Pic2Plot

By: Gianna Finear (gfinear1), Catherine Lasersohn (claserso), Alan Gu (agu10), Edward Xing (exing1)

Github Link: https://github.com/gfinear/dl-final-project

Introduction:

Our project seeks to generate a story synopsis based on an image. From the image, our model extracts important features via image captioning and uses them as inspiration for the story. We were inspired by existing papers on content generation but wanted to explore further possibilities. For example, Microsoft developed a model that generates poetry from images. This model uses multi-adversarial neural networks along with an embedding model. First, the images are processed with an embedding model to match image features to poetry. Then, the results are passed through an RNN generator before being refined with discriminators. The paper evaluates the model using poetry-specific metrics, such as style and subjectivity which are less relevant to plot summaries since summaries are less abstract than poetry. Microsoft's invention inspired us to use what we learned in deep learning alongside what we learned in computer vision and artificial intelligence to create a style-shifting model to generate a synopsis from an image. Many writers face writing blocks and we believe this tool could help writers capture an image and turn it into an idea that inspires a novel, movie, or play.

Methodology:

Data/Preprocessing:

We began by gathering and preprocessing our data. To extract significant words from an image, we used the Flickr 8k dataset from the Image Captioning homework. We did similar preprocessing as in the homework. First we processed all images by resizing them and extracting important image features using ResNet50. Then we processed all captions by removing stop words and punctuation, and padding each caption to our desired window size. To generate a synopsis based on these words, we combined a book summary dataset with a movie synopsis dataset. We took the synopsis data from both of these datasets and preprocessed them to be a uniform length of 150. Similar to the captions, we removed all punctuation and discarded words

with special characters in them. Additionally, we replaced rare words (words that appear fewer than 500 times in the synopsis data) with a filler to make our model more accurate. Both the caption and summary datasets had vocabularies consisting of close to one thousand words each. We increased the threshold frequency for rare words to account for the larger corpus size and exclude placeholder information, such as character names, while preserving plot points.

Architecture:

Our model architecture was largely based on existing work developed for image to text applications, namely image to poetry and image to story. The model architecture may be thought of in two parts: caption generation followed by summary generation. Each component was trained separately and deployed in conjunction.

Image caption generation is used to convert images to text form. This component consisted of a convolutional neural network (CNN) for image feature extraction and a transformer neural network for caption generation. We tried a few different models for caption generation: a predictive model such as in the image captioning homework and a visual-semantic embedding, which maps captions and images to a common embedding space. We trained this model on the Flickr 8k dataset and found that the transformer neural network provided the best results in terms of perplexity and accuracy of printed captions.

The skip-thought vector model is used to learn skip-thought embedding representations for novel summaries. Skip-thought vector embeddings are task-agnostic semantic embeddings of sentences. The encoder-decoder model is trained in a self-supervised manner similar to an autoencoder. The encoder seeks to capture the most important semantic information from which the decoder can most accurately reconstruct neighboring sentences. Our architecture only uses the encoder since it is able to map into a latent embedding space with useful semantic properties.

Since the skip-thought vector space captures semantic information, the style of the text can be changed without changing its meaning. A style transfer function is necessary to convert from a short descriptive caption to a full plot summary. This function takes the form:

$$F(x) = x - c + s$$

where x is our image caption skip-thought vector, c is the average of our image caption skip-thought vectors, and s is the average of our plot summary skip-thought vectors. Intuitively,

we maintain the semantics of the image caption, while replacing its structure with that of a plot summary. The resulting skip-thought vector can be decoded into a summary.

The decoder is the final component of the summary generation model. The decoder is trained to decode the skip-thought vectors of summaries to their corresponding passages. It is implemented as a Recurrent Neural Network (RNN) with Gated Recurrent Unit (GRU) cells that use the encoded summary skip-thought vectors as the initial hidden state. The semantic information captured by the skip-thought vectors are used to inform the predictions of the RNN. By decoding the style-transferred caption into a summary, we can approximate a function to convert from a caption into a summary without needing a parallel corpus to train on.

Combined, our pipeline generates an image caption, encodes it into a skip-thought vector, performs style transfer, and decodes it into a text summary.

Results:

The caption model generates relevant and readable captions on a wide variety of images. The captions are able to capture the most important elements of the image and with varying temperature parameters, can be quite creative. The model trains to a perplexity of around 15. For example, an image and its corresponding caption are:

"two people in winter clothes are riding down snowy hill"

The summary model does not perform as well as the caption model. Since the skip-thought encoder used was pre-trained, the decoder itself was trained in isolation. The decoder trains to a perplexity of around 50, which is much lower than our outlined stretch goals. Although the model can produce certain phrases that make sense, the full summaries produced

by the model are too inaccurate to produce sensible sentences. However, the presence of relevant words in the generated summaries implies that the idea of the image may have been preserved by the decoder. The image above results in the following summary with a temperature value of 0.1:

> *"back there man he was all this can them tells over home some out meets one their woman into when own from death how <unk> she has one up not when friend tells her and which however her then the can two through its save in love on one are now be who their son an daughter into able all where they out more then both they after an they both up out down with all later has becomes up going later woman <unk> at stop on before so in when house for friends is over but home its while under as day from months found new it two found out and had into school friend so so with where was later by who can this when before would he who as more had him john life had as before and seen father house a its found but also"*

The failure of the decoder model potentially lies in the quantity of data used to train. Our data set was comprised of only 25 thousand passages. Similar language models are trained on tens of millions of passages. Given the limited data, the model may not have seen enough examples to learn the structure of the English language to be able to accurately generate sensible summaries. The presence of simple correct phrases implies that the model only learned the most common phrases which were encountered frequently. More complex sentence structures were not learned because they occurred less frequently.

Another potential failpoint considered was the types of style that skip-thought vectors could capture and differentiate. Textual styles vary amongst many different dimensions such as composition and genre. Possibly, skip-thought vectors could not capture differences between the style of captions and summaries. However, both t-SNE and PCA illustrate that the differences between captions and summaries are captured by skip-thought vectors.

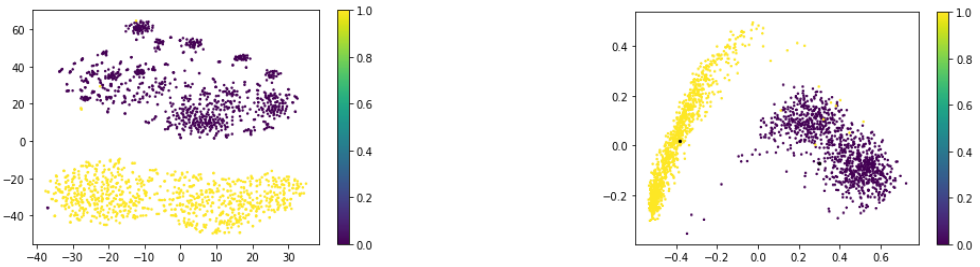

Figure 1: t-SNE and PCA projections

In our attempts to create a better performing decoder, we tested a wide variety of architectures and hyperparameters. Ultimately, we found that smaller architectures performed better as they were less prone to overfitting on our relatively small dataset. Our final architecture consisted of 430 thousand parameters. While generating our summaries from the logits produced by the RNN, we only sample from the k most probable succeeding words, rather than all the words in the vocabulary. We also experimented with different temperature values, and found the best results using a temperature value of 0.1 with a k value of 100.

Challenges:

We faced challenges related primarily to the data we used for training and validating our models, the multi-component aspect of the model architecture, as well the training of the model.

We use three datasets, the Flickr8k dataset for image captioning, and the CMU and Kaggle datasets for book and movie summaries. The number and sizes of the datasets imposed restraints on our ability to collaborate effectively. In addition, we relied on a number of pretrained models which were also quite space intensive and, at times, restricted the pace of software development.

Our project involves many components: image captioning, style transfer, skip-thought embedding, and skip-thought decoding, which also complicated our approach. Firstly, we faced challenges with respect to maintaining an organized code base and directory structure and hierarchy. Additionally, we had to invest time analyzing each piece of the project to determine what could be accomplished based on our prior knowledge, and the architectures we had familiarity with based on class. For instance, we determined that the image-to-text component could be performed using transformer-based image captioning, and we determined that skip-thought decoding could be accomplished in a manner similar to machine translation using an RNN, despite translating between two textual styles instead of languages and lacking a parallel corpus. On the other hand, we were unable to implement a robust skip-thought embedding because of data and computation limitations, and we had to resort to a pretrained model, which posed additional challenges. The open source model was outdated, i.e. running an older version of Python with calls to deprecated libraries, so a large amount of time was invested updating the code base to work within our model.

Due to the size of the overall architecture, training and running the model requires high computing power. As a result, training was done on GPUs to accelerate the process. Sourcing as well as setting up the GPUs was a new challenge. Even with GPU acceleration, testing different models was still time consuming, resulting in slow development pace.

Reflection:

Despite the roadblocks we faced, we were able to implement our original idea and generate results that surpassed our stretch metrics, despite their content being largely unreadable. We ended up using a much larger data base than expected. This made us adapt our model slightly from what we originally expected. We decided to use a pre-trained model rather than encoding the skip-thought space. This was because we were unable to embed a robust latent space because it was too computationally expensive. However, with extensive research we were able to find a detailed open-source pre-trained model that would be able to be adapted to our data. This also made us change our idea for our decoder because we needed to adapt it to work with the pretrained model. If we were to do this project over with more time, computational power, and text data, both in terms of quantity and style variation, then we would like to experiment with training our own skip-thought embedding space.

In terms of our generated summaries, we believe that our failure to meaningfully attach a synopsis to an image can mostly be attributed to our decoder. Originally, we hypothesized that our decoder's ineffectiveness may be related to the size of our vocabulary, where the large number of words contributed to underfitting in the decoder. We were able to decrease our perplexity scores by increasing our threshold for inclusion in the vocabulary, i.e. decreasing our vocabulary size; however, this did not translate to better summaries. Due to the relatively small size of our summary dataset, we also hypothesized that we may be overfitting with larger models greater than a few million parameters. This proved correct, and we were able to train smaller models of a few hundred thousand parameters with much lower perplexities. Ultimately, however, lowering our decoder perplexity to our stretch goals did not produce sensical summaries. We hypothesize that this is caused in part by errors compounding across multiple components; imperfect generated captions are used to create the skip-thought vectors from which we generate summaries, resulting in imperfect summaries despite our decoder achieving relatively low perplexity. Furthermore, our summaries have a maximum length of 150,

drastically reducing the probability that each word chosen is sensible as it grows in length. Our current iteration of the decoder performs much better than our original plan, however, if we had more time we could improve the summaries by exploring new architectures that would allow us to reach even lower perplexity. We believe that our current loss minimization task is ill-defined in some way; it is possible that the training data is not reflective of our target task, or that the skip-thought embedding space itself is not conducive to encoding our captions. To further improve our project, we would investigate these possibilities and consider alternate approaches to this task.

If we were to continue to make changes to our model there are some other small changes we would implement. The first of which would be to add punctuation to support an official summary generator rather than a summary idea generator. However, we believe this could lead to ethical dilemmas because it would allow someone to pass off our generation as their own work. This was the main reason we did not adapt our model to do this originally. Since our generator is trained on real movie and book synopsis' this would mean that new summaries generated with our model would not be completely original and unique. This is part of the reason why we kept it as a summary idea generator so it could spark an idea and inspire a book or movie but could not be used exactly as it draws upon other's work. We were not able to think of a way to protect the work we generate to ensure it was used ethically, thus we decided the only safe way was to ensure what we generated would not be able to be used immediately or easily passed off as someone else's work.

In terms of technology, the biggest takeaway was skip-thought vectors. This was an intriguing piece of technology that none of us had previously used. The skip-thought vector allowed us to create task agnostic word representation of text. This allowed us to maintain style in a better manner and be a more accurate translation than traditional word2vec methods we learned in class. Additionally, a major takeaway was that translation is more than just translating languages. Previously, we solely thought about translation as a way to transfer text into various languages; however, this project showed us that it can also have important implications on translating style. Now we can take a piece of text that is written in English and a specific style and translate it so that it is still in English, but in a different style. This allows the original body of text to elicit different emotions and understanding out of readers while preserving the idea.