CS230

DeepMeme: Generating Memes with Deep Neural Networks

A.L.Peirson V
Department of Physics
alpv95@stanford.edu

E.Meltem Tolunay
Department of Electrical Engineering
meltem.tolunay@stanford.edu

Abstract

We introduce a novel meme generation system, which given any image can produce a humorous and potentially relevant caption. Furthermore, the system can be conditioned on not only an image but also a topic sentence, giving a handle to the user on meme content. The system uses a pretrained Inception network [10] to return an image embedding which is passed to an LSTM producing the caption - inspired by image captioning methods [11]. We implement a modified beam search to encourage diversity in the captions. We evaluate the quality of our model using perplexity and human assessment on both the quality of memes generated and whether they can be differentiated from real ones. Our model produces original memes that cannot on the whole be differentiated from real ones. https://github.com/alpv95/MemeProject .

1 Introduction

'A meme is an idea, behavior, or style that spreads from person to person within a culture often with the aim of conveying a particular phenomenon, theme, or meaning represented by the meme. A meme acts as a unit for carrying cultural ideas, symbols, or practices, that can be transmitted from one mind to another through writing, speech, gestures, rituals, or other imitable phenomena with a mimicked theme.' [3].

Memes are ubiquitous in today's day and age; their language and ideologies are in constant flux. Memes come in almost every form of media, with new formats constantly evolving. Primarily they function as a medium for humor to be shared, utilizing cultural (especially subcultural) themes. However, they can also be manipulated to further political ideals, magnify echo chambers and antagonize minorities. It can take a remarkably high level of understanding to produce a good meme (image, culture, language etc.). As AI grows in leaps and bounds it requires new and challenging tasks. The contemporary relevance of memes and the high level of understanding required to generate them motivate this project.

We approach this task by considering only the image with caption class of meme, an example of which is shown in Fig.4. This reduces the problem greatly and allows for relatively simple collection of datasets, §2. By 'generating a meme' we mean given an image (which can be a meme template or otherwise) our model will caption the image humorously and in a manner that is relevant to the image itself. Inspired by recent progress in image captioning, §2, we apply an image captioning system similar to the one outlined in [11], consisting of a pretrained CNN image embedding initial stage, followed by an LSTM RNN. This takes any image as input and outputs a caption for that image. We also explore an image with label model, which takes both an image and a label (the name of the image for instance) and outputs a caption. We make use of pretrained GloVe word vectors unlike [11].

Evaluation of generated meme quality is difficult to reliably automate. We proceed by taking a perplexity score on an evaluation set of memes with repeated format, §3 and depend on 2 separate human tests for final model evaluation, §4.

2 Related work

The advent of sequence-to-sequence machine translation models [9] established the idea of encoding information (such as a French sentence) and using at is an input to an RNN that generates language.

It was not long before this encoder-decoder framework was extended to image captioning [11], where the encoder for the model is a CNN taking the image as input. These were further improved by using bi-directional LSTMs [12] and including attention mechanisms [4]. Although these models perform very well on metrics such as BLEU for factual descriptions of images, there has been little work on generating humorous captions. Models such as StyleNet [5] have attempted to produce humorous caption using an encoder-decoder architecture with limited success (the jokes are very one-level). Successful meme generation requires a diverse range of humorous captions for the same image which are related to concepts portrayed by the image, not necessarily the content of the image itself. To achieve this we make use of much of the previous work above while incorporating our own ideas [7].

3 Dataset and Features

Our dataset consists of approximately 400,000 image, label and caption triplets with 2600 unique image-label pairs, acquired from [2] using a python script that we wrote. Labels are short descriptions referring to the image, i.e. the meme template, and are the same for identical images. Accordingly, each image-label pair is associated with several (roughly 160) different captions. A sample from our dataset is shown in table 1.

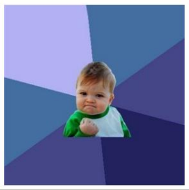
Image	Label	Caption
	• success kid	• Didnt study for a test still get a higher grade than someone who did • Ate spaghetti with a white shirt on no stains…

Table 1: Sample dataset

Before training our model with the captions, we performed a preprocessing on our dataset. Each word in the caption was lowercased to match the GloVe format, punctuation marks were assigned their own tokens, and start, end and unknown tokens were introduced into the vocabulary with randomly initialized word vectors. We performed a cut on the words that appear in the captions: every word that appears fewer than 3 times in the corpus is set to UNK. Captions with more than 2 UNKs are removed from the dataset. Small image distortion were also applied during training, explained in §4.

4 Methods

4.1 Encoder

The motivation behind the encoder is to provide a meaningful initial state to the decoder to initiate the text generation process. We implemented two different encoder architectures: an alexnet CNN [6], shown in Fig.2, and an Inception v3 CNN, Fig.3, as in the original 'Show and tell' model [11]. The alexnet is pretrained on imageNet [1] and the Inception on the ILSVRC-2012-CLS (subset of [1]) image classification dataset. The alexnet encoder takes the output from the first fully connected (dense) layer in the architecture (shown in Fig.3). It has 8 hidden layers in total. Similarly, the Inception encoder takes the last hidden layer of the CNN as the encoder output. Note that these encoders output a 4096 or 2048 dimensional vector respectively which results in a mismatch with our word embedding space that is 300 dimensional and is required as input to the decoder. Hence, we project the image embeddings into the word embedding space using a trainable fully connected layer. In this project, we implement 2 different variants of the proposed encoder scheme. The first variant explained above just trains with the meme templates and disregards the labels completely, this is show in fig.1. Hence, the inputs to the decoder solely include encodings from the images. Only an image is required to generate a meme.
The second variant of the encoder includes the meme labels. In this model, we first obtain the image embeddings by running the images though Inception as previously. Now, we also get the pretrained GloVe embedding for each word present in the meme label and compute their average. This averaged vector is concatenated to the image embedding vector and then fed into a trainable fully connected layer. We average rather than concatenate as this keeps size constant. This variant was motivated by the assumption that the label words contain semantic encodings that can be mapped into word

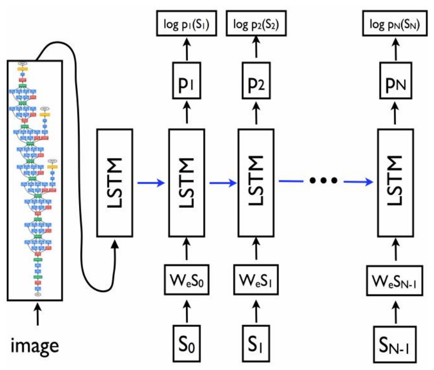


Figure 1: Our base model architecture [11]. The image encoder on the left features an Inception v3 network. $\{s_0, ..., s_{N-1}\}$ is the caption example from the training set or a caption being generated. $\{W_e s_0, ..., W_e s_{N-1}\}$ are the word embeddings of the sentence. $\{p_1, ..., p_N\}$ are the softmax probabilities for choosing the corresponding word s.

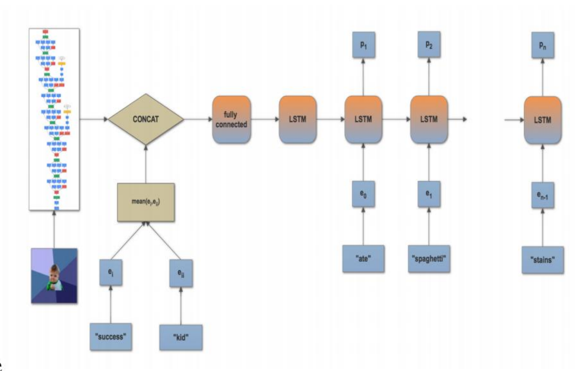


Figure 2: A visualization of our second encoder-decoder model variant. We refer to GloVe word embeddings as e_i and the probability distribution obtained at each LSTM time-step as p_j.

embedding space to deliver contextual meaning for the language generation that occurs in the decoder. The output of the fully connected layer is fed into the decoder as the initial state in the same manner as the first encoder scheme. Fig.2 shows this architecture.
We experimented with finetuning the last few layers of the encoder only for the Inception case, because it became clear early on that this worked much better than the alexnet §5. Since the number of unique images in our data is very small, we decided to finetune very only very few layers and to train without finetuning first. We also experimented with applying basic distortions to the images (horizontal flips and slight colour distortions) randomly to augment our image dataset and allow the model to learn more robust representations of memes. In practice this reduced the perplexity scores of our model slightly, but significantly increased qualitative performance.

4.2 Word Embeddings & Decoder

We found that initializing the word vectors randomly led very quickly to overfitting (the model achieved reasonable perplexity scores but only ever generated very common, unvaried captions). In light of this we implemented GloVe word embeddings [8], pretrained on an internet trawl. We chose to use 300d word vectors as these are shown in the GloVe paper to provide the best word representations [8]. We tried both trainable and non-trainable word embeddings, finding that trainable ones performed much better qualitatively and quantitatively. This makes sense as the vocabulary of our corpus consists of over 40,000 individual words and memes use language in a very idiosyncratic manner, perhaps not well captured by the GloVe algorithm on its internet trawl.
Once the batch of images (and labels in variant 2) has passed through the encoder, the output is passed as input to the first hidden state of the decoder. The decoder consists of a 1 or multilayer unidirectional LSTM.

4.3 Beam Search & Evaluation

For original and humorous meme generation, greedy search is completely ineffective. Furthermore, we found that standard beam search which takes the top k most probable next words gives adequate but non-optimal results as can be seen in §5. In order to generate the freshest memes we implement a temperature function into the beam search algorithm. Instead of selecting the top k most probable words, the k words are selected from a probability distribution of the top 100 words, where the temperature of the distribution is a hyperparameter. A probability distribution p can be modified with temperature T by the function f shown below.

$$f(p)_i = \frac{p_i^{1/T}}{\Sigma_j p_j^{1/T}} \tag{1}$$

where $T = 1$ corresponds to unchanged probabilities, high T lead to a very flat distribution (random pick) and low T leads to argmax (greedy search).
To quantitatively evaluate our model, we created an evaluation set of 105 memes + captions taken from the training set which have repeated formats. E.g in Fig.1 the Boromir meme almost always starts the caption with 'one does not simply', so the boromir image + 'one does not simply' would constitute one example in the eval set. For the case where we additionally condition on the name of the meme, this would also be provided in the eval set for each example. We calculate the perplexity scores of the model on this eval set. Perplexity (PP) is a measure of the inverse probabilities of predicting the next word in the example caption (C):

$$PP(C) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1...w_{i-1})}}. \tag{2}$$

$\{w_1...w_N\}$ are the words in caption C. The probabilities P can be found from the cross entropy loss calculated on the LSTM output. Low perplexity means the model frequently assigns high probabilities to the given eval example captions. This metric tells us how well the model is learning to caption images of different formats with the correct style. It is a limited metric for success as it tells us nothing about whether the captions are humorous, original and varied. To solve this problem, we additionally implement a function to check whether generated captions, or exceptionally similar captions, are in the training set. For our final test we show 20 different memes to 5 people from diverse backgrounds and note if they can differentiate them from real (training set) memes of the same format and random text generated memes. The same people also ranked the memes being shown on how funny they found them on a scale of 0-10.

5 Experiments

5.1 Training

To maximise the performance of all of our competing models we take the following steps for each. Using the perplexity score of the model, explained in §3, as our minimisation target we conduct a thorough hyperparameter search for the learning rate and its decay function for both SGD and momentum optimization techniques. Minimising perplexity ensures the model can generalise to wide variety of meme formats and eventually overfit the data. We qualitatively observe for all the models that overfitting begins to clearly occur after roughly 55 epochs, so we use employ early stopping before this point. As in [11] we further employ dropout with 0.7 keep-probability on the LSTM decoder to delay the overfitting time. Secondly we test a single, double and triple stacked LSTM decoder layer using both perplexity and qualitative metrics. Finally, once the optimal hyperparameters are found, we finetune the encoder parameters for a final performance boost. Our decoder parameters are all initialised with a random uniform distribution between -0.08 and 0.08 as in [11]. The encoder and embedding parameters are initialised pretrained, explained in §3.

5.2 Results & Discussion

Due to the wide variety of meme formats, we found that different beam search parameters delivered better results for different styles of meme. Overall we settled on a beam search with a length normalisation of 0.2, a temperature of 1.3 and k=5 as a formidable one size fits all. This beam search was also observed to generalise well to unseen images. From fig.4 we can see that the Inception encoder performs much better on the perplexity metric than the alexnet (fig.4 also shows final hyperparameter choices). This performance discrepancy is also observed in the quality of generated memes from the two encoder models. Therefore from now on our models incorporate only the Inception encoder. We observed no noticeable quantitative or qualitative gain in performance from using 2 or 3 LSTM layers, so we stick to 1.
Fig.5 summarises the results of our models, applied to both images seen in the training set and unseen images. It is clear that both models performed similarly and well, both being almost indistinguishable from real memes and equally as funny. This performance extends to unseen images which are close to the same quality, although with slightly more copied captions from the dataset. This suggests the models were both able to generalise well. We found that the label argument passed for unseen images did not provide a handle on the content of the generated caption (as in fig.3), but did increase the variety in generated memes (as can be seen from the lower % in data score for the Image+Name model for unseen). This is perhaps to be expected as there were very few labels in the training set. Finetuning of the Inception encoder turned out to train too slowly to provide meaningful results.

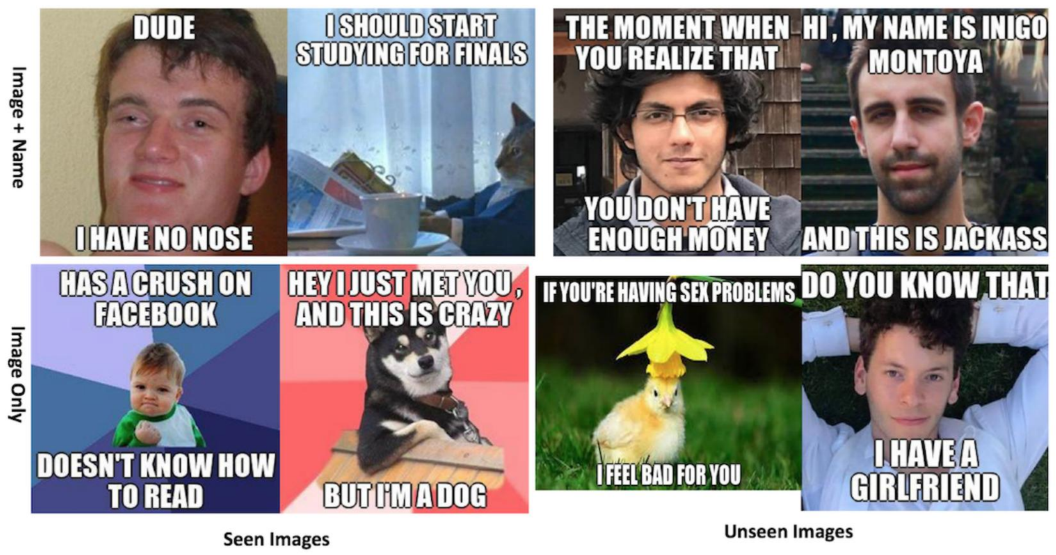


Figure 3: Original memes generated by both model variants. The input labels for the Image + Name model seen images are their labels from the training set while for the unseen image we use 'AI is the new electricity' for both. Upper-lower text breaks are added manually.

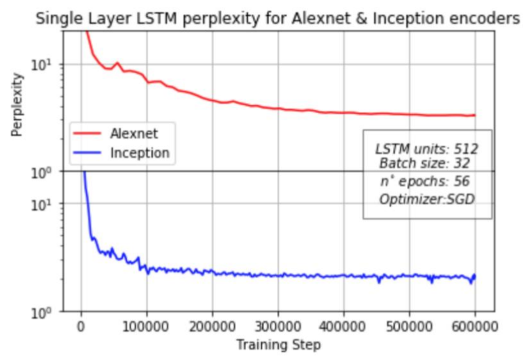


Figure 4: Perplexity scores on the evaluation set for the Alexnet and Inception encoders for the first model variant (image only).

Model	% in data	Perplexity	Hilarity	Differentiability
Seen Images				
Image	17	2.01	7.5	57%
Image+Name	18	2.28	6.8	63%
Unseen Images				
Image	29	-	6.1	-
Image+Name	26		6.9	-

Figure 5: Real Memes scored on average 7.3 in the hilarity test, between 1-10. Differentiability shows how often our human testers were able to distinguish real memes from generated ones, if indistinguishable this score would be near 50%. Since memes are generated by beam search, one has to manually select a caption from the k generated, introducing a slight selection bias.

6 Conclusion

In this paper, we have successfully demonstrated how to use a neural network model to generate memes given an input image. We suggested different variants of encoder schemes that can operate with or without labels, and reported a fine-tuned LSTM network for language modeling. Obtained results indicate that memes can be generated that in general cannot be easily distinguished from naturally produced ones, if at all, using human evaluations.
Future work for improving this system would include training on a dataset that includes the break point in the text between upper and lower for the image. These were chosen manually here and are important for the humor impact of the meme. If the model could learn the breakpoints this would be a huge improvement and could fully automate the meme generation. Another avenue for future work would be to explore visual attention mechanisms that operate on the images and investigate their role in meme generation tasks, based on publishings such as [13].

7 Contributions

A L Peirson V: (enrolled in both CS230 and CS224n) Scraping dataset, data conversion to tfrecords and preprocessing, encoder visual system implementation, alexnet and inception comparison, beam search modification, attention encoder LSTM implementation, glove averages implementation, training glove average models, perplexity evaluation, caption to meme code.

E Meltem Tolunay: (enrolled only in CS224n) data visualisation, attention theory and implementation, attention beam search implementation, glove averages theory and implementation, training attention models, perplexity evaluation, glove embedding, human testing.

Everything in this project (other than the human testing and a small part of the GloVe averages implementation and the concatenation idea) was thought of and done by the lead author. The second author worked mainly on the attention side of things for CS224n [7], not mentioned here. The alexnet and inception comparison and the image only model is solely for CS230, while the glove averages model is shared between CS224n and CS230.

References

[1] ImageNet. http://www.image-net.org/.

[2] Meme Generator | Create Your Own Meme. https://memegenerator.net/.

[3] Meme, Mar. 2018. Wikipedia, https://en.wikipedia.org/w/index.php?title=Meme&oldid=830994195.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*, Sept. 2014. arXiv: 1409.0473.

[5] C. Gan, Z. Gan, X. He, J. Gao, and L. Deng. StyleNet: Generating Attractive Visual Captions with Styles. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 955–964, July 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[7] E. M. A. L. Peirson V, Tolunay. Dank Learning: Generating Memes Using Deep Neural Networks. 2018.

[8] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to Sequence Learning with Neural Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. *arXiv:1512.00567 [cs]*, Dec. 2015. arXiv: 1512.00567.

[11] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and Tell: A Neural Image Caption Generator. *arXiv:1411.4555 [cs]*, Nov. 2014. arXiv: 1411.4555.

[12] C. Wang, H. Yang, C. Bartz, and C. Meinel. Image Captioning with Deep Bidirectional LSTMs. *arXiv:1604.00790 [cs]*, Apr. 2016. arXiv: 1604.00790.

[13] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv:1502.03044 [cs]*, Feb. 2015. arXiv: 1502.03044.