Live, Laugh, Learn: Algorithmic Humor for Generalized Images

Hisham Bhatti
University of Washington
hishamb
hishamb@uw.edu

Samarjit Kaushik
University of Washington
samarjit
samarjit@uw.edu

Parum Misri
University of Washington
parum
parum@uw.edu

Abstract

An understanding of humor requires an understanding of context that can be difficult for all but the most complex modern language models to achieve. However, a distillation of what makes a given image "funny" is a viable task given suitable training data. By utilizing LLMs to generate humorous captions for images and using it to train a much smaller transformer-based model, we built a model to extract the semantics of humorous context from images and generate a suitably amusing caption. Initial tests proved our model can generate coherent text, but suffers from underfitting to common catchphrases or overfitting by directly copying training data. Further experimentation and scaling of the model eliminated this problem, and resulted in significant improvements in semantic structure. Future work can employ new techniques in self-supervised learning to integrate the standard image captioning task with meme generation, and generate new metrics to quantify humor quality. The code is publicly available and the model is free to play with on our project website.

1. Introduction

While humor plays a significant role in human communication, it presents distinct challenges in natural language processing due to its inherently subjective nature. Something humorous to one person might not amuse someone else, or be amusing for different reasons altogether. Humor is deeply intertwined with current social and cultural norms, which are evolving at an increasingly rapid pace due to the widespread adoption of online communities and social media platforms. Making a humorous remark requires not only general understanding of a topic, but also a recognition of how the topic fits within the current cultural landscape, so that the remark may creatively subvert expectations or make an insightful connection (see Figure 1).

Captioning images using LLMs is an avenue that has been thoroughly explored [2]. However, much less work has been done on developing humorous captions, or memes, given highly diverse input images. Our goal is to develop a simple yet highly robust model to humorously caption images given an arbitrary image, as well as construct explicit benchmarks to measure our output. The input image should not need to adhere to existing meme templates, and may instead contain people, landscapes, common objects, etc. This task has far-reaching real-world implications: whether through social media content creation, ad generation, marketing strategies, digital communication, etc., industries and individuals can benefit greatly by using humor to connect with their audience.

Figure 1. Clever meme captioning requires both visual understanding (cat on keyboard) and cultural context (e.g., consonant-heavy Nordic languages). The caption goes beyond the literal image to make a creative connection.

Prior attempts to address this task have presented various shortcomings, with the most obvious being the lack of quality. Due to dataset limitations, such as outdated captions or a lack of generalized images, generated memes often lack the cultural context necessary to create truly clever remarks. This led us to our primary innovation - the creation of a custom dataset that could be easily fed into standard transformer-based architectures to generate appropriately humorous captions. Through this, we constructed a

tool that generates culturally relevant, understandable humor for any image. With the gracious help of Rishit Khare, our model has been deployed online, free for users to play around with. See the code at https://github.com/hishambhatti/meme-generator.

2. Related Work

Automated meme generation is not a novel task - it's essentially an extension of image captioning, which has been an active area of research in deep learning for a decade [2]. Architectures such as transformers, CNNs, RNNs, LSTMs, attention, and scene graphs have historically resulted in varying levels of success, each presenting unique challenges. Since image captioning often requires a substantial amount of data, techniques in self-supervised learning have been developed to help mitigate the need for massive datasets. However, there are shortcomings with preexisting investigations in this field, which we hope to address.

Most obvious is simply the lack of usable data. Consider one of the most popular open-source meme generator models, Deep Humor. Borovik et al. investigated various architectures, but trained all of them on data taken from MemeGenerator.net. The data consisted of 900,000 meme captions for 300 meme template images (3000 captions per template). This approach clashes with the goal of our model, as illustrated in Figure 2. Given that we aim to develop models with sufficient global context to make humorous remarks in response to any input, we would require a large, diverse dataset to broaden the scope of learning. Tolunay and Peirson V's approach [7] suffers from the same problem.

Hwang and Schwartz [4] sought to create a new dataset, "MemeCap," specifically designed for captioning and interpreting memes. Raw images were sourced from the r/memes subreddit on Reddit, and were subsequently annotated with literal and meme captions through crowdsourcing on Mechanical Turk. Unfortunately, this data does not reach a sufficient level of generalization, since memes on Reddit tend to fit specific categories or themes. Other sources such as Met-Meme exist as packages on common ML-frameworks such as Kaggle. However, these datasets consist of the top 100 or 1000 most common meme templates, which, once again, are generic and do not aid to the role of image processing implicit in our captioning model.

Although our motivation differed from similar investigations in this topic, we incorporated similar metrics when calculating the quality of our memes. Hwang and Schwartz [4], as well as Chen et al. [1] utilize BLUE and ROGUE metrics to perform n-gram overlap between human written reference and generated captions, and BERTscore to uncover semantic meaning between the two. Peirson V and Tolunay [7] employ explicit human assessment to see if

Figure 2. Examples of generic meme templates in MemeGenerator.net, which is the training data for most current models. These fixed-template images limit visual diversity and do not support the open-ended, context-rich image captioning that our model aims to achieve.

generated memes can be differentiated with human created ones. We incorporate semantic loss by calculating the cosine similarity between embeddings of the trained and outputted text - captions with similar meanings will look similar in feature space.

3. Methods

3.1. Dataset Creation

Humor is based on context, and to provide an adequate breadth of context for a successful captioning strategy to arise, we knew that we would need a colossal amount of raw data. Scraping images with existing humorous captions from the internet proved to be infeasible - there was no way to efficiently verify that the captions would be related to the image in a way that would be possible to capture semantically at the small scale our model promised. Additionally, sites like Reddit and Instagram have limited API access in efforts to control data harvesting. As mentioned earlier, existing meme datasets proved to have their own issues, either suffering from a lack in image variety (impeding our efforts to make a generalizable model), or simply being too small in scale to effectively train on. Therefore, we decided to establish a pipeline wherein we collected images from raw image datasets and captioned them via an existing LLM.

It's important to note that the image collection process required a degree of discernment to ensure that our training data spanned a broad variety of categories and visual elements, since we wanted the model to be generalizable to any set of image features. The Google OpenImages and Microsoft COCO 2017 datasets contain millions of photos evenly distributed across a wide variety of categories and settings. We used 3000 photos from OpenImages to train our base model, then used COCO to train subsequent models with 30,000 and 60,000 training photos.

For the captions, we wrote a series of prompts for Google Gemini's Flash-1.5 model designed to produce captions that

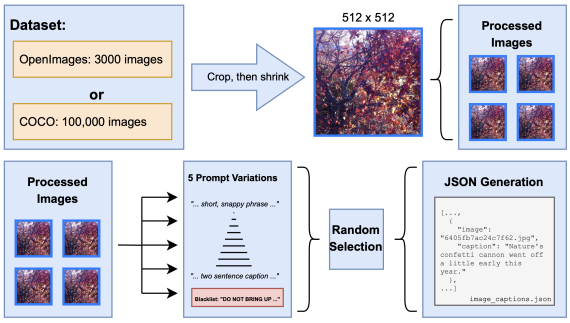


Figure 3. A diagram of our dataset creation pipeline

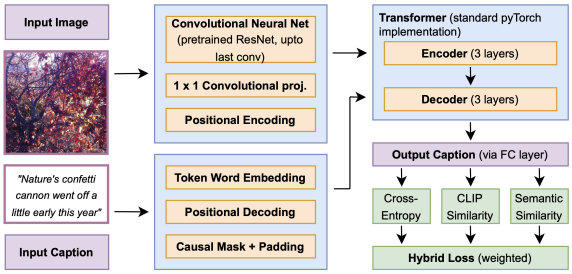


Figure 4. A diagram of our model architecture

captured something humorous about a given image utilizing modern writing styles. In an effort to mitigate our model's tendency to repeat the same sentence structure across inputs, for each image we captioned, we randomly selected one of several prompts, which instructed the LLM to either output a short phrase, a 1-sentence caption, or a 2-sentence caption.

Once images were compiled and captions generated, we converted each image to a matrix of 512 by 512 values across three channels (RGB), collating them with captions to create a dataloader (see Figure 3. We also randomly split these image-caption pairs into train (80%), test (10%), and validation (10%) data.

3.2. Model Design

Initial Design. In the first iteration of our data pipeline, we built an end-to-end deep learning model that did not use transformers. This simple model used the feature-space output of a pretrained convolutional neural network to encode images, and then used an LSTM-based decoder to generate captions based on this semantic information. This model was used as a comparison point, only trained for 20 epochs and on a small library of 70 images from the Google OpenImages dataset. After establishing a working pipeline, we scaled up to 30 epochs with 3000 images from the same dataset. Training captions were generated using a simple prompt for Gemini, requesting 1-2 sentences that captured something funny about a given image without describing it directly.

Transformer Architecture. Next, we designed a transformer-based model, still using a CNN-based ResNet architecture pretrained on ImageNet [3] (up to the last convolutional layer) before creating a convolutional projection to the input space of our transformer, and adding positional encoding. Captions were tokenized in reference to a vocabulary built via a simple word-based tokenizer, and both causal and padding masks were applied. Finally, the image features were fed into a 3-layer transformer encoder, and the output (along with caption embeddings) was fed into a 3-layer transformer decoder, which uses self-attention to produce output embeddings (which follows a single fully-connected layer). This output takes the form of a probability distribution over the vocabulary, which the decoder process greedily uses to generate an output sequence (feeding in previous tokens through the transformer and picking the most probable next token). The transformer architecture was based on the pytorch implementation [8], with a 0.1 dropout rate (see Figure 4).

Loss Computation. We originally used simple Cross-Entropy loss, comparing our decoded outputs to the ground-truth (training data) outputs. However, since the goal of the model is to learn common formats and adapt to image information (not just reproduce captions word-for-word), we defined a hybrid loss function to combine three complementary objectives:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{CE}} \cdot \mathcal{L}_{\text{CE}} + \lambda_{\text{CLIP}} \cdot \mathcal{L}_{\text{CLIP}} + \lambda_{\text{SEM}} \cdot \mathcal{L}_{\text{SEM}} \quad (1)$$

$\mathcal{L}_{\text{CLIP}}$ is used to encourage visual-textual alignment via CLIP embeddings [5]. Given an image I and generated caption $\hat{y}$, this loss component is defined as:

$$\mathcal{L}_{\text{CLIP}} = 1 - \cos\left(f_{\text{image}}(I), f_{\text{text}}(\hat{y})\right) \quad (2)$$

where f_{image} and f_{text} are the image and text encoders from a pretrained CLIP model. Next, to go a step beyond cross-entropy loss and compare the semantic similarity of our text to the target, we embed the predicted caption and reference caption using the model's token embedding layer and compute their cosine similarity:

$$\mathcal{L}_{\text{SEM}} = 1 - \cos\left(\text{Embed}(\hat{y}), \text{Embed}(y)\right) \quad (3)$$

Finally, to construct our total loss, we apply weightings λ for each loss component. λ_{CLIP} is kept constant, while λ_{CE} and λ_{SEM} are modulated over the course of training via a cosine warmup:

$$\lambda_{\text{SEM}}(t) = \lambda_{\text{SEM}} \cdot \text{cos\_warmup}(t, 0, 1), \quad (4)$$

$$\lambda_{\text{CE}}(t) = \lambda_{\text{CE}} \cdot (1 - \text{cos\_warmup}(t, 0, 0.5)) \quad (5)$$

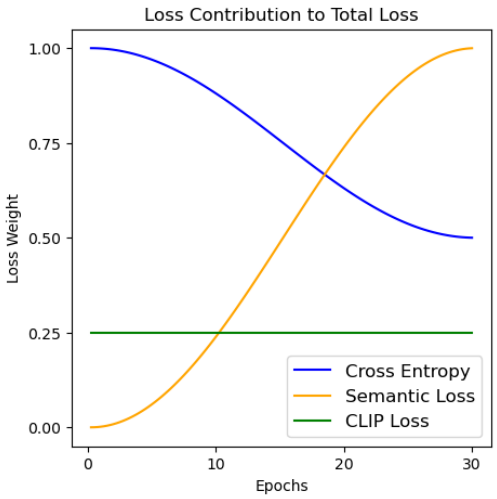


Figure 5. Change in contribution (weight) of loss components in calculating total loss, over the course of our training regime.

Where the warmup function goes from a start to end value over the course of training for N total epochs:

$$\begin{aligned}&\text{cos\_warmup}(t, \text{start}, \text{end})\\&= \text{start} + \frac{1}{2}(\text{end} - \text{start})(1 - \cos(\pi t/N))\end{aligned}$$

This lets the model focus more on token accuracy (via cross-entropy loss) early on, and gradually shift towards higher-level semantic meaning, as depicted in Figure 5.

3.3. Evaluation

We used validation testing to monitor overfitting/underfitting in our model, calculating similar loss components for the validation set after each epoch. Additionally, after building each version of the model, we generated captions for twenty random images from the test set to assess quality by hand: based on whether the captions made grammatical sense, had humorous content, and if they were at all relevant to the image.

For our final models, we used more rigorous evaluation, specifically checking CLIP similarity (image relevance) and measuring fluency. Specifically, we used perplexity as a measure of fluency via a pretrained GPT-2 model in HuggingFace's Transformers library [6], computing the exponential average of the negative next-word log-likelihood of the model generating the specified caption. Lower average perplexity scores mean that the caption was more likely to be generated by a natural language model, and therefore is likely more fluent and grammatically correct.

For each model being evaluated, we computed and compared perplexity for the ground-truth and generated captions, for equal subsets of the training and testing image datasets. This allowed us to use ground-truth scores as a sanity check, and to compare relative fluency across models.

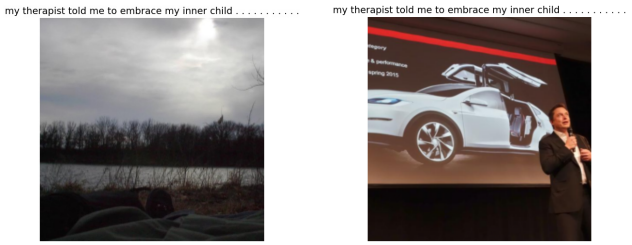
4. Experiments

4.1. Initial Results

Our initial model demonstrated promising syntactic coherence, successfully preserving sentence structure. However, it was not generalizable, and certainly not original. It transitioned rapidly from reproducing overrepresented captions in the dataset to overfitting to the dataset as the number of epochs increased. When training for only 20 epochs, we noticed that the generated captions tended to follow the same format, regardless of objects in the image. For example, consider three distinct images: a presentation of car, a lake on an overcast day, and a street sign on a highway. All three captions produced by the model had the same text: "my therapist told me to embrace my inner child..." (see Figure 6a). Other common captions included "this is my life choices," "more personality than my dating profile," "existential dread/existential crisis," etc. We hypothesized that the reason for this was due to the fact that these phrases were overused by the LLM generating our training data. Since our model was under-trained, it was not yet able to understand the objects and context of the image, and instead defaulted to the most common catchphrases. To address this, we updated our prompt for generating training captions to reduce the frequency of such phrases, and noted a substantial decrease in the occurrence of such phrases in outputted captions (although not a complete elimination of those phrases, perhaps due to limitations in Gemini's context window).

After training for more epochs, we soon ran into problems with overfitting, exacerbated by the small size of the initial dataset. When given an image of a giraffe, our model outputs "He's got that "I've seen things" look." While amusing, this caption is directly stolen from an image of a bug in the training dataset (see Figure 7). Rather than synthesizing context and generating original captions, our model extracted some key features, and then directly copied text from the training data.

This raised concerns regarding whether we built a captioning model at all, or merely an image classifier with extraneous flavor text. As shown in Figure 6b, when feeding images of a cat, car, and gravestone, the model invariably outputs "this X got more personality than my dating profile". From this, we believe that our initial model seemed to classify a key object in the image (if it is known), and subsequently incorporate it into one of the common joke templates. Another challenge involved making the caption comprehensible and grammatically correct. Oftentimes the generated caption would end mid-sentence, or reuse the same keyword excessively. Furthermore, we lacked a for-



(a) With fewer epochs, our model defaults to overrepresented captions in the training set.

(b) With many epochs, our model appears to perform image classification, then plugs the subject into common meme templates.

Figure 6. Initial results for our model trained on 3,000 images

(a) Test image of a giraffe with the generated caption: "He's got that 'I've seen things' look."

(b) Training image of a bug with the identical caption. The model memorized and reused this text.

Figure 7. Training and test image illustrating overfitting.

mal metric to balance underfitting and overfitting. To establish a threshold for early stopping, we incorporated semantic loss on the validation set.

4.2. Enhancing Training Data

Our initial 3,000 image model's loss plateaued after about 20 epochs (see Figure 8a), but we believed it was highly likely that as we introduced more training data, the model would be able to produce more generalizable captions. Given what we learned from the common-phrase issue (Fig. 6a), we needed to ensure the inputted data had a reasonable variety of caption formats. Ensuring variety of outputs when using an LLM is complicated, since given similar two inputs and no knowledge of the other request, LLMs often return similar responses. To mitigate this, we experimented with a variety of prompts to generate different formats, and eventually settled on a randomly rotating selection of 5 different formats for LLM caption outputs: 2-5 words, a short phrase, a single sentence, 2 sentences, and open interpretation, as well as introducing a blacklist for overrepresented phrases ("my therapist", "existential dread", "more personality than X", etc.).

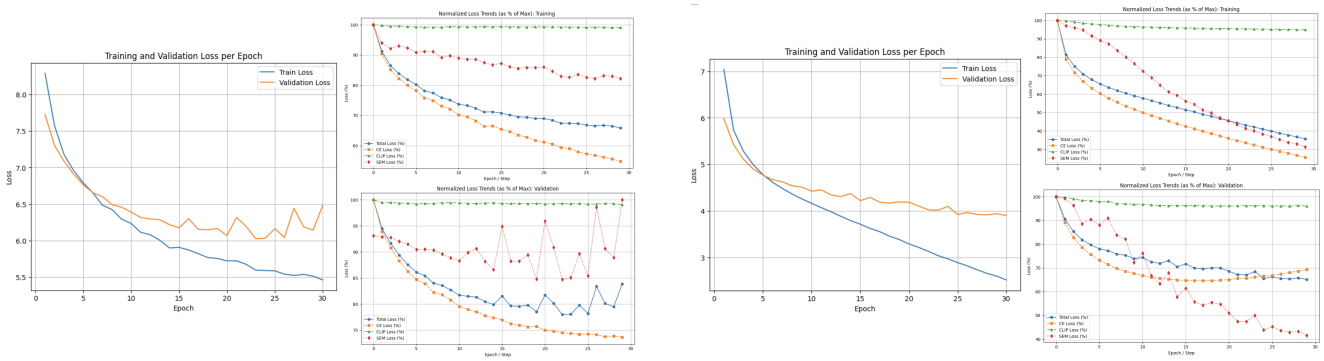
Due to licensing and API issues, we were unable to access a larger subset of Google's OpenImages dataset beyond the 3000 images used to train our initial model. As such, we pivoted to the COCO 2017 photo dataset, which contained 118,000 real-world photos across a similar category distribution to OpenImages. Utilizing our new prompt, we were able to generate captions for all 118,000 images, and subsequently use this dataset to train 30,000 and 60,000 image models.

This proved to be very effective - contrasting the validation loss between the 3,000 image model and the 60,000 image model, it's clear that there are significant improvements across nearly all loss categories (Figure 8b). Further increasing the training data would likely have resulted in further improvements, but due to compute limitations, we were unable to train larger models.

Looking at the loss components revealed interesting trends: while semantic loss fluctuates wildly in the 3k model, it plays a much larger role in the training process for the 60k model, decreasing significantly in the second half of training. The training loss follows an expected pattern (given our adaptive loss weights scheme), but surprisingly, semantic loss decreases much more rapidly in the validation set while cross-entropy loss plateaus and actually starts *increasing* in the second half of training. This could mean that with the larger dataset, the model has more time to learn basic vocabulary and grammatical rules (via cross-entropy loss) and can reasonably preserve similarity to the target caption while starting to simultaneously improve semantic similarity (matching the high-level meaning). Additionally, despite not playing a role in back-propagation (functioning more as a regularizer), CLIP similarity loss also steadily decreases over the course of training, and by a larger proportion in the model with 60k images.

4.3. Byte-Pair Encoding

Our initial model used word-level embeddings over the entire training set vocabulary. As discussed in class, such a scheme has several downsides. Word embeddings require a large, fixed vocabulary, and is not resilient to misspelled, rare, or newly invented words. A common practice to mitigate this is to use byte-pair encoding, which starts with character-level tokenization and one-by-one merges the most frequent tokens together. In the end, byte-encoding



(a) 3K Model

(b) 60K Model

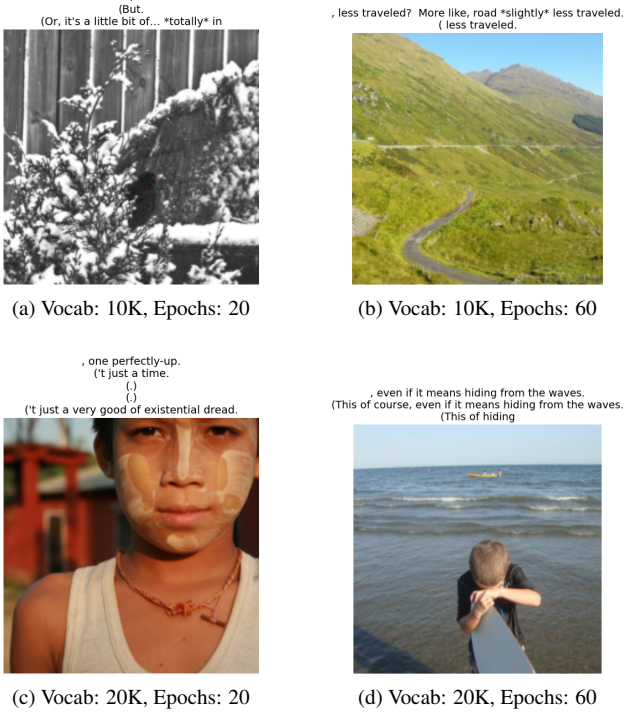
Figure 8. Training and Validation Loss for 3K/60K Models

provides the high coverage of character-level representations with the computational efficiency of word-level representations.

To test this, we replaced word-level tokens to byte-pair encoding. The results are displayed in Figure 9. For all vocabulary sizes and epochs trained, our model's syntax and grammar was less comprehensible than the baseline. In particular, image captions using BPE was blocky and often grammatically incorrect, consisting of strange punctuation, unmatched parenthesis, excessive colons, etc. With more epochs, the model picked up on semantic information in the image, as evidences by the "road" and "waves" in the right images. However, none of these captions rivaled that of the baseline. In fact, increasing the vocab size, and other hyperparameters related to the embedding had no empirical effect on the quality of captions produced.

There are several hypotheses as to why byte-pair encoding was not effective for our dataset. Foremost is the relative lack of training data and computation - larger datasets provide greater opportunities for models to learn token relations and grammar rules. Since byte-pair encoding treats every punctuation element as a token, it may be harder to learn high level sentence structure and semantic relationships early in training. This results in cases with unclosed parenthesis, since the model has not had enough training time to conceptualize this rule. With little understanding, the model tends to weight uncommon byte tokens (colons, parenthesis, etc.) disproportionately in its captions.

The structure of the training captions themselves likely plays a role in BPE's poor performance. Standard text trained for image captioning models have simple, consistent grammar: for example, "A cat is playing with yarn in the backyard." However, meme captions are not homogeneous. Part of humor lies in verbal wordplay, resulting in training text with wildly varying syntax, sentence structure, and punctuation. This exacerbates the issues related to limited training data, and results in inconsistent and illogical tokenization when using byte-pair encoding.



(a) Vocab: 10K, Epochs: 20

(b) Vocab: 10K, Epochs: 60

(c) Vocab: 20K, Epochs: 20

(d) Vocab: 20K, Epochs: 60

Figure 9. Examples of model outputs using byte-pair encoding (BPE), illustrating the effect of vocabulary size and number of training epochs.

4.4. Semantic Score

Our perplexity evaluation on our smallest and largest models demonstrated that although there is a slight fluency improvement in the larger model (for both the train and test scores, see Figure 10), the scores are significantly worse than for the ground truth captions. Our training regime wasn't built to optimize fluency, and achieving scores on par with state-of-the-art language models given significantly lower training times would be an unrealistic goal. It's pos-

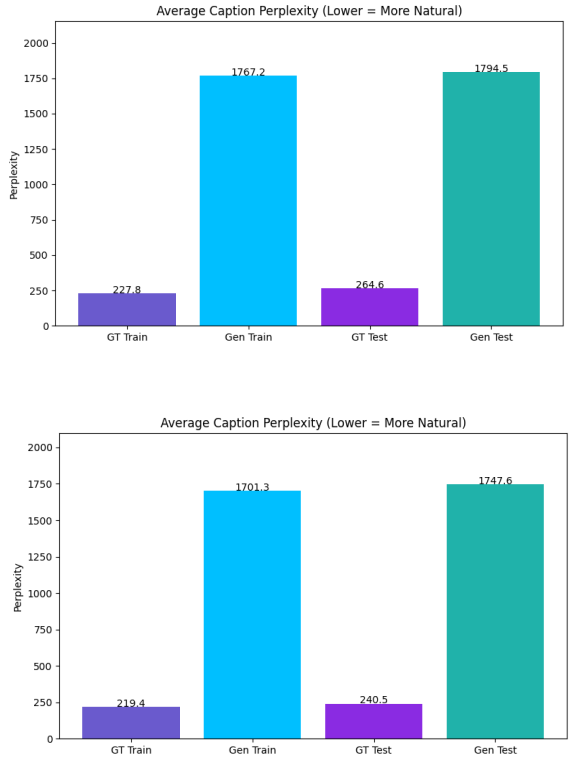


Figure 10. Average perplexity scores for models trained on 3,000 images (top) and 60,000 images (bottom). Lower scores indicate fluency, and GT indicates ground-truth captions (targets).

sible that the scoring system penalizes some of our model's captioning quirks (like the use of excessive spaces around punctuation), and raises the question of whether meme-style captions need to have the same level of fluency as most natural language in order to be humorous.

Regardless, manual verification demonstrated that the majority of captions made grammatical sense (with most errors trending towards the end of longer captions). The CLIP similarity scoring on both our validation (Figure 8b) and testing image sets showed that the 60,000 image model slowly improved in image relevancy over time.

5. Discussion

We found that our initial baseline transformer model produced coherent sentences, but did not produce original, generalizable insight. Rather, our model either underfitted to the most common catchphrases, or overfitted by detecting features in the image, and directly copied phrases from the training set. Increasing the dataset size proved to be the most significant boost to our performance. With more input, the model drastically improved both in semantics and grammatical syntax, as evidenced by declining semantic loss and a slightly smaller perplexity score. Byte-pair encoding proved to be ineffective given the size of the dataset, as well as atypical syntax unique to meme captions.

Our findings confirmed our initial expectations about the challenge of generating humor. Given the wide context necessary to generate creative insights, learning humor indeed requires a diverse, massive, and constantly-updating dataset. Additionally, humorous captions are not as straightforward as standard image captioning. Attributes unique to humor, such as inconsistent phrasing, subtle visual cues, and captions that don't directly reflect the context of an image, made the model especially challenging to train.

As this paper is exploratory in nature, further research can be done to expand our model's capabilities. Once again, it is clear that our model benefited from a larger dataset. Due to time and computing constraints, we could only train a 60,000 image model, but based on the dramatic improvement over the baseline, we expect an even larger dataset to perform even better. One technique that we did not explore, but could be valuable given limited data, is self-supervised learning. This involves pre-training the model on a large dataset of images and "normal" image captions, extracting the feature encoding, then post-training on the specific task of meme captioning (with a relatively smaller dataset of "meme" image captions). Such an architecture relies on the fact that creative meme captions rely on having a solid grasp of the content in an image.

Another avenue for improvement involves finding new metrics to interpret our model and benchmark performance. We did not employ any explicit techniques in mechanistic interpretability to gauge why or how our model is generating what it does. Such insight could reveal biases or spark additional development. Although we have metrics to quantify semantics, it might be insufficient for humor. Two different captions for the same meme can be wildly different in terms of meaning, but be equally "funny." In some cases, the training meme caption might not match the content of an image at all. Research into developing a metric for "humor" is not well-established, but might prove valuable going forward.

Lastly, though our prompt has undergone multiple iterations, it still under-performs at times. Despite explicit instructions to avoid common phrases, a non-negligible fraction of Gemini-generated images still use "therapist," "existential dread," or other generic, outdated templates. Many captions were convoluted or hard to interpret, and if we want a better model, improving the quality of our training dataset and caption generation methodology is essential.

References

[1] Yuyan Chen, Songzhou Yan, Zhihong Zhu, Zhixu Li, and Yanghua Xiao. Xmecap: Meme caption generation with sub-image adaptability, 2024. 2

[2] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. Deep learning approaches on image captioning: A review. *ACM Computing Surveys*, 56(3):1–39, Oct. 2023. 1, 2

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 3

[4] EunJeong Hwang and Vered Shwartz. Memecap: A dataset for captioning and interpreting memes, 2023. 2

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 3

[6] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 4

[7] Abel L Peirson V and E Meltem Tolunay. Dank learning: Generating memes using deep neural networks, 2018. 2

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 3