What is OpenAI? According to their website:

OpenAI’s mission is to ensure that artificial general intelligence (AGI)—by which we mean highly autonomous systems that outperform humans at most economically valuable work—benefits all of humanity. We will attempt to directly build safe and beneficial AGI, but will also consider our mission fulfilled if our work aids others to achieve this outcome.

What does a Google Search of OpenAI return?
OpenAI’s GPT-3 Wrote This Short Film—Even the Twist at the End

A robot wrote this entire article. Are you scared yet, human?

**GPT-3**

We asked GPT-3, OpenAI’s powerful new language generator, to write an essay for us from scratch. The assignment? To convince us robots come in peace.
Elvis back from the dead? Artificial intelligence is used to create eerie 'deepfake' pop songs that sound like they are being sung by deceased stars

- Neural network made by US firm OpenAI imitates musicians both living and dead
- 'Jukebox' generates approximates of pop songs ranging from Elvis to The Beatles
- The weird audio samples are available to listen for free in OpenAI's online library
OpenAI: A Quick Glance

• AI research laboratory formed in 2015
• Founded by Elon Musk, Sam Altman, Ilya Sutskever and others
• 120 employees as of 2020
• Recently partnered with Microsoft after a 1 billion dollar investment in 2019
OpenAI Projects

• GPT, GPT-2, GPT-3
• Image GPT
• Jukebox
• Other Projects
  • Gym/Deep Representation Learning
  • Microscope
What is GPT?

- GPT stands for Generative Pre-Trained
  - Pre-train a language model on a HUGE corpus of data and then fine-tune
- GPT uses Transformer Decoder blocks
  - Attention is computed using only the words preceding the given word, outputting one word at a time

Gif from http://jalammar.github.io/illustrated-gpt2/
What is a Decoder Block?

The Transformer

Decoder Block

- Feed Forward Neural Network
- Encoder-Decoder Self-Attention
- Masked Self-Attention

Input

<table>
<thead>
<tr>
<th></th>
<th>&lt;s&gt;</th>
<th>robot</th>
<th>must</th>
<th>obey</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Image from http://jalammar.github.io/illustrated-gpt2/
Decoder Block: Masked Self Attention

Image from http://jalammar.github.io/illustrated-gpt2/
Stack only Transformer Decoder Blocks and remove Encoder-Decoder layer

Image from http://jalammar.github.io/illustrated-gpt2/
Simplest way to Allow GPT to operate:
Let it “ramble”
Add 1\textsuperscript{st} output to our input and predict the 2\textsuperscript{nd} token:
Slight Differences: GPT-2 vs. GPT

- Layer Normalization was moved to the input of each sub-block (similar to a pre-activation residual network)
- Another additional layer is added after the final self-attention block
- A modified initialization which accounts for the accumulation on the residual path with model depth is used.
Original GPT:

Figure 1: (left) Transformer architecture and training objectives used in this work. (right) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.
4 different sizes of GPT-2:

- **GPT-2 SMALL**
  - Model Dimensionality: 768
  - 12
  - ... [Decoder]
  - 1
  - [Decoder]

- **GPT-2 MEDIUM**
  - Model Dimensionality: 1024
  - 24
  - ... [Decoder]
  - 2
  - [Decoder]
  - 1
  - [Decoder]

- **GPT-2 LARGE**
  - Model Dimensionality: 1280
  - 36
  - ... [Decoder]
  - 4
  - [Decoder]
  - 3
  - [Decoder]
  - 2
  - [Decoder]
  - 1
  - [Decoder]

- **GPT-2 EXTRA LARGE**
  - Model Dimensionality: 1600
  - 48
  - ... [Decoder]
  - 6
  - [Decoder]
  - 5
  - [Decoder]
  - 4
  - [Decoder]
  - 3
  - [Decoder]
  - 2
  - [Decoder]
  - 1
  - [Decoder]
Differences between GPT-2 and GPT-3:

• GPT-3 uses alternating dense and locally banded sparse attention patterns in the layers of the transformer

• 175 billion parameters vs. 1.5 billion in GPT-2

• Training using the lowest cost cloud provider estimated to cost $4.6 million and take 355 years.
Zero-shot vs. One-shot vs. Few-shot

- **Few-shot** – aka in-context learning where as many demonstrations are provided that will fit into a context-window (between 10-100 in GPT-3)
- **One-shot** – only one demonstration is provided in addition to natural language instructions
- **Zero-shot** – only instructions in natural language are provided
Results of GPT-3 on Lambada

Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.
GPT DEMO
Quick Intro to Image GPT

• After success with GPT on NLP, why not try it to generate images?
• Like GPT, there is a pre-training stage:
  • Autoregressive, BERT objectives explored
  • Apply sequence Transformer architecture to predict pixels instead of language tokens
• and a fine-tuning stage:
  • adds a small classification head to the model, used to optimize a classification objective and adapts all weights.
Image GPT Approach Overview

Figure 1. An overview of our approach. First, we pre-process raw images by resizing to a low resolution and reshaping into a 1D sequence. We then chose one of two pre-training objectives, auto-regressive next pixel prediction or masked pixel prediction. Finally, we evaluate the representations learned by these objectives with linear probes or fine-tuning.
Quick Intro to Jukebox

• A model that generates music with singing
• VQ-VAE model:
  • compresses audio into a discrete space, with a loss function designed to retain the maximum amount of musical information, while doing so at increasing levels of compression
  • downsamples extremely long context inputs to a shorter-length discrete latent encoding using a vector quantization
  • First applied to large scale image generation in Generating Diverse High Fidelity Images
Quick Intro to Jukebox (continued)

• Training
  • VQ-VAE has 2 million parameters and is trained on 9-second audio clips on 256 V100 for 3 days
  • The upsamplers (which recreate lost information at compression) have one billion parameters and are trained on 128 V100s for 2 weeks, and
  • the top-level prior (needed to learn to generate samples) has 5 billion parameters and is trained on 512 V100s for 4 weeks
Figure 1. We first train three separate VQ-VAE models with different temporal resolutions. At each level, the input audio is segmented and encoded into latent vectors $h_t$, which are then quantized to the closest codebook vectors $e_{z_t}$. The code $z_t$ is a discrete representation of the audio that we later train our prior on. The decoder takes the sequence of codebook vectors and reconstructs the audio. The top level learns the highest degree of abstraction, since it is encoding longer audio per token while keeping the codebook size the same. Audio can be reconstructed using the codes at any one of the abstraction levels, where the least abstract bottom-level codes result in the highest-quality audio, as shown in Figure 4. For the detailed structure of each component, see Figure 7.
What is ONNX? According to their website:

We believe there is a need for greater interoperability in the AI tools community. Many people are working on great tools, but developers are often locked in to one framework or ecosystem. ONNX is the first step in enabling more of these tools to work together by allowing them to share models.
Background on ML frameworks

• Deep learning with neural networks is accomplished through computation over dataflow graphs.

• These graphs serve as an Intermediate Representation (IR) that
  • capture the specific intent of the developer's source code, and
  • are conducive for optimization and translation to run on specific devices (CPU, GPU, FPGA, etc.).
Why do we need ONNX?

• Each framework has its own proprietary representation of these dataflow graphs
  • For example, PyTorch and Chainer use dynamic graphs
  • Tensorflow, Caffe2 and Theano use static graphs
• But, each framework provides similar capabilities:
  • Each is just a siloed stack of API, graph and runtime
• Although one framework may be best for one stage of a project’s development, another stage may require a different framework
How does ONNX do this?

• ONNX provides a definition of an extensible computation graph model, as well as definitions of built-in operators and standard data types.

• Each computation dataflow graph is structured as a list of nodes that form an acyclic graph.
How does ONNX do this? (continued)

• Nodes have one or more inputs and one or more outputs.
• Each node is a call to an operator.
• The graph also has metadata to help document its purpose, author, etc.
• Operators are implemented externally to the graph, but the set of built-in operators are portable across frameworks.
How does ONNX do this? (continued)

• Every framework supporting ONNX will provide implementations of these operators on the applicable data types.
Example from keras to ONNX:

```python
In [ ]:
import onnxmltools
from keras.models import load_model

In [ ]:
# Update the input name and path for your Keras model
input_keras_model = 'model.h5'

# Change this path to the output name and path for the ONNX model
output_onnx_model = 'model.onnx'

In [ ]:
# Load your Keras model
keras_model = load_model(input_keras_model)

# Convert the Keras model into ONNX
onnx_model = onnxmltools.convert_keras(keras_model)

# Save as protobuf
onnxmltools.utils.save_model(onnx_model, output_onnx_model)
```
Open AI Links

OpenAI API request
GPT-3 wrote this short film
GPT-3 writes Guardian article
GPT-3 Reddit account
Write with Transformer (hugging face)
AllenNLP (generate sentences using GPT-2)
Text Generation API (generate more text)
OpenAI Soundcloud
https://jukebox.openai.com/
OpenAI github
ONNX Links

ONNX github
ONNX website
ONNX tutorials