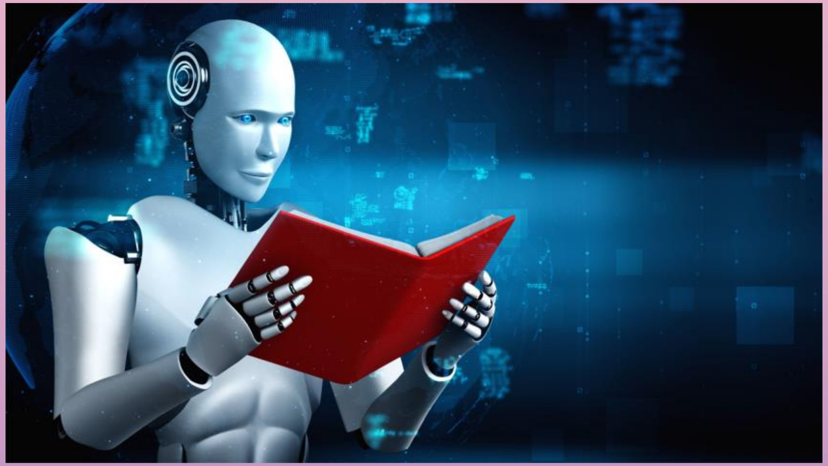
Generating images with AI

A basic introduction by Bruno A. Lomelirubi Vargas

Purpose

Contrary to popular belief, what I used to believe before this project, AI does not just understand words and process them like humans do.

So, I set out to discover how this meme makes sense.



How does AI understand text?

Steps:

1. Word Embeddings
2. Self-Attention Mechanism.
3. Positional Encoding.
4. Feedforward layers.
5. Training and Tokenisation.
6. Optimisation.
7. Fine-Tuning.

Understanding the text.

Embedding

$$\text{similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Embedding refers to the process of identifying words as vectors that represent characteristics of their meaning, i.e. context. Additionally, it computes the similarity among words through its cosine similarity.

Attention score

Each word will have three matrices: query, key (best change of basis to assess similar words) and a value (best change of basis to predict the next word.)

$$\text{Attention Score}_{i,j} = \frac{\exp(Q_i \cdot K_j)}{\sum_k \exp(Q_i \cdot K_k)}$$

Positional encoding

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

As vector and matrix operations are performed in parallel, we use this step to assign patterns to positions. Nearby words have similar position scores.

Processing the text.

Feedforward

Now, we pass our pre-processed input through our neural network. Chat GPT-4 has about 120 hidden layers with ReLU activation functions.

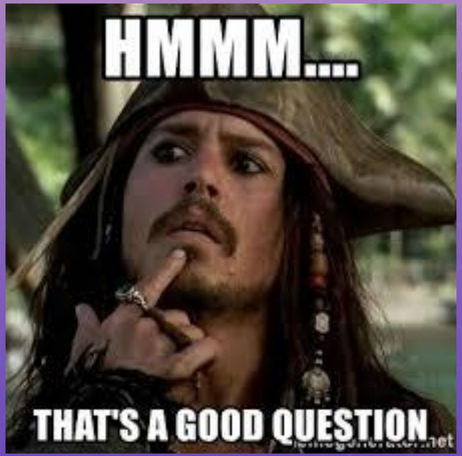
Optimisation

We minimise the Cross-Entropy Loss Function with a gradient descent algorithm

Fine-Tuning with Reinforcement Learning

We can generate another LLM that gives inputs and adjusts parameters by reinforcing correct answers.

So, what did I really do?



I coded these steps into a silly AI in python.

```python
# Model with separate pathways for color, shape, and size
class ShapeNetSeparate(nn.Module):
    def __init__(self):
        super(ShapeNetSeparate, self).__init__()

        # Embedding layers for each input component
        self.color_embedding = nn.Embedding(len(color_vocab), 8)
        self.shape_embedding = nn.Embedding(len(shape_vocab), 8)
        self.size_embedding = nn.Embedding(len(size_vocab), 8)

        # Processing layers for each component
        self.color_fc = nn.Sequential(nn.Linear(8, 16), nn.ReLU())
        self.shape_fc = nn.Sequential(nn.Linear(8, 16), nn.ReLU())
        self.size_fc = nn.Sequential(nn.Linear(8, 16), nn.ReLU())

        # Final layers combining all components
        self.fc_combined = nn.Linear(16 * 3, 16)
        self.fc3_color = nn.Linear(16, 3)  # RGB color prediction
        self.fc3_shape = nn.Linear(16, 3)  # Shape classification prediction
        self.fc3_size = nn.Linear(16, 1)   # Size prediction (continuous)

    def forward(self, color_input, shape_input, size_input):
        # Process each component independently
        color_feat = self.color_fc(self.color_embedding(color_input))
        shape_feat = self.shape_fc(self.shape_embedding(shape_input))
        size_feat = self.size_fc(self.size_embedding(size_input))

        # Concatenate features from each pathway
        combined_feat = torch.cat((color_feat, shape_feat, size_feat), dim=1)

        # Final processing and separate outputs
        x = torch.relu(self.fc_combined(combined_feat))
        color_output = torch.sigmoid(self.fc3_color(x))  # RGB output
        shape_output = self.fc3_shape(x)  # Shape classification
        size_output = self.fc3_size(x)  # Size output (regression)

        return color_output, shape_output, size_output
```

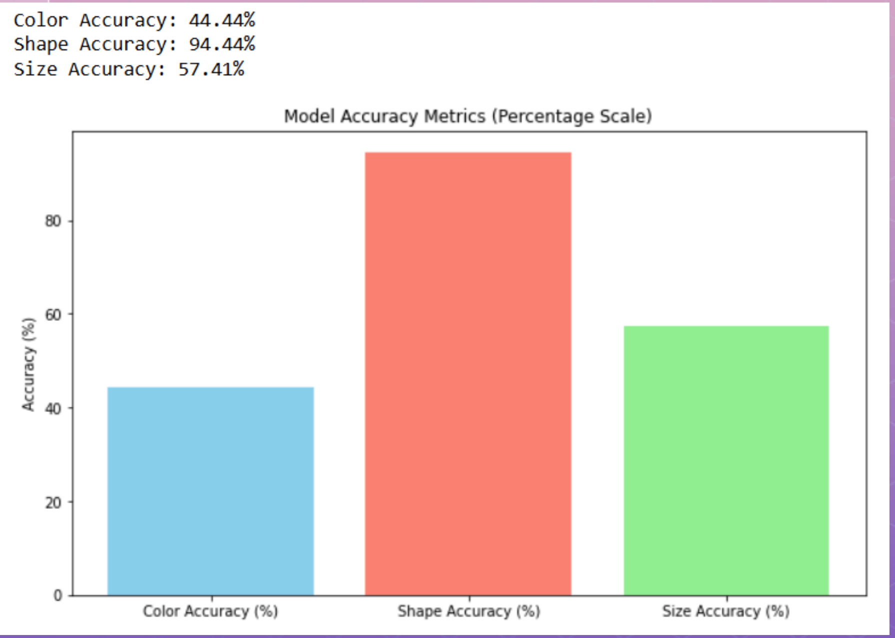
What does my AI do?

This is simple, it takes natural language asking to generate shapes with a specific size and colour and then generates them.

My data

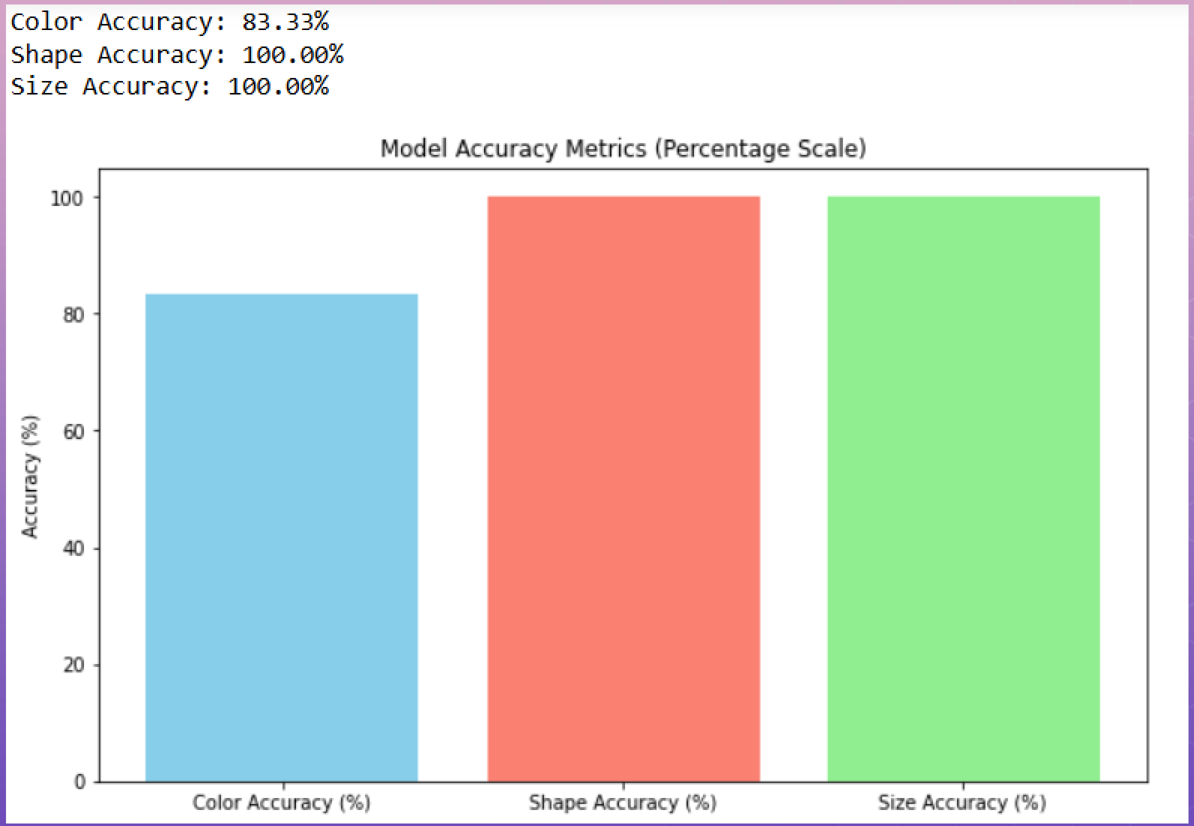
```python
# Sample dataset with separate components for color, shape, and size
data = [
    ("red", "circle", "large"),
    ("green", "square", "medium"),
    ("blue", "triangle", "small"),
    ("yellow", "circle", "small"),
    ("purple", "square", "large"),
    ("orange", "triangle", "medium")
]

# Define vocabularies for color, shape, and size
color_vocab = {"red": 0, "green": 1, "blue": 2, "yellow": 3, "purple": 4, "orange": 5}
shape_vocab = {"circle": 0, "square": 1, "triangle": 2}
size_vocab = {"small": 0, "medium": 1, "large": 2}

# Define RGB values for colors, one-hot encoding for shapes, and numerical values for sizes
color_values = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 1, 0], [0.5, 0, 0.5], [1, 0.5, 0]])
shape_labels = ["circle", "square", "triangle"]
size_values = np.array([20, 30, 40])  # Small, Medium, Large
```

Initial results



After 3000 epochs, we have a loss of 2.5 X 10^(-4), but the predictions were still inaccurate.

What did I do then? I retrained.



Identify areas for improvement.

I identified that size and colour seemed to be the most problematic for the model.

Generate focus data.

So, I generated more data with a focus on these areas and retrained the model.

Conclusions

I am pretty disappointed that AI is not as big of a monster as I thought it was. It is more like a friendly giant, because of its size.
On the other hand, my model might not have much applicability, but I tried to make it as simple and visual such that other people like me can start learning about AI just as I did.

Thank you!