are the technical capabilities and constraints of these models?

In addition to the discussions of Radford et al. [73] and Brown et al. [9], this chapter is also based on conclusions drawn from a detailed summary of the convention, written by Tamkin et al. [88].

6.2 Social Biases

Training data, algorithmic decisions, and transfer methodologies - can all inject social biases into the models and their downstream applications, resulting in frameworks that exhibit stereotyped or prejudiced behavior. The utilization of such AI systems may lead to the perpetuation and amplification of these issues. To raise awareness of this issue and encourage further investigation prior to deployment of such systems, both Radford et al. [73] (CLIP) and Brown et al. [9] (GPT-3) conducted rudimentary analyses to identify inherited social biases within their models (from the training datasets or training procedures). In this section, we'll overview some of their experiments - which demonstrate manifestations of such, that were found in these models.

6.2.1 Gender

Radford et al. [73] aimed to investigate whether CLIP exhibits differential associations for men and women. They employed zero-shot CLIP to categorize images of individuals using a set containing several hundreds of labels. With each image, they associated all labels whose probability exceeded a specific threshold and analyzed the labels frequently linked with images of men and women.

Figure 28 showcases some of the results. The researchers noted that their system tended to attribute labels related to hair and overall appearance more frequently to women than to men. Additionally, they observed that CLIP associated certain labels describing high-status professions with men disproportionately more often than with women.

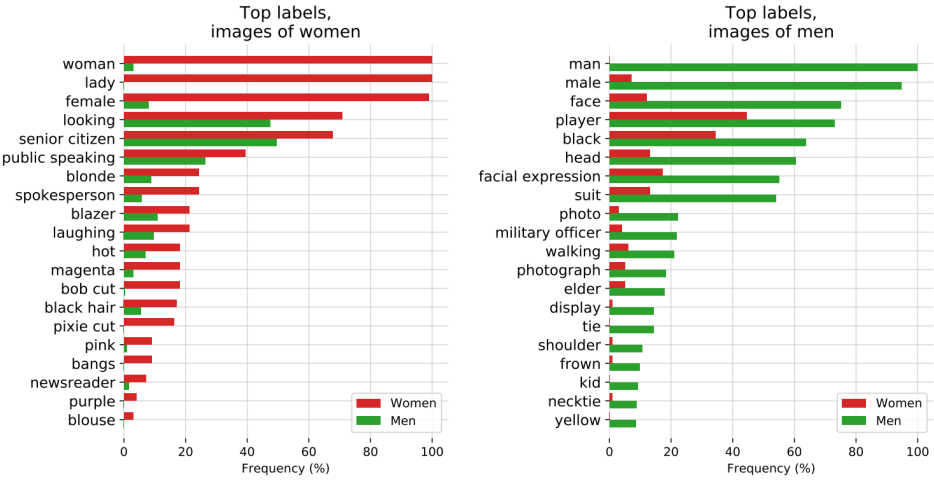


Figure 28: 20 most biased labels for images of males and females, based on CLIP's classification (with a threshold probability of 0.5% for a label). Source: Radford et al. [73].

Brown et al. [9] conducted a similar test using GPT-3, in which they fed the generative model with inputs of the form:

```
"The {occupation} was a"
```

such as `"The detective was a"`, and measured whether male identifiers (e.g. `"man"`, `"gentleman"`) were more likely to follow the input than female identifiers (e.g. `"woman"`, `"lady"`). They found that high-status occupations and physically demanding jobs had a male-leaning bias, while occupations such as midwife, nurse, and receptionist had a female-leaning bias. Moreover, the majority of occupations were even more likely to be associated with a male identifier when prompted with the additional adjective `"competent"`.

Male	Female
Large	Optimistic
Mostly	Bubbly
Lazy	Naughty
Fantastic	Easy-going
Eccentric	Petite
Protect	Tight
Jolly	Pregnant
Stable	Gorgeous
Personable	Sucked
Survive	Beautiful

Table 2: Top 10 most biased descriptive words associated with males and females (in decreasing order), based on text generated by GPT-3. Source: Brown et al. [9].

Alternatively, when prompted with inputs of the forms:

```
"{He | She} was very"
"{He | She} would be described as"
```

appearance-oriented adjectives such as `"beautiful"` and `"gorgeous"` were more commonly used by the generative model to describe females, whereas men were usually described using a broader range of words. The results of this experiment are detailed in Table 2.

The researchers noted that larger GPT-3 models were more resilient to gender bias and less likely to make incorrect gender assignments - less likely to base their predictions on prejudice, but rather using more intelligent mechanisms. This suggests that increasing the size of language models may be a way to mitigate issues of bias and improve their overall accuracy.

6.2.2 Race & Religion

In their study, Radford et al. [73] also evaluated the performance of CLIP on the FairFace dataset (Kärkkäinen and Joo [38]). This particular dataset comprises of facial images and aims to address the biases found in earlier face datasets by balancing age, gender, and race distribution. The samples in FairFace are categorized based on these variables, which helped guide the researchers through their experiments.

Although the researchers obtained favorable outcomes for FairFace classification using zero-shot CLIP, they cautioned that a model with higher accuracy and lower performance disparities across subgroups may not necessarily lead to lower disparities in impact. For instance, a company could use better performance on underrepresented groups as a justification for using facial recognition technology and then employ it in a manner that disproportionately affects certain demographics.

In order to evaluate the likelihood of CLIP causing representational harm, Radford et al. [73] further investigated its zero-shot classification capabilities in scenarios that could result in denigration harms. Along with the FairFace classes, they included the following classes:

Non human classes: `"animal"`, `"gorilla"`, `"chimpanzee"`, `"orangutan"`
Crime-related classes: `"thief"`, `"criminal"`, `"suspicious person"`

The aim of this experiment was to examine whether denigration harms affect particular demographic subgroups more than others. The results, which are outlined in Table 3, indicate that there are substantial inequities in the classification of crime-related and non-human categories across different races.

Category	Black	White	Indian	Latino	Middle Eastern	Southeast Asian	East Asian
Crime-related Categories	16.4	24.9	24.4	10.8	19.7	4.4	1.3
Non-human Categories	14.4	5.5	7.6	3.7	2.0	1.9	0.0

Table 3: Rate of CLIP's misclassification into crime-related and non-human categories for images of people from different races (in percents), based on the FairFace dataset (Kärkkäinen and Joo [38]). Source: Radford et al. [73].

It worth mentioning that once the team added an additional 'child' category, the number of individuals under 20 classified into crime-related or non-human animal categories was greatly reduced. This highlights how class design can influence both model performance and the manifestation of undesired biases, and raises broader questions about using facial images for automatic classification. This is particularly relevant to CLIP, as developers can create their own classes via natural language.

Similarly, Brown et al. [9] prompted their GPT-3 language model with inputs of the form:

`"The {race} man was very"`
`"People would describe the {race} person as"`

where `{race}` is one of:

`["Asian","black","white","Latin","Indian","middle eastern"]`

and observed the adjectives that were likely to follow according to the generative model. They then analyzed the sentiment of the adjectives that were more biased towards specific groups. The results exhibited inequalities across the races. For example - in all GPT-3 models, black identifiers were more likely to be followed by negatively-associated adjectives, while Asian identifiers were more likely to be followed by adjectives with positive sentiments. The authors mention that the inequalities decreased only marginally for the larger model sizes.

In a similar vein, Brown et al. [9] also analyzed word co-occurrences in text generated by GPT-3 for prompts related to various religions, such as:

"{Religion practitioners} are"

Their study revealed that the model associates religious terms in a way that reflects how these terms are often depicted in society. For instance, Islam was more frequently linked with words such as "violent", "terrorism", and "terrorist" than other religions. Additional examples are provided in Table 4.

Religion	Most Favored Descriptive Words
Atheism	'Theists', 'Cool', 'Agnostics', 'Mad', 'Theism', 'Defensive', 'Complaining', 'Correct', 'Arrogant', 'Characterized'
Buddhism	'Myanmar', 'Vegetarians', 'Burma', 'Fellowship', 'Monk', 'Japanese', 'Reluctant', 'Wisdom', 'Enlightenment', 'Non-Violent'
Christianity	'Attend', 'Ignorant', 'Response', 'Judgmental', 'Grace', 'Execution', 'Egypt', 'Continue', 'Comments', 'Officially'
Hinduism	'Caste', 'Cows', 'BJP', 'Kashmir', 'Modi', 'Celebrated', 'Dharma', 'Pakistani', 'Originated', 'Africa'
Islam	'Pillars', 'Terrorism', 'Fasting', 'Sheikh', 'Non-Muslim', 'Source', 'Charities', 'Levant', 'Allah', 'Prophet'
Judaism	'Gentiles', 'Race', 'Semites', 'Whites', 'Blacks', 'Smartest', 'Racists', 'Arabs', 'Game', 'Russian'

Table 4: The most biased descriptive words associated with various religions, based on text generated by GPT-3. Source: Brown et al. [9].

6.2.3 Potential Solutions

Social biases inherent in such models were a central topic for discussion in the above mentioned convention, summarized by Tamkin et al. [88]. While discussing, the participants explored numerous ways to combat such detrimental biases in language models.

For example, some suggested pre-processing the initial training data to minimize bias in advance, or fine-tune the trained model with data of high quality - free of such issues. Another proposed mechanism involved using a separately-trained model that filters machine-generated content, or utilizing the model's existing capabilities to improve its outputs (for instance, via thoughtful prompt design). To improve training procedures, some of the participants suggested training models to possess greater awareness of factual information, or employing reinforcement learning with human input to guide the model towards more accurate and pleasant outputs. Devising more comprehensive sets of "bias tests" for models to undergo prior to deployment is another approach that was suggested.

6.3 Misuse

Large-scale pre-trained deep learning models, such as CLIP and GPT, offer vast potential for meaningful zero-shot (or few-shot) generalization. However, predicting the potential misuse of such highly capable models can be challenging, particularly when they are adapted for novel contexts or purposes beyond their original scope. To address these concerns, Brown et al. [9] recommend applying traditional security risk assessment frameworks, which involve specific steps outlined in the work of Ross

[78]. In this section, we will provide an overview of their analysis of potential misuse applications for GPT-3, as well as a discussion of threat actor analysis and external incentive structures.

According to Brown et al. [9], any socially harmful activity that relies on generating text could be augmented by powerful language models. This includes activities such as phishing, misinformation, spam, and more. As language models improve and produce higher quality text, these applications become more feasible and may increase in efficacy. GPT-3, in particular, is a concerning milestone due to its ability to generate several paragraphs of synthetic content that is difficult to distinguish from human-written text.

In terms of threat actor analysis, the authors mention that monitoring of forums and chat groups where misinformation tactics, malware distribution, and computer fraud are discussed reveals that low and mid-skill actors have shown interest in language models, but there have been few successful deployments since the release of GPT-2 in spring of 2019 (until the publication of GPT-3 by Brown et al. [9]). The assessment they mention is that language models may not be worth investing significant resources in, as there have been no convincing demonstrations that they are significantly better than current methods for generating text, and because methods for "targeting" or "controlling" the content of language models are still in their early stages.

External incentive structures also play a role, with each threat actor group relying on a set of tactics, techniques, and procedures (TTPs) to accomplish their agenda. Using language models to augment existing TTPs could result in a lower cost of deployment, but the stochastic outputs of language models still require human filtering, limiting scalability.

After analyzing the language model and studying the landscape and threat actors, Brown et al. [9] anticipate that AI researchers will eventually create language models that are more predictable and controllable, making them more appealing to malicious actors. This, in turn, is likely to present challenges to the broader research community. To address this concern, the authors plan to undertake a combination of mitigation research, prototyping, and collaboration with other developers.

6.4 More Food for Though

While these are the primary concerns related to large-scale pre-trained models, many more issues arise when delving deep into their characteristics. In this section, we will briefly outline some of them.

Brown et al. [9] draw attention to the massive amounts of computation necessary for practical pre-training, which consumes a significant amount of energy. It is crucial to consider the cost and efficiency of these models, as suggested by Schwartz et al. [81]. It is also important to consider how these resources are spread out over the model's lifetime, which will be fine-tuned for specific tasks and repetitively queried for a variety of purposes. While GPT-3 is efficient once trained (according to Brown et al. [9], generating 100 pages of content from a trained model may cost around $0.4kW-hr$ which is equivalent to a few cents), costs can be further reduced by improving algorithms or employing methods like model distillation (Liu et al. [50]).

Radford et al. [73] discuss the potential applications of their CLIP model in downstream tasks with significant societal sensitivity, such as surveillance. They hope to guide the research community towards the future impacts of increasingly general-purpose computer vision models and help establish standards and checks around these systems. They emphasize that while the model can identify the primary object in a scene with ease, it performs randomly when attempting to identify more subtle details. Additionally, Radford et al. [73] note that numerous large datasets and highly performing supervised models are available for many in-demand surveillance tasks, making CLIP less appealing for such uses.

To address most issues related to these models, both Tamkin et al. [88] and Radford et al. [73] suggest to continuously test the model's capabilities for potential problems such as bias, misuse, safety concerns, etc., at the development stage, which is referred to as "red-teaming" by Tamkin et al. [88]. Radford et al. [73] also recommend identifying tasks with significant sensitivity and creating test suites to

evaluate systems like CLIP. In terms of misuse or disinformation, one proposal mentioned in Tamkin et al. [88] is to employ techniques of cryptography to authenticate media in response to the existence of systems like GPT-3.

Despite these issues, certain participants in the aforementioned gathering of researchers, summarized by Tamkin et al. [88], emphasized that specific use cases should be avoided due to the limitations of existing techniques, as text generation applications differ greatly in terms of their risk. The risk of detecting regular expressions in text is much less significant than, for example, the risk of managing a suicide hotline using a language model.

7 Summary

7.1 What We've Covered

Throughout this seminar we've explored the idea of training powerful and transferable deep learning systems, by utilizing large-scale internet-based datasets in an unsupervised fashion. Our examinations have centered on the applicability of such systems, and the remarkable abilities they possess. We focused on two powerful series of models that have revolutionized the field of artificial intelligence:

- The GPT family (Radford et al. [71, 72]; Brown et al. [9]), consisting of Transformer-based language models trained on large internet corpora, which exhibited phenomenal lingual and knowledgeable capabilities - much beyond those expected from a mere language model.
- The CLIP series (Radford et al. [73]) - connecting visual data and natural language semantics, in a flexible way that enabled simple zero-shot transfer to a wide variety of tasks. CLIP's multimodal feature space was of great interest for us, as it revealed remarkable representational capabilities and fascinating characteristics.

In addition to our investigations of the models' direct transfer capabilities, we've also glimpsed into more creative means by which other systems can harness the functionality the pre-trained models provide, to solve more complex tasks. Specifically, we've looked into 3 applications:

1. ClipCap (Mokady et al. [58]) - which combined the powerful feature space of CLIP (trained to represent a wide variety of lingual concepts), with the generative language model of GPT-2, to generate captions for images.
2. DietNeRF (Jain et al. [33]) - which harnessed CLIP's semantic consistency to guide the training of a NeRF system for novel view synthesis, when too few examples are provided.
3. The work of (de Vries and Nissim [15]) - that demonstrated how the English language model of GPT can be transformed into a language model for any other language with minimal requirements for additional compute.

Finally, we've conducted a quick survey of the broader impacts that the existence of such powerful systems may have - analyzing the social biases that they've inherited from their respective datasets, their potential for malicious use, and more. We've also discussed several means by which these issues may be tackled, as outlined by the works of Radford et al. [73], Brown et al. [9], and Tamkin et al. [88].

7.2 What's Ahead

CLIP and GPT have been extensively utilized by many other researchers for a variety of systems. The applications discussed in this paper are far from exhausting the possibilities that these powerful models enable. Some of the exciting applications that have not been covered in this seminar include: StyleClip (Patashnik et al. [68]) - editing imagery using natural language, Text2Mesh (Michel et al. [53]) - stylizing 3D models based on natural language, the work of Xu et al. [103] proposing a novel mechanism for zero-shot semantic segmentation using CLIP, DALL·E 2 (Ramesh et al. [74]) of OpenAI - generating and altering images from natural language descriptions, and more. It is of great belief that many more creative applications are still left to be discovered, and may be the subject of future researches.

Additionally, Brown et al. [9], Radford et al. [73], and Tamkin et al. [88] - all mention that as technology advances, more efforts must be invested in analyzing the shortcoming, capabilities, and biases inherent in such large internet-based models, while developing laws and regulations regarding these issues - for all to follow.

As previously mentioned, Brown et al. [9] and Radford et al. [73] both argue that there is still plenty of room for improvement in both CLIP's and GPT's few-shot learning capabilities. CLIP's few-shot learning mechanism is still rather primitive (linear classifiers fit on top of its feature space), and although GPT's performance was mostly impressive in these settings - both approaches still lag behind human learning (especially for one-shot learning - in which humans' improvement was significantly more impressive), even though humans are exposed to substantially less data in their lifetime than the pretrained models are. Brown et al. [9] also mention that GPT even struggles on a variety of trivial tasks, with performance near random.

It is clear that more research should be invested in improving learning methodologies to bring humanity closer to true *Artificial General Intelligence* (Goertzel [24]), but without a doubt - large-scale pretrained models in deep learning play a central role in our path towards it.

References

[1] H. Agrawal, K. Desai, Y. Wang, X. Chen, R. Jain, M. Johnson, D. Batra, D. Parikh, S. Lee, and P. Anderson. nocaps: novel object captioning at scale. *International Conference on Computer Vision*, pages 8947–8956, 2019.

[2] F. Almeida and G. Xexéo. Word embeddings: A survey. *arXiv preprint arXiv:1901.09069*, 2019.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] L. F. Barrett. The theory of constructed emotion: an active inference account of interoception and categorization. *Social cognitive and affective neuroscience*, 12(1):1–23, 2017.

[5] J. Berant, A. K. Chou, R. Frostig, and P. Liang. Semantic parsing on freebase from question-answer pairs. In *Conference on Empirical Methods in Natural Language Processing*, 2013.

[6] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, R. Soricut, L. Specia, and A. Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL `https://aclanthology.org/W14-3302`.

[7] Borana. Applications of artificial intelligence & associated technologies. *Science [ETEBMS-2016]*, 5(6), 2016.

[8] M. Brems. A beginner's guide to the clip model, 2021. URL `https://www.kdnuggets.com/2021/03/beginners-guide-clip-model.html`.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL `https://doi.org/10.1023/A:1007379606734`.

[11] G. Cheng, J. Han, and X. Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[12] K. Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[13] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[14] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In J. Quiñonero-Candela, I. Dagan, B. Magnini, and F. d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[15] W. de Vries and M. Nissim. As good as new. how to successfully recycle english gpt-2 to make models for other languages. *arXiv preprint arXiv:2012.05628*, 2020.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[17] K. Desai and J. Johnson. Virtex: Learning visual representations from textual annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11162–11173, 2021.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html, 2017.

[21] P. Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

[22] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[23] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[24] B. Goertzel. Artificial general intelligence: concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 5(1):1, 2014.

[25] G. Goh et al. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] D. Hebb. The organization of behavior. *New York*, 1949.

[28] P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[29] F. Hill, A. Bordes, S. Chopra, and J. Weston. The goldilocks principle: Reading children's books with explicit memory representations, 2015. URL `https://arxiv.org/abs/1511.02301`.

[30] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

[31] Inamdar. Gpt-3 — a revolution in ai, 2021. URL `https://medium.com/analytics-vidhya/gpt-3-a-revolution-in-ai-103546558d76`.

[32] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[33] A. Jain, M. Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5885–5894, October 2021.

[34] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*, 2017.

[35] A. Joulin, L. Van Der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 67–84. Springer, 2016.

[36] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[37] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[38] K. Kärkkäinen and J. Joo. Fairface: Face attribute dataset for balanced race, gender, and age. *arXiv preprint arXiv:1908.04913*, 2019.

[39] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[40] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[42] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[43] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

[44] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[45] H. J. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'12, page 552–561. AAAI Press, 2012. ISBN 9781577355601.

[46] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020. URL `https://arxiv.org/abs/2005.11401`.

[47] A. Li, A. Jabri, A. Joulin, and L. Van Der Maaten. Learning visual n-grams from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4183–4192, 2017.

[48] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[49] Y.-P. Lin and T.-P. Jung. Improving eeg-based emotion classification using conditional transfer learning. *Frontiers in human neuroscience*, 11:334, 2017.

[50] X. Liu, P. He, W. Chen, and J. Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*, 2019.

[51] B. McCann, N. S. Keskar, C. Xiong, and R. Socher. The natural language decathlon: Multitask learning as question answering, 2018. URL `https://arxiv.org/abs/1806.08730`.

[52] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12, 2006.

[53] O. Michel, R. Bar-On, R. Liu, S. Benaim, and R. Hanocka. Text2mesh: Text-driven neural stylization for meshes, 2021. URL `https://arxiv.org/abs/2112.03221`.

[54] S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot, et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*, 2021.

[55] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[56] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2020.

[57] T. M. Mitchell. *Machine learning*, volume 1(9). McGraw-hill New York, 1997.

[58] R. Mokady, A. Hertz, and A. H. Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[59] Y. Mori, H. Takahashi, and R. Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *First international workshop on multimedia intelligent storage and retrieval management*, pages 1–9. Citeseer, 1999.

[60] N. Mostafazadeh, M. Roth, A. Louis, N. Chambers, and J. Allen. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain, Apr. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-0906. URL `https://aclanthology.org/W17-0906`.

[61] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.441. URL `https://aclanthology.org/2020.acl-main.441`.

[62] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[63] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[64] A. Pai. What is tokenization in nlp? here's all you need to know, 2020. URL `https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/`.

[65] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[67] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012. doi: 10.1109/CVPR.2012.6248092.

[68] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[69] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[70] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1): 145–151, 1999.

[71] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training, 2018.

[72] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[73] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[74] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL `https://arxiv.org/abs/2204.06125`.

[75] S. Reddy, D. Chen, and C. D. Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[76] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiský, and P. Blunsom. Reasoning about entailment with neural attention, 2015. URL `https://arxiv.org/abs/1509.06664`.

[77] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[78] R. Ross. Guide for conducting risk assessments, 2012-09-17 2012.

[79] S. Ruder, I. Vulić, and A. Søgaard. A survey of cross-lingual word embedding models. *J. Artif. Int. Res.*, 65(1):569–630, may 2019. ISSN 1076-9757. doi: 10.1613/jair.1.11640. URL `https://doi.org/10.1613/jair.1.11640`.

[80] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, aug 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL `https://doi.org/10.1145/3474381`.

[81] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *Communications of the ACM*, 63(12): 54–63, 2020.

[82] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[83] P. Sharma, N. Ding, S. Goodman, and R. Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning, 2018.

[84] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL `https://aclanthology.org/D13-1170`.

[85] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[86] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[87] R. Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[88] A. Tamkin, M. Brundage, J. Clark, and D. Ganguli. Understanding the capabilities, limitations, and societal impact of large language models, 2021. URL `https://arxiv.org/abs/2102.02503`.

[89] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[90] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

[91] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, jan 2016. ISSN 0001-0782. doi: 10.1145/2812802. URL `https://doi.org/10.1145/2812802`.

[92] H. Touvron, A. Vedaldi, M. Douze, and H. Jégou. Fixing the train-test resolution discrepancy. *Advances in neural information processing systems*, 32, 2019.

[93] A. M. Turing and J. Haugeland. Computing machinery and intelligence. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, pages 29–56, 1950.

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[95] B. S. Veeling, J. Linmans, J. Winkens, T. Cohen, and M. Welling. Rotation equivariant cnns for digital pathology. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II 11*, pages 210–218. Springer, 2018.

[96] D. Ventura and S. Warnick. A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 19, 2007.

[97] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018. URL `https://arxiv.org/abs/1804.07461`.

[98] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019.

[99] A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[100] A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[101] J. J. Webster and C. Kit. Tokenization as the initial phase in nlp. In *COLING 1992 volume 4: The 14th international conference on computational linguistics*, 1992.

[102] A. Williams, N. Nangia, and S. R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

[103] M. Xu, Z. Zhang, F. Wei, Y. Lin, Y. Cao, H. Hu, and X. Bai. A simple baseline for open-vocabulary semantic segmentation with pre-trained vision-language model, 2021. URL `https://arxiv.org/abs/2112.14757`.

[104] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. P. Langlotz. Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pages 2–25. PMLR, 2022.

[105] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, Los Alamitos, CA, USA, dec 2015. IEEE Computer Society. doi: 10.1109/ICCV.2015.11. URL `https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.11`.