Multimodal Retrieval Augmented Generation for Instruction Manual Understanding via Contrastive Learning

Ali Hindy
Stanford University
557 Mayfield Avenue, Stanford, CA
ahindy@stanford.edu

William Denton
Stanford University
557 Mayfield Avenue, Stanford, CA
wdenton@stanford.edu@

Abstract

In this paper, we extend the commonly used text-based Retrieval Augmented Generation (RAG) architecture to the problem of downstream vision in the context of instruction manual understanding. We use a multimodal text and image embedding hybrid model for the retrieval task that outperforms vanilla RAG. We build a custom dataset for this task and introduce a contrastive learning framework for better retrieval results. We compare these results to baseline retrieval models, and discuss the implications in our overall question-answering RAG pipeline. Our learned retrieval system achieves **25% *top-1*** *accuracy and* **83% *top-5*** *accuracy on our custom dataset, significantly outperforming vanilla RAG.*

1. Introduction

Vision-based Retrieval Augmented Generation (RAG) models, in parallel with text-based LLMs, have undergone significant advancements in recent years [13, 8]. These strides offer fresh perspectives for tackling model hallucinations, knowledge limitations, and other challenges with large, pretrained language models. Among these challenges lie the comprehension and navigation of complex multi-step tasks such as instruction manuals, a perennial source of frustration.

Instruction manuals represent a ubiquitous source of guidance across various domains, yet they often pose significant challenges to comprehension and utility. The opacity and length of instruction manuals frequently leads to frustration and inefficiency in task completion. It can be incredibly tedious to find the information you need, despite it being present somewhere in the manual. Consequently, the development of a model capable of processing these manuals and offering instructive and relevant answers to specific queries emerges as a compelling endeavor.

Moreover, the complexity inherent in understanding instruction manuals underscores the necessity for multi-faceted comprehension mechanisms within such models. Beyond merely parsing textual instructions, these systems must exhibit robust reasoning capabilities to assimilate various components, including explicit instructional text, cautionary advisories concerning parts, and visual cues depicting component assembly. Thus, the efficacy of such models hinges not only on their ability to interpret textual information but also on their adeptness in synthesizing textual and visual inputs to generate coherent and informative responses.

1.1. Problem Statement

We propose a multimodal RAG pipeline for question answering and image understanding that leverages contrastive learning to improve retrieval. Currently, deep learning RAG methods exist in text form, but we extend this pipeline for image-based instruction understanding. Existing implementations exist for vanilla multimodal RAG, but these models struggle when given negative samples and consequently are not robust to noise in the context database. One improvement to vanilla RAG is to *train* the retrieval model with negative samples, but these models often require large amounts of compute during pretraining in order to learn to identify negative samples. We apply a contrastive learning approach that does not require large amounts of compute yet significantly outperforms vanilla RAG. We will compare these vanilla implementations to our adaptive deep learning approach.

A typical RAG pipeline includes the following components: a dataset of relevant context that we want to retrieve from, an embedding model, a vector database of embeddings for retrieval, a re-ranker model, a language model for generation, and a user prompt. As seen in Figure 2, the documents (in our case, the images from instruction manuals), are encoded using an image embedding model. The user's prompt is also encoded using a text encoder model. This prompt is then fed to the retriever, alongside the encoded documents, at which point the retreiver model de-

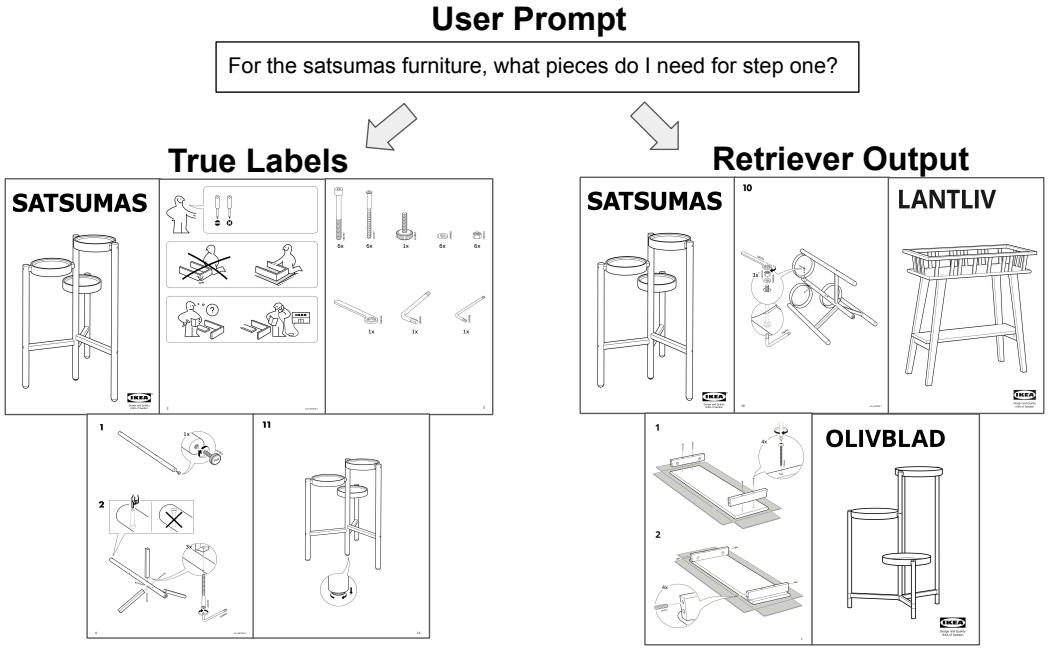

Figure 1: Sample prompt with true labels (left) as compared with the output from the vanilla retriever (right) using top-5 accuracy. Given a prompt, we aim to retrieve the most relevant images and pass them along as context to the generator. The true labels are hand labeled as part of our novel dataset. As we can see, the cosine similarity retriever identifies the first image correctly; however, the other four images are not correctly retrieved from the naive implementation of the retriever. We improve upon this retrieval mechanism with our learned approach and contrastive learning.

cides which contexts (i.e. pages) are most important using some similarity metric. The retriever then outputs this context either set of images or a single image, and these rankings are adjusted if a reranker model is in use. In our case, since the retrieved pages are already in order of relevancy, no reranker model is used. Using the prompt and relevant context, the model generates its answer to the question alongside the retrieved context.

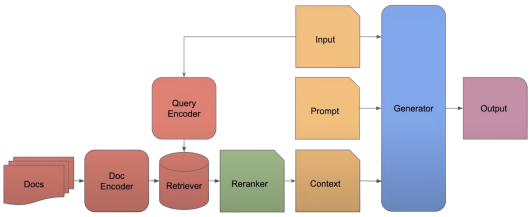

Figure 2: Example RAG Pipeline. Source: Contextual AI [12]

Our main technical contributions in this paper are threefold: 1) We introduce a novel dataset of instruction manuals and relevant prompts leveraging synthetic data for the retrieval task. 2) Rather than using an out of the box frozen retriever, we train a custom deep learning retriever that learns a combined embedding space given the image and text embeddings. 3) We use contrastive learning to distinguish between positive and negative samples in order to improve retrieval. The input to our retriever model will be a set of images consisting of an instruction manual, as well as the relevant query pertaining to that instruction manual, and the output of our model will be a prediction of the most relevant pages from that manual with regard to that given query. These relevant pages are fed to the context of a generator model (as well as the original prompt) to produce an answer.

2. Related Work

Text-Based Retrieval Language models often struggle with knowledge intensive or complex tasks, which has been mitigated in the text field by Lewis et al. with their Retrieval Augmented Generation architecture, which combines a generator model with a retriever in order to augment a user's prompt with relevant context fetched by the retriever. [13] RAG models provide a valuable framework for reliably fetching relevant information, which reduces the likelihood of hallucinations and incorrect information. The original RAG paper and subsequent improvements focus mainly on text augmentation by retrieving relevant text from embeddings in a large vector database. [11, 8] How-

ever, these methods use a frozen retriever, which means the retrieval model is not learning what relevant context looks like, which causes many of the issues with RAG, such as hallucination, struggling with false information, and negative rejection. [4] Contextual AI's RAG 2.0 addresses the frozen model issue by introducing an end-to-end pipeline for text-based retrieval augmented generation that outperforms state of the art models like GPT-4, Mixtral, and Claude 3 Opus. [12] Although RAG 2.0 is an architecture focused on text, we apply a *learned* retrieval approach to the world of computer vision and use similar frameworks in approaching the problem of instruction based retrieval.

Multimodal Retrieval Vanilla approaches exist for multimodal understanding as well. Most of the research in multimodal RAG focuses on different retrieval metrics or improved context databases, but they still leverage frozen retrieval. [3, 10] One recent multimodal model architecture, MuRAG, introduces a pretrained retriever that is also finetuned on the task-specific multimodal context. MuRag demonstrates significant performance increases in retrieval, yet still struggles significantly with negative samples. [5] In order to achieve state of the art retrieval performance, the authors had to apply significant amounts of compute with hand-labeled negative samples. Our method leverages contrastive learning in order to improve retrieval and recognition of negative samples in the dataset, demonstrating improved image understanding with significantly less compute.

Contrastive Learning A few different approaches currently exist that leverage contrastive learning for downstream vision tasks. One such approach leverages contrastive learning for image retrieval using vision transformers, but mainly focuses on category-level retrieval, not question answering. [7] COCA, introduced by Yu et al. leverages contrastive learning to create an image-text encoder-decoder model that achieves state of the art performance on zero shot image retrieval. [18] This architecture deploys contrastive loss between the text embedding and image embedding within each (text, image) pair, whereas our method finds negative samples within a context database (i.e. between different images). Furthermore, this architecture relies on pretraining a full encoder-decoder on all of ImageNet (and much more) in order to achieve state of the art performance. Our model requires significantly less compute and leverages contrastive loss in a vision-native setting (i.e. identifying negative samples given a set of images). The current research in contrastive learning applied to computer vision motivates the need for low compute, vision-native contrastive learning that allows for better image understanding in high complexity settings like instruction manual understanding.

3. Methods

We train our model with a custom dataset of Ikea instruction manual PDFs, sourced from Ikea.com and the dataset used by Wang et. al. in using Ikea manuals with computer vision to learn shape assembly.[16] During training, we feed our model a query and the most relevant image from the manual so it learns to predict high scores for those pairings. In addition, we feed it counterexamples, i.e. text queries and a page from the instruction manual which is irrelevant. We created a custom set of prompts that ask specific questions about these manuals, such as "Where would nail 8 fit in to component C of the Agam chair?" The output of the overall RAG model will be an answer from the generator, as well as the highest scoring relevant images from the retriever.

Retrieval Stage The primary task in developing our pipeline is training our retrieval model, as this is the predominant vision problem we need to solve to deploy our multimodal RAG pipeline. In creating our retrieval model, we build upon the work of large pretrained models which have created robust embedding spaces. The primary task of the retrieval model thus becomes to take 2 types of embeddings, a text embedding representing the user query and image embeddings representing the pages of the instruction manual, and decide on the relevancy of that image embedding for the given text query.

We experimented with a variety of embedding models for generating both text and image embedding inputs to our retrieval model. For the text embeddings, we used `MiniLM-L6-v2`, a general purpose text embedding model for sentences, and compared its performance to `word2vec` which embeds each word independently. [17, 14] We directly compare the performance of sentence-level embeddings to word-level embeddings in the retrieval task. For the image embeddings, we used the last layer of a pretrained `ResNet-18` and compare it to OpenAI's natively multimodal pre-trained embedding model, `CLIP`. [9, 15] We compare these two image embedding models on the retrieval task.

Given the text and image embeddings, we train our retrieval model to project both embeddings to the same space, and then use a relevant distance metric to evaluate relevancy, and then use that distance metric to output a score between 0 and 1 which represents the probability that the inputted image embedding is the relevant image for the inputted text embedding. Thus, a single training example for our model consists of $(x_{\text{i}}^{\text{image}}, x_{\text{i}}^{\text{query}}, y_i)$ where $x_{\text{i}}^{\text{image}}$ is the image embedding for a selected image, $x_{\text{i}}^{\text{query}}$ is the text embedding for the given query, and y_i is the true label (0 or 1) representing whether the inputted image was the most relevant image from the manual for that text query or not. In projecting both embeddings to the same vector space, we are using MLPs (multi-layer perceptrons) which train independently for the two sets of embedding spaces based on

the cross entropy loss in Equation 1 against the true y_i.

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{1}$$

By this process, we seek to teach the relevant projecting neural networks to find the most important features from the embedding space for determining relevancy between a text and image pair. We wrote all the code for this model and training process ourselves, using the PyTorch deep learning framework. Of note is that this model architecture approach has parallels to the approach explored in the `CLIP` paper. [15] The main differences are the type of model we use to project the embeddings, and the fact that their loss is in terms of the representation of the combined embeddings, with the goal of generating similar representations for similar images/text while ours is in terms of the similarity score between text and image, with the goal of maximizing relevant image retrieval. As a result, we learn a different shared embedding space via a distinct loss and backpropogation. In other words, the general architecture design has similarities, but the distinct tasks that these architectures are intended for results in differing implementations and embedding representation results.

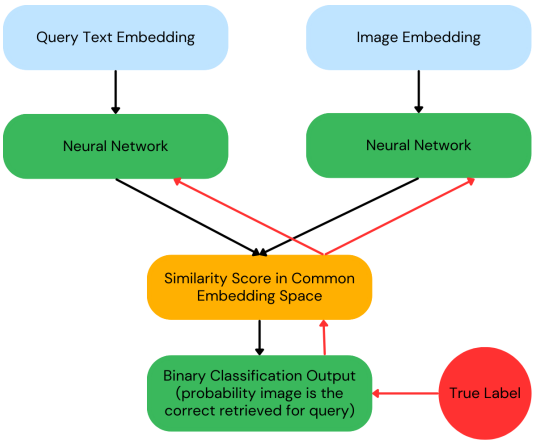

Figure 3: Diagram representing the retrieval model. The query and image embedding vectors are projected into a common embedding space, and a similarity score is calculated between the query and image embedding. Then the model outputs a score between 0 and 1, representing the probability that the image corresponds to the correct instruction manual page to be retrieved for that query. The true label is the actual probability that the image corresponds to the query (0 or 1). The key insight here is that using binary cross-entropy loss we backpropogate (red arrows) through the shared embedding space in order for the model to learn to output correct probabilities for (image, text) pairs.

We experimented with a number of different architecture choices for the MLPs, but the primary architectural features remained consistent, namely ReLU activation functions, Dropout layers for regularization after activation, and Layernorm layers for normalizing the embeddings.

Negative Samples After noticing that the retrieval accuracy was not improving with the baseline training process, we revised the original method of training to include a contrastive approach to negative sampling to teach our model to perform better at the more difficult task of differentiating between the most relevant pages to find the single most relevant page. With the original training scheme, the negative examples were obvious enough that the model was not learning the more subtle features important in determining relevancy. For example, one hand labeled negative sample might be a blank page at the end of a manual. Alternatively if, for a given text query, we took the most relevant page from an instruction manual to be the positive example and all the rest of the pages from the manual to be negative examples then we would have been left with a significant dataset imbalance.

During training, we only fed the model positive examples, which were text queries and the single most relevant page from their corresponding instruction manual, and the model selected the negative examples to be the ones it found most "confusing". Note that each text query has a corresponding set of images representing an instruction manual. At training time, for each minibatch of positive examples we chose the negative example for each of the text queries in that minibatch to be the highest scoring image from the relevant instruction manual which was not the correct (i.e. most relevant) image. One of those images is the positive sample (i.e. the relevant page which we want our model to learn to retrieve). We run all of the images from that instruction manual through the model, along with the text embedding of the query, then we choose as our negative example to be the image which has the highest predicted probability of being retrieved but is not the correct image to retrieve. This helped our model learn to differentiate better between multiple pages which are all relevant but by differing levels. Via the contrastive learning process our results significantly improved even though our training loss struggled more as our model learned to differentiate between the top few most relevant images from the instruction manual, but during the training process the model adaptively chose datapoints that made the loss high.

4. Dataset

Initial Dataset Our dataset is comprised of Ikea instruction manuals, which we process in order to create a dataset such that we can train our multimodal retriever. We use this dataset as previous research has shown model interpretability and understanding of these manuals, such as Wang et. al.

who used computer vision and 3D representations of various Ikea components in order to train agents to build the relevant furniture. [16]

The initial dataset is comprised of 102 Ikea instruction manuals, which detail how to assemble the following furniture types: bench, chair, desk, table, shelf, and miscellenous furniture. [1] From these PDFs we extract the individual pages and store their embeddings within the document encoder database. In total, we extract 996 raw images, all individual pages from the 102 Ikea instruction manuals. Each individual page is represented as a $596 \times 842 \times 3$ image in our dataset.

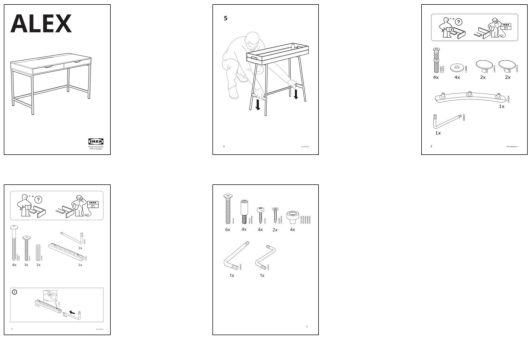

Figure 4: Example of images from the IKEA dataset. Source: IKEA [1]

Feature Extraction We used a variety of different models to generate both text embeddings of the given prompt and image embeddings of the context database. In order to extract the image embeddings as features, we preprocess with the relevant preprocessing necessary for the image embedding model. For example, in order to extract features from the `ResNet-18` model, we resize the images to $224 \times 224 \times 3$ with center cropping and normalization across each channel ($\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$). [9]

Data Processing In training our multimodal RAG model, we created a custom dataset of 112 prompts, i.e. questions about various instruction manuals and pair them with a page which details the relevant information for the given prompt. Using these (prompt, page) pairs, we can train our model and evaluate it on retrieval accuracy, among other metrics. This is especially important for the baseline vanilla model, which is not trained at all but rather only evaluated on its retrieval abilities. For each prompt, we labeled the top 5 images most relevant to the prompt. We assigned the most relevant image a relevancy score of 1 (highly relevant) and the rest a score of 0 (not relevant). Note that we label negative samples *within* a certain instruction manual, as opposed to across the entire dataset, in order to train the model to learn relevant features within a specific manual.

Data Augmentation Furthermore, we augmented our dataset to a total of 6380 (prompt, image) pairs. We used a 70-15-15 split for training, validation, and testing. To achieve this, we employed a variety of traditional NLP augmentation techniques, including back-translation, synonym replacement, and template-based augmentation. With synonym replacement, we substituted specific words in the prompts with their synonyms while maintaining the overall meaning of the sentence. This helped in diversifying the language used in the prompts without altering their intended message.

Additionally, we utilized back-translation, a process where the original text is translated into another language and then translated back to the original language. We implemented this technique using French, Spanish, and Dutch. By translating the prompts to these languages and then back to English, we introduced variations in phrasing and structure while preserving the original semantic content. This method has been shown to improve performance when scaling and augmenting text data on a variety of NLP tasks like machine translation. [6]

Moreover, we applied template-based augmentation, where we created several templates with placeholders that could be filled with different words or phrases. Throughout this augmentation process, we manually reviewed and ensured that the augmented dataset's prompts retained their semantic meaning. This step was crucial to maintain the quality and relevance of the dataset, ensuring that each augmented prompt still accurately described the corresponding image. By combining these NLP augmentation techniques and careful manual validation, we significantly expanded and enriched our dataset, enhancing its robustness and diversity for retrieval.

5. Experiments

In order to create the highest performing RAG pipeline possible, we experimented with several different approaches to our retrieval model. The baseline retrieval model was a frozen layer which computed the cosine similarity between the stored image embeddings and a given text query. This similarity score was not trained, and did not benefit further from more data.

We sought to train our retrieval model to perform optimally. The architecture of this model is explained in depth in the methods section. In order to optimize for the best possible performance of this model, we varied over several different hyperparameter and architecture design choices.

For learning rate, we tried several different orders of magnitude, but found that if that learning rate was below 0.001 that the model got stuck in local minima within the loss space. Thus, we primarily found success with learning rates between 0.01 and 0.05. Furthermore, we used the SOTA Adam optimizer in training our model, as we hypoth-

esized that an optimizer with momentum would be advantageous given the high variability of the data.

The more interesting experiments were with respect to the model architecture and training process. To this end, we tried multiple neural network architectures for projecting the original embeddings into the same embedding space (including number of layers, and size of hidden layers). Ultimately, based on results and the qualitative aspects of the model, we decided on a three layer neural network which used Layernorm layers, Dropout layers, and ReLU nonlinearities. We furthermore used Contrastive Learning when our regular training processes were producing underwhelming results. This change helped boost our performance, as discussed in Section 6.

The outputs of these neural networks were embeddings projected into the same space, and from that point we chose a final layer to calculate the similarity of those embedding projections. We ultimately found greatest success with cosin similarity, but we tried using the L_2 norm as well in calculating the similarities of these projected embeddings. Depending on the process we used for calculating similarity, we then rescaled the score to be between 0 and 1 for our task of binary classification.

Finally, we also experimented with different combinations of embedding models to create the inputs of our retriever. We tried the `Resnet-18` and `CLIP` embedding models for the images, and the `Word2Vec` and `MiniLM-L6-v2` for the query embeddings. Over the course of these experiments, we were able to produce promising results with respect to retrieving the first most relevant instruction manual page. Our ability to retrieve the top k most relevant instruction manual pages benefitted greatly from training, and in practice feeding multiple image embeddings to an LLM for generation is practical, and thus our generator also had improved performance due to the strides we made in retrieval.

6. Results

Experiment 1: Dealing with Negative Samples In order to achieve better retrieval performance, we experimented with hand-labeled negative samples and a contrastive learning approach. For the hand-labeled negative samples, each prompt was assigned a positive (1) and negative (0) image from its instruction manual. For the contrastive learning approach, during training the model is only given the positive sample and it adaptively selects negative samples in the relevant manual for a respective prompt, as explained in the methods section. The contrastive learning approach outperformed hand-labeled negative samples, but the training loss for contrastive learning stagnated. We evaluate whether, for each prompt, if the retriever correctly returns the most relevant image (top-1 accuracy) or if it correctly returns the most relevant image in the top 5 retrieved images (top-5 accuracy). By choosing each "closest but wrong" guess as a negative sample, the model is able to quickly learn and generalize as to the structure of the pages that correspond to types of prompts. We used the same test set for these experiments, and kept the hyperparameters fixed (learning rate, model depth, embedding models, similarity score) except for the type of negative samples. The results of this experiment is summarized in Table 1.

Table 1: Retrieval Results with MLP Retriever with Adversarial Learning vs Human Generated Negative Samples

Training Method	Top-1 Accuracy (%)	Top-5 Accuracy (%)
Adversarial	**0.25**	**0.83**
Human Generated	0.16	0.66

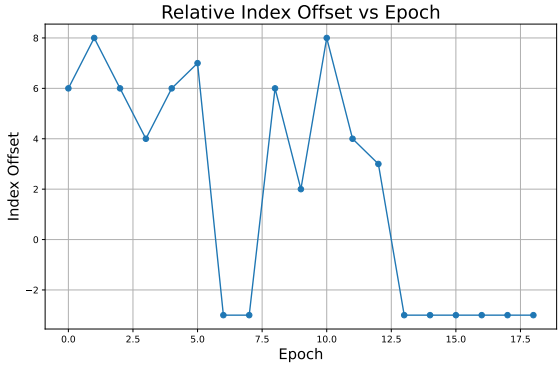

Figure 5: Example of selected negative samples for a single positive datapoint using the contrastive learning approach during training. At each epoch the model identifies a different image as the negative sample. Here each image is represented by its index in the instruction manual, and we show the offset between the selected negative sample image and the true most relevant image. In this example, the query is "Can you tell me about for the Herman chair, what is the last step?" Initially, the model identifies the beginning of the manual as negative samples, but then learns that a "better" negative sample is an image closer to the last step, namely, a few steps before the desired "last step" of the query. This negative sample is much more informative for the model than the hand labeled negative samples, which may be any arbitrary index of the manual.

Experiment 2: Similarity Score In a further effort to improve our retrieval results, we experimented with two different similarity scores: cosine similarity and L2 similarity. Although L2 is a commonly used similarity metric in deep learning, it consistently underperformed when compared to the cosine similarity metric in all of our experiments. We used the same test set for these experiments, and kept the hyperparameters fixed (learning rate, model depth, embed-

Table 2: Retrieval Results with MLP Retriever with Cosine vs L2 Similarity Scores

Similarity Score	Top-1 Accuracy (%)	Top-5 Accuracy (%)
Cosine	**0.25**	**0.83**
L2	0.16	0.75

ding models) except for the similarity score. The experiments are summarized in Table 2.

Experiment 3: Embedding Models We experiment with 2 different types of pretrained text embedding models (`word2vec`, `MiniLM-L6-v2`) and 2 different types of pretrained image embedding models (`CLIP`, `ResNet-18`). As mentioned in the Methods section, `word2vec` is a word level embedding model, whereas `MiniLM-L6-v2` is a sentence level embedding model. The sentence level embedding outperforms the word-level model, which makes sense since we want to capture dependencies across words in our understanding of the prompts. `CLIP` is a multimodal native image embedding model, whereas `ResNet-18` is a deep CNN. The "zero-shot" capabilities of `CLIP` allow for better generalization to our novel dataset. We used the same test set for these experiments, and kept the hyperparameters fixed (learning rate, model depth, similarity score, negative sample type) except for the type of embedding models. Results for this experiment are summarized in Table 3.

Experiment 4: Comparison to Vanilla RAG We compare the retrieval results of our best MLP retriever to a vanilla retriever on our custom test set of 850 (image, prompt) pairs. As a baseline, we use vanilla RAG, i.e. we calculate cosine similarity between the image embedding and the prompt embedding. For the vanilla RAG model, the image embeddings are generated using the `CLIP` embedding model, and the text embeddings are generated using the `MiniLM-L7-v2` sentence embedding model. We then calculate cosine similarity between the text and image embeddings and return the highest scoring image as the retriever output. Using the vanilla retriever, we achieve a top-1 retrieval accuracy of 12.1% on the test set and a top-5 retrieval accuracy of 35%. As seen in Table 4, our best MLP model (using cosine similarity, `CLIP`, `MiniLM-L7-v2`) significantly outperforms vanilla RAG, achieving 25% top-1 accuracy and 83% top-5 accuracy on the test set. The top-5 evaluation metric is highly relevant since in practice our text generator may recall the top k most relevant images from the instruction dataset before generating a reply to the query.

7. Discussion

Retrieval Performance The contrastive learning approach outperformed the hand labeled negative sample performance. We struggled to increase our accuracy even further due to limited compute constraint and the size of the dataset. In the future, we hope to increase the size of the dataset as well as the diversity of the prompts, so that the model retrieval performance can improve. Furthermore, we hope to apply closed source retrieval models like RAG 2.0 or COCA and compare the retrieval results to ours. In order to avoid overfitting with our training set we used Dropout with $p = 0.5$.

Generator Performance We evaluated our entire pipeline with the answer generated by an LLM given the initial prompt plus the context (either the true most relevant image vs the predicted most relevant image). We initially set out to fine-tune a generator on our dataset, but due to limited GPU availability (we needed Ampere GPUs or higher, which were not available on GCS, AWS, or Azure) we were unable to fine-tune the generator. We used Scale AI's LLMEngine Model Zoo with a pretrained `mixtral-8x7b` model with temperature 0.2. [2] As a result, the output of the generator was reasonable at times but unreasonable at other times. For example, given the true context image (correctly predicted from the model) and the prompt

> Given a target question and a set of image embeddings, representing relevant images related to the question, answer the question in as much detail as possible. Target question: "What is the Alex Desk?" Image Embeddings: [1.89, -0.96, ...]

the model output is

> Alex's desk is a desk that is used by Alex. It is made of wood and has a black finish. The desk has two drawers and legs. The desk is located in Alex's bedroom.

As we can see, the first and last sentence are complete hallucinations, although the second and third sentences correctly describe the Alex desk and require explicit knowledge from the retrieved image embeddings.

Limitations of Our Method Our method has a few explicit limitations. 1) The generator model is frozen and not fine-tuned on our dataset, leading to strange and hallucinated outputs by the generator in Section 7. 2) Our method solely focuses on text-image retrieval. In addition to the image context database, one might include text as well, in which case the text embeddings must also be projected into the embedding space. One solution might be to use image captioning to add more information to the images and prevent hallucinations in the text generation. 3) In an attempt to use the most current, state of the art models, we found ourselves limited by package compatibility and GPU constraints. For example, the `FlashAttention 2` library is necessary to finetune current state of the art models like `LLama-3` and `mixtral-8x7b`, but `FlashAttention`

Table 3: Retrieval Results with MLP Retriever on Test Set with Different Embeddings

Image Embedding	**Text Embedding**	**Top-1 Accuracy (%)**	**Top-5 Accuracy (%)**
CLIP	**MiniLM-L7**	**0.25**	**0.83**
CLIP	word2vec	0.17	0.67
ResNet-18	MiniLM-L7	0.17	0.34
ResNet-18	word2vec	0.16	0.58

Table 4: Retrieval Results with MLP Retriever vs. Vanilla Frozen Retriever

Method	Top-1 Accuracy (%)	Top-5 Accuracy (%)
MLP (Ours)	**0.25**	**0.83**
Vanilla	0.121	0.350

`2` only supports Ampere GPUs or newer (i.e. NVIDIA A100s). However, none of these GPUs were available to us on leading cloud provider platforms. Furthermore, `FlashAttention 1` is deprecated on the GPUs we could access (T4), which prevented us from fine-tuning earlier generator models. Additionally, very little support exists for different types of chips like TPUs, which presents a major problem in the field for researchers with monetary constraints or minimal access to computational resources.

8. Conclusion

In this work, we introduce a contrastive learning approach to image retrieval with a novel retrieval architecture, building off of the work of contrastive learning in computer vision. [18, 15] We also introduce a custom dataset of instruction manuals and augmented prompts, to help researchers working on image understanding and retrieval but who are limited by compute. Our MLP-based retrieval model outperforms vanilla RAG significantly on our custom dataset, paving the way for further development on learned retrieval methods using contrastive learning. Many current libraries today, like `LlamaIndex`, use frozen RAG, and we hope to provide researchers with a *learned*, robust methodology for retrieving relevant documents in their database.

9. Contributions and Acknowledgements

Ali Hindy: Ali worked on generating the dataset that we used for retrieval, generating the augmented data using synthetic data techniques, generating image and text embeddings using the ensemble of models we attempted, as well as connecting the retriever model to the generator using Scale AI's LLMEngine. We both contributed equally to the paper.

William Denton: William focused on the retrieval model and took lead of running experiments for different hyperparameter experimentation. He also created the custom contrastive learning scheme, and implemented a training and hyperaparameter framework for different types of embeddings using PyTorch. We both contributed equally to the paper.

References

[1] Download IKEA product assembly instructions - ikea.com. `https://www.ikea.com/ch/en/customer-service/product-support/assembly-guides/`. [Accessed 15-05-2024].

[2] GitHub - scaleapi/llm-engine: Scale LLM Engine public repository — github.com. `https://github.com/scaleapi/llm-engine`. [Accessed 04-06-2024].

[3] D. Caffagni, F. Cocchi, N. Moratelli, S. Sarto, M. Cornia, L. Baraldi, and R. Cucchiara. Wiki-llava: Hierarchical retrieval-augmented generation for multimodal llms, 2024.

[4] J. Chen, H. Lin, X. Han, and L. Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[5] W. Chen, H. Hu, X. Chen, P. Verga, and W. W. Cohen. Murag: Multimodal retrieval-augmented generator for open question answering over images and text. *arXiv preprint arXiv:2210.02928*, 2022.

[6] S. Edunov, M. Ott, M. Auli, and D. Grangier. Understanding back-translation at scale. *arXiv preprint arXiv:1808.09381*, 2018.

[7] A. El-Nouby, N. Neverova, I. Laptev, and H. Jégou. Training vision transformers for image retrieval, 2021.

[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[10] Z. Hu, A. Iscen, C. Sun, Z. Wang, K.-W. Chang, Y. Sun, C. Schmid, D. A. Ross, and A. Fathi. Reveal: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 23369–23379, 2023.

[11] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. Active retrieval augmented generation, 2023.

[12] D. Kiela, A. Singh, and C. A. Team. Introducing RAG 2.0 - Contextual AI — contextual.ai. `https://contextual.ai/introducing-rag2/`. [Accessed 13-05-2024].

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[15] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

[16] R. Wang, Y. Zhang, J. Mao, R. Zhang, C.-Y. Cheng, and J. Wu. Ikea-manual: Seeing shape assembly step by step. In *NeurIPS 2022 Datasets and Benchmarks Track*.

[17] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[18] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu. Coca: Contrastive captioners are image-text foundation models, 2022.