Mini-Project 1

Noah Foster

CSCI 2952N

I really liked this miniproject and found it is relevant in many ways to my own research and even my final project from deep learning! I will walk through the project and discuss some of the ways that I think my implimentation stands out and some of my justifications and logics for the code I wrote. Hopefully this ordering is relatively similar to the Jupyter Notebook I submitted. I should note that I tried to keep the envirnonment as close to the original as possible, but I did have to make some changes to get the code to run on my machine. I also wanted to use the library Spacy as well as a couple others. Anything not included in the email this report surely arrived in is probably in this GitHub Repository. In the name of making all the code work, I have also renamed gpt3-sandbx to gpt3sandbox.

Task 1: Zero Shot

Note: The project guidelines mention CIFAR-10, while the provided notebook uses CIFAR-100. I use both just for fun.

The logic for this part is very similar to that of the provided Interacting with CLIP notebook. I am using a Macbook Air for this project and thus struggle to run anything except ResNet-50, so further experiments are solely done with ResNet-50. I tried the other but mainly just found the difference in time. The core logic is that CLIP is trained to embed images and captions so that the cosine similarity between the two is high when they have similar content. So we build a caption that corresponds to each class and see which classes align best. I tried a couple methods of building captions and found some interesting things. The format of caption that works best for CIFAR-10 is not the one that works best for CIFAR-100. It is also interesting that in both cases, ending the sentence with a period does not hurt and in fact helps despite making the tokenization a little more strange.

My zero shot accuracy was 68% with the format of prompt provided in the stencil code and I got that up to around 76.4% with a slightly modified prompt format. For CIFAR-100 I got much lower accuracy, around 35%. I tried a couple different formats for the prompt and found that the best was to use the format that had a couple more words in it. This brought me to a validation accuracy of almost 40. This was borne across both the training and validation data so I really can't explain it. I also check the skew of my samples and find totally reasonable results.

Task 2: Linear Probing

I once again repeated this task for both of the CIFAR datasets. I also chose two methods for finding a probe. The method that I suspect most will use is to train up a single linear layer. While this of course makes sense, we are thinking about convex optimization here and so with the MSE loss function, we actually have a closed form solution. I tried both an got wildly different results. I spent a long time trying hyperparameters for the single linear layer and found that a relatively high learning rate and batchsize did best for both the CIFAR datasets.

On CIFAR-10, my linear layer achieves 87% validation accuracy while the regression is stuck at 21.4. The training accuracy of both is near 100 (or actually 100 for the regression).

On CIFAR-100, my linear layer achieves 44% validation accuracy while the regression is stuck at 4.4. The training accuracy of both is near 100 (or actually 100 for the regression).

It is very interesting to me that I get such different results with SGD and closed form regression. I am not sure why this is the case, but I suspect it is unrelated to MSE vs Cross Entropy.

Task 3: Images to Incoherent Text

We would like to find a process by which to iteratively try captions with CLIP in order to eventually end up with one that is a good. I think I have a novel way of implementing this. I did not want to implement exactly what the Socratic Paper had done, but I wanted to keep their philosophy intact. To do this, I decided to harness all the training captions I have available. I use Spacy to pull out the nouns from the training captions, then also grab all the adjectives used to describe those nouns. This way I can first check which nouns are in the image, then check just just that noun, which of the adjectives work best. I slow this does substantially by checking all the adjectives for the top 5 nouns, but I think it is worth it, because it does surprisingly well. I think that this approach sets me apart from any other student on its own. I think it is also more in the spirit of the data-centric era of machine learning. In the socratic paper they provide an extensive framework that directs the models line of questioning but I am using a fully learned approach that asks first what objects are in the image (using the prompt format I learned was best in task 1). Then I use my learned knowledge of what typically modifies these object to direct the rest of my questioning. I think this is an exciting approach.

Task 4: Prompting GPT-3

I think that my approach on the above task sets up a very nice way to build context and prompts for GPT-3. I simply check other captions that have the desired subjects and attributes and use the captions of those images as my context. This could be make even fancier by incorporating the information about how well these captions were rated, but I don't think that would help right now as I am only interested in whether the captions could theoretically describe an image with the desired subject, not whether they actually describe the one in question. I have managed something rather clever in the construction of the prompts. By elaborately correlating which images gave me assosiactions between particular nouns and verbs, I can use the lematized prompts for finding information out in new images, but use the fully gramatical prompts for generating new captions. This is a very nice way to get the best of both worlds.

Task 5: GPT-3 as a Classifier

It turns out that everything comes together very well. In my notebook I randomly select 10 images to try it on. Having gone through many, I can say that about 1/3rd of the time, it toally nails it and the rest of the time it is just totally off the mark. I think this comes down mostly to my limited selection of nouns. There are really only the top 300 that make much sense to use so when I search the training captions for nouns, I cut my search off to these top 300. I think that performance could certainly be improved by expending this list, but that would also mean it would take longer to evaluate and therefore not be suitable for my Macbook Air.

Task 6: Quantitative Evaluation

In large part, this is a waste of time for my algorithm. I am most limited by computing power and so I cannot expand the amount of information that GPT-3 will be passed about this image. Theoretically, I could train my adjective-noun dictionary off the internet and end up with an amazing system to probe deep into the image. I cannot do that here so my performance will be massively limited. I can report that the average blue score for my algorithm's best geuss ata a caption against the 5 real captions is around the order of 0.056. This is from the sad fact that BlEU is hard to evaluate on such small amounts of text. Oh Well. The code in my notebook can be easily expanded to evaluate using more information passed into GPT-3 but I am simply limited in the amount of compute I have access to for CLIP.

I do want to make some qualitative judegements – this system is amazing. It is often very accurate and I think that it's really hard to overstate how impress the performance is given how limited a vocabulary it is working with. The highest score I have seen it produce is 0.6 (very high). I think this is impressive.

The question of which incontext examples matter is super interesting. I found that I was pretty incorporating tokens that used a way array of adjectives tended to confuse the model (this is not left in my code.) In my code it was very easy to add more examples that used different adjectives and nouns tended to confuse the model more than it helped. I think this is because keeping the examples on the theme of the image is important.

Because of errors in the OpenAI API, I have made my evaluation of 100 images modular so that I cache the data intermittently.

Task 7: Improvement!

In many ways, this system resembles the final project I did for 2470 last semester. For that class, I built a mutli-modal modal that used a facial recognition model that has been based on a foundation model but then further trained for a specific set of faces that extracted information about the content of a screenshot of the TV show The Office. It also used CLIP to pull a couple objects from the scene. It never checked Socratically to improve the information, but instead just fed that all straight into a fine-tuned GPT-2 that would use the information as well as context to produce more possible lines from the office. This is essentially the same as this project but does not use the CLIP language encoder. The link to that project is here. It points directly to many of the ways this could be taken further (ie more customizable prompts and more information passed into GPT-3). It also shows the potential of fine tuning on the language model to tailor the output to what we want. In that project, we saw that script-generation is not an inherent ability of language models. Much in the same way, single sentences to describe scenes is not what GPT-3 was primarily trained on. This means that we could improve the results if we could fine-tune GPT-3. Sadly due to incompatibilities in the environment, this is impossible.

Other thoughts on improvement, I think that the way that I have implemented Socrates is relatively novel in an of its self. In order to further improve on this system, we should start sending the embeddings from CLIP straight into GPT-3. Though this sounds impossible, it was shown to be possible with minimal training to CLIP and a LLM in the MAGMA Paper and then shown to be possible with no training to CLIP or the LLM in the LiMBeR Paper. This latter paper is by Jack Merullo, the Grad student for whom I am a research assistant. We are currently working on understanding the implications for large language models that this possibility shows. To understand the power of this approach, I have written a script based CLIP (Resnet 50) and GPT-J that uses the projection trained by Jack Merullo. The code and weights are available here. I will use my own code to run through LiMBeR.

The architecture of this model that I propose is as follows:

1. CLIP (RN50) with all but the final attention pool layer. This is a purely convolutional network that takes a 224x224x3 image and produces a 12x12x3072 output.

2. Let us flatten this into 144x3072.

3. We will use the weights from the LiMBeR paper to project each of the 144 3072 dimensional vectors individually into a 4096 dimensional vector.

4. This is the embedding dimension of GPT-J! Now we simply bypass GPT-J's embedding lookup table and feed in a 1x144x3096 tensor as though it is batch x sequence length x embedding dimension, just like a normal prompt for GPT-J.

5. There is a little bit of prompt engineering required because after those 144 vectors, we need to add the prompt that we want to use to generate the caption. This is a simple matter of adding the prompt to the end of the 144x3096 tensor. We will add the words "An image of" and let the model finish the rest.

Important Acknowledgments I would like to be very clear about what code of mine is completely original and where I have rewritten others' code. For this component of this project, I took the LiMBeR

codebase and rewrote it to be self-contained and pull directly from the CLIP and transformers libraries. This means that instead of mysteriously pulling all weights from a single pytorch file, I build all the models from their vanilla pre-trained versions. This was not a minor task and took nearly 6 hours. It is certainly worth noting that the code I wrote for task 7 is partially cannibalized from the MAGMA Paper and even uses the weights found in the LiMBeR Repository, however the structure of my code is completely different and was written expressly for this project. To emphasize the way in which what I have done is my own: I have reimplemented a model I currently study using only publicly available weights. I have also written a script to run the model and generate captions (and even evaluate).

Now that we've discussed the architecture and code, let's talk about the results. The speed of inference using the socratic method is severely limited by the number of captions necessary to embed. When projecting the embeddings directly into a large language model, we can run inference very fast (the same speed we would normally run inference on a model). Thus I will generate captions for the entire validation set. I will then use the same evaluation method as before to evaluate the captions (BLEU). We are now using a much smaller language model (6B params instead of 175B) but the pipeline of data from CLIP into the model is vastly improved. Instead of grasping at nouns via CLIPs language encoder, we are harnessing the massive dimensionality of the embedding space of GPT-J to project the CLIP embeddings into an interpretable (by GPT-J) subspace.