Generate synthetic data to train image captioning models

Abhishek Singh
abs006@ucsd.edu

Chang Zhang
chz230@ucsd.edu

Umesh Singla
usingla@ucsd.edu

Abstract

(1)The image captioning task is to first understand the content in the image and then generate a concise description of it. Image Captioning models are designed in such a way that encoder takes an image as input and then the image representation is used in the decoder model to generate text describing the image. Given that this task involves both image representation and text generation, it requires a large parallel corpus of images and corresponding text to train such a model. In this project, we propose a data augmentation technique that involves four levels of optimization to improve the performance of existing models and given datasets. For most purposes, models trained on the synthetic data instead of the real data do not show a significant loss of performance.

1 Introduction

The task of image captioning requires understanding the image which involves the subject, background, activities, and their relationship (2). Once the image and its attributes are represented as vectors they are used to generate syntactically and a semantically correct short description of the image.

Various deep learning methods have been developed for image captioning tasks. Attention models are trained to learn to fix their gaze on salient objects while generating the corresponding words in the output sequence (3). Encoder-decoder model which uses CNN to encode image features and RNN to generate the captions gives good results when trained end to end using stochastic gradient descent. Reinforcement learning is used to optimize the image caption generation task and it is shown that by carefully optimizing the system it is possible to improve the model's performance (4).

Most of the techniques developed so far either try to increase the size of the models or increase the size of the training corpus to improve their performance. In the past, it has been shown in various applications that data augmentation can be helpful to improve the performance of existing models and datasets (5). In this project, we work on a unified four-stage framework that generates synthetic image and caption pairs in addition to existing training data to train models for image captioning.

This four-stage optimization technique can be applied to any encoder-decoder model, which means that the method is not specific to any specific text generation of the image captioning model. The unified framework has four stages which are trained together: 1) Caption generation for initial random tokens. 2) Image generation using a conditional Generative Adversarial Network (cGAN) model. 3)

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Training encoder-decoder image captioning model that uses training as well as generated data. 4) Using validation performance to update parameters used in stage 1 and stage 2.

2 Related Work

2.1 Image Captioning

The earlier works of image captioning, such as (6), project images onto a meaning space where then can be further mapped onto the sentence space for captioning. Describing the content of an image is a challenging problem in artificial intelligence that connects the two disciplines of computer vision and natural language processing - as it requires a fine-grain understanding of image objects as well as their attributes and relationships. Despite substantial progress in recent years, image captioning techniques are still far from being perfect. Sentences produced by existing methods, e.g. those based on RNNs, often lack diversity and seem inflexible.

There have been a variety of work on the image-captioning problem recently. Initially inspired from the machine translation, the Encoder-and-Decoder paradigm became popular where an "encoder" RNN reads an input sentence and transforms it into a rich vector representation, which in turn is used as the initial hidden state of a "decoder" RNN that generates the target sentence. Some works (Mao et al. 2014 (7); Vinyals et al. 2015 (8)) then proposed to replace this RNN with a CNN to encode image features. The CNN-RNN architecture has been one standard configuration of image-captioning models since then. Some of the networks include but are not limited to multi-modal CNN, and bidirectional LSTM (9). The most recent state-of-the-art approach uses pre-trained language generators, such as BERT, XLNet ,and GPT (10), to achieve cohesive and coherent texts.

Another state-of-the-art method for this task is using the maximum likelihood principle (MLE) for learning. Using the MLE approach to train a generator to describe images suffers from the problem of generated sentences containing "repeated patterns" since it encourages high resemblance to the "ground-truth" captions used during training and suppresses other potential descriptions (8). Numerous caption generation methods and evaluation metrics in the literature primarily focus on the similarity to the training samples while following the MLE principle. While this is a safe way to generate potential descriptions, it lacks variability. To solve this, there has been work on generating image descriptions recently that try to balance a few key properties of human expression - fidelity, naturalness and diversity while evaluating on one of the language metric scores (BLEU, CIDEr, SPICE, etc.) or a combination of them. Although, in the current work, we will be following the encoder-decoder based on the MLE approach.

Dai et al. 2017 (11) is one of the first few studies that explored the use of conditional GANs to generate diverse and natural captions conditioned on images. Their model uses a VGG16 as the CNN architecture to encode the image and an LSTM net for both the generator and the evaluator where the parameters for the two are independent of each other. Lastly, to train, their model uses a similar policy gradient from reinforcement learning literature where the reward signal comes from the evaluator and is a weighted combination of measures of naturalness and semantic relevance.

Further, there has been work on improving image captioning with CGANs using a classification-based evaluator (12) instead of relying on language metrics (BLEU, etc). The classification-based evaluator learns to classify whether the generated caption is human-described or machine-generated.

2.2 Generation

Data synthesis is often used to tackle issues that arise from having insufficient data. One of the first uses of synthesized data comes from the classification domain. Research has shown that the use of synthesized data produces high-performance in classification tasks (13). In recent years, the computer vision field has also begun to adopt the use of synthetic data. Synthesized data has become a popular method in training deep learning models in problems, such as object detection or autonomous driving, where collecting a large sample of real-world data is difficult. (14)

One approach to generating synthetic data for computer vision problems is using 3D environments from video games. In Qiu et. al. (2016) (15) they generated images from a video game engine using various camera angles and light settings. Even though those images do create virtual realism; yet, this approach is challenged by the lack of content variability and the structures of 3D mesh.

GAN and conditional GANs have also been popular methods for learning generators in computer vision to generate images and overcome some of the problems that arise by following the MLE. Yu et al 2017 (16) proposed a text generator-based SeqGAN to generate structured sequences that uses an RL expected future reward signal by the discriminator to provide early feedback to the generator using policy gradient via Monte Carlo search. It is an unconstrained generator that do not take into account any conditions.

In Mirza and Osindero 2014 (17), they demonstrated the potential of Conditional-GAN (cGAN) by training a cGAN on the MNIST handwritten digits dataset conditioned on the class labels. One advantage of cGAN is that they are able to utilize the additional information given by the class labels to solve problems that have a large number of predicted output categories. Another advantage of cGAN they stated is that we can control the mode of the data that is being produced using a conditional probabilistic model. For example, in our usage of cGAN, we are able to control the type of images being generated based on the caption the image is conditioned on. At the end of their paper, their conjecture is that the results would improve if the model is conditioned on multiple tags for each training sample, which is what we are trying to accomplish with cGAN - with the vectorized captions being the "multiple tags".

As reported in additional studies, ((18); (19)), conditional GANs can produce natural images nearly indistinguishable from real photos, freely or constrained by conditions. Overall, with fine turning, CGAN is able to compete with other state of art generative models and it is suitable for tasks where there are additional information given.

2.3 Generative Pre-trained Transformer

LSTM has been a popular tool in text generation, however, training using LSTM is very inefficient because word prediction cannot be parallelized. To solve this issue, the transformer model was proposed. In a transformer model, hidden tokens are predicted in parallel, hence it achieves a more efficient computation structure. In recent years, fine-tuning a generative pre-trained transformer has become a popular method to generate human-like text. GPT-2 is one of the pre-trained transformer models. It contains 1.5 billion parameters and these parameters were trained on a massive number of websites. With a large parameter size and heavy training, GPT-2 is able to perform a variety of tasks without explicit supervision (20) One application is audio captioning. (21) In their paper, they used an encoder to convert audio input into a token sequence. This token sequence is then processed by a decoder where GPT-2 is used. They found that the performance of using the pre-trained transformer model is superior to the performance of training conventional models from sketch. Another application of GPT-2 is video summary generation. (22) In the paper, they devised a framework that utilizes a video summary controller, an interactive attention network, and a video summary generator. GPT-2 was used in the video summary controller to effectively encode input text-based inquiry. These two studies have shown us the use of GPT-2 in text-generation tasks, as well as its promising performance.

2.4 Neural Architecture Search

In this work, we have used Neural Architecture Search to search the optimal architecture of discriminator used in the GAN model. Though most well-known and successful architectures have been developed by researchers, it need not be the most optimal in the search space of the components of the architecture. The most naive way to search can be a random search. However, this has proved to be very effective in hyper-parameter search (23). Reinforcement learning-based algorithm is used to propose a child model for evaluation, and the reward being higher accuracy(24). This method works even when the reward is non-differentiable. Evolutionary Genetic Algorithms are also used to search for high-performance architectures. AmoebaNet (25) uses Genetic Algorithm (GA) to produce sets of mutated and samples the best of them for the next iteration. Differentiable Architecture Search (DARTS) was proposed to make the architecture search differentiable over continuous search space (26). DARTS introduces a continuous relaxation on each path in the search supergraph, making it possible to jointly train architecture parameters and weights via gradient descent.

3 Implementation

In this problem we have access to training set which contains image and its corresponding caption which is $D_{tr} = (x_i, y_i)_{tr}$ and validation data of image and caption pairs $D_{val} = (x_i, y_i)_{val}$. The proposed optimization framework is divided into four stages.

3.1 Motivation

In this method we leverage existing architectures and datasets to optimize the models further, therefore it does not require large datasets or models. There are four stages in the optimization of the framework which we train as a unified framework. When we generate a new caption from random input we weigh the generated caption, which means that if the generated caption is useful in improving the performance it is promoted whereas if generated caption does not help it is given less importance. During this process, the framework learns to generate good quality sentences that can be used as captions.

Then we use training data of image and caption pairs to train a Conditional Generative Adversarial Network, which learns to generate new images if we give caption as input and add some noise to it. This way we are able to generate new image-caption pairs from existing dataset. We leverage this generated data along with training data to train the image captioning model. Eventually, we evaluate the trained model on left out data to update weights of the generated captions and search for the optimal discriminator architecture of the GAN model.

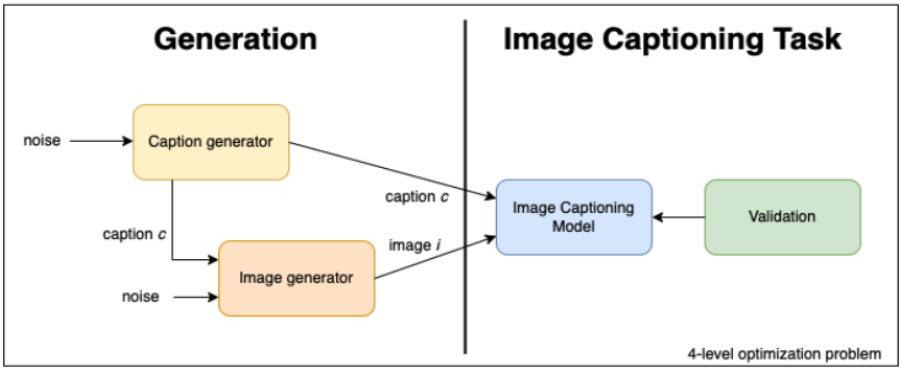

Figure 1: Proposed model architecture

When we put all the pieces together of the framework it gives an optimized model that can leverage existing data and model to improve its performance (Figure 1).

3.2 Background

Generative Adversarial Networks: Generative adversarial networks (GANs) consist of a generator G and a discriminator D that compete in a two-player minimax game: The discriminator tries to distinguish real training data from synthetic images, and the generator tries to fool the discriminator. The distinction is based on a classification task that D is trained on but in some of the works,the discriminator is also referred to as evaluator when the distinction is based on another metric (like a quality score). Overall D and G play the following game on V(D,G) if the evaluation is formulated as a classification task:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D(G(z)))] \quad (1)$$

Goodfellow et al. (2014) (27) in the original formulation of GANs prove that this minimax has a global optima when $p_G = p_{data}$.

Joint Embedding for sentence representation: To train sentence representation we use the methods as proposed by Reed et al. 2016 (28). Images are encoded using ResNet50 pretrained model and

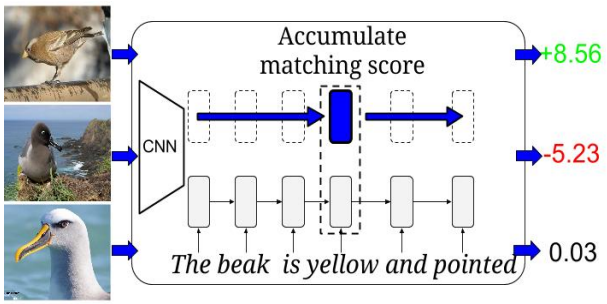

Figure 2: Training Sentence representation

outputs are mapped to desired feature length. Sentences are first tokenized using pretrained subword level BPE tokenizer. These token are then used as input to bidirectional RNN model to encode the sentence representation to desired feature length (Figure 2). Then these representations of images and captions are used to calculate loss and train the model. The minimization task of the loss function of the model is defined as:

$$argmin_{f_v} argmin_{f_t} \frac{1}{N} \sum_{n=1}^{N} \triangle(l_n, f_v(v_n)) + \triangle(l_n, f_t(v_t)) \tag{2}$$

Where l_n is the class label, f_v is the encoder of images, f_t is the encoder of sentence descriptions and $\triangle$ is the 0-1 loss. The class label is the index inside of a given batch. Essentially we treat each pair as its own class. This trained sentence encoder is then used to encode the captions and fed as input to further stages of the task.

3.3 Problem Formulation

3.3.1 Stage 1: Caption generation from random input

In the first stage, we train text generation model GPT2 on $\{y_i\}_{i=1}^{N}$, which takes a few initial tokens as inputs and generates a sentence using input tokens. Let G denote the network weights of the GPT2 model. The training loss L will be the language modeling loss . Candidate options are perplexity, cross-entropy, etc.

Each training example $\{y_i\}$ is weighted by $a_i = [0, 1]$ and we solve the following problem:

$$G^*(A) = argmin_G \sum_{i=1}^{N} a_i L(y_i; G) \tag{3}$$

At this stage, weights of GPT2 model are updated using gradient decent but weights a_i are not updated at this stage as it will yield a trivial solution, with all the weights a_i getting very close to 0 so that the loss value is nearly 0. Using this step, we are able to generate a sentence using only a few random initial cues using text generation model.

3.3.2 Stage 2: Image generation from caption

At the second stage, we use conditional GAN based approach to model the image-generation task. We use the given training dataset $(x_i, y_i)_{i=1}^{N}$ to train this generative adversarial net M. This cGAN model will eventually help us generate an image given a caption. We use a conditional version of GAN because by adding the caption y as an additional parameter to the generator, we hope to have more control over the modes of the images to be generated and may give a significant head start to GAN for what to look for. During training both caption and image is given as input to the model and during inference only caption is used as input to generate the image.

Let E and F denote weights of the generator and discriminator. Let B denote the architecture of discriminator. At this stage we solve the following problem:

$$E^*(B) = argmin_F argmax_E L(\{x_i, y_i\}; E, F, B) \tag{4}$$

In the above stage weights of the generator and discriminator are updated where as the architecture of discriminator is not updated at this stage. If architecture B is trained by minimizing the loss function the architecture can become complex and therefore overfit on the training dataset and eventually perform poorly on unseen data.

3.3.3 Stage 3: Using the generated dataset to train a image-captioning task.

Augmented data: Now that we have a setup for caption generator as well as image generator given a caption. At the third stage, we use G*(A) from stage 1 and E*(B) from stage 2 to generate a synthetic image captioning dataset. Given a random vector z, it is fed into G*(A) to generate a synthetic caption $f(z; G^*(A))$. The caption is then fed into E*(B) to generate an input image $g(f(z; G^*(A)); E^*(B))$. The generated data, $g(f(z; G^*(A)); E^*(B))$ and $f(z; G^*(A))$, form a (image, caption) pair that we can use to train our image captioning model in addition to the given training dataset. We denote the generated dataset at this stage as $S(G^*(A), E^*(B))$.

We use W to denote the the weights of image captioning model. As mentioned before, we calculate the loss L on the training dataset $data$ as well as the generated dataset S and combine both losses using a weighing hyperparameter (λ). The formulation of this step is as follows:

$$W^*(G^*(A), E^*(B)) = argmin_W \; L(\{x_i, y_i\}_{i=1}^N; W) + \lambda L(S(G^*(A), E^*(B)); W) \tag{5}$$

In the above equation, first loss term denotes the loss on training dataset whereas the second term denotes the weighted loss on generated dataset. G*(A) and E*(B) are the optimized captions and image generation models from stage 1 and 2.

3.3.4 Stage 4: Measuring performance on validation set

Once we have optimized the image captioning model we measure the validation performance of W^* on the validation data D_{val}. At this stage we use the validation loss to optimize the weights A used at stage 1 and the architecture (B) of the discriminator used at stage 2 of this framework. This helps to unify the four stages of the framework together.

We formulate this optimization stage as:

$$min_{A,B} L(D_{val}; W^*(G^*(A), E^*(B))) \tag{6}$$

The validation loss calculated above is used to perform gradient decent update on the parameters A and B of the framework used in stage 1 and stage 2 respectively. This way we can avoid overfitting of this framework.

Putting these four stages together leads to a unified framework that can optimally generate synthetic data during the process and use it to train an image captioning model.

$$\begin{aligned}
&min_{A,B} L(D_{val}; W^*(G^*(A), E^*(B))) \;\; s.t. \\
&W^*(G^*(A), E^*(B)) = argmin_W \; L(\{x_i, y_i\}_{i=1}^N; W) + \lambda L(S(G^*(A), E^*(B)); W) \\
&E^*(B) = argmin_F argmax_E L(\{x_i, y_i\}; E, F, B) \\
&G^*(A) = argmin_G \sum a_i L(y_i; G)
\end{aligned} \tag{7}$$

3.4 Evaluation

BLEU: To evaluate the quality of the generated captions, the most commonly used metric in the literature is the BiLingual Evaluation Understudy metric, which is a form of precision of word

n-grams between generated and training sentences (29). To evaluate the model's performance on the validation set, we will use the automated BLEU evaluation metric. This evaluates a generated caption against reference caption(s). For each generated caption, we will use all captions available for that image as the reference captions.

Beam Search: To avoid the decoder to simply choose the words with the best score at each decode-step, we employ beam search. Beam Search is useful for any language modeling problem because it finds the most optimal sequence. It chooses the sequence that has the highest overall score from a basket of candidate sequences by keeping track of top k best choices for the previous word.

4 Dataset

We train and validate our model on two benchmark datasets: Flickr30K and MSCOCO.

Flickr30K consists of 31,783 images and 158,915 captions obtained via crowd-sourcing. Each caption has a one-sentence description that concisely summarizes the most significant features of an image that could be of an everyday activity, an event or a scene. Each image is described independently by 5 annotators who are not familiar with the specific entities and circumstances depicted in them. Different annotators use different levels of specificity, from describing the overall situation to specific actions (30).

MSCOCO caption dataset is a large-scale object detection, segmentation, and captioning dataset. It also has 5 captions per image (31).

5 Model

5.1 Stage 1: Caption generation from noise

We use **DistilGPT2** as our pretrained model of caption generation. Under the supervision of GPT2, DistilGPT2 is an English language model trained on OpenWebTextCorpus. Fakes captions are produced by feeding the model the first 5 words of the real captions and the model then fills in the rest to make a complete caption. We set the maximum length of the generated caption to be 20. The beam size is set to 2 so the model will only consider the top 2 most probable words at each timestep when computing the most probable sequence of words. Setting the beam size low is also for speeding up the caption generation task as the higher the beam size the more expensive it is for the model to generate a caption.

5.2 Stage 2: Image generation from captions

We call our image generator model **Stage2GAN**.

We use a deep convolutional GAN (DCGAN) conditioned on text features. A text encoder ψ based on a character-level CNN as described in the background section is used to get the text features. Both the discriminator D_m and the generator G_m perform the inference conditioned on the text features. This model is inspired from Reed et al. (2016) (19). Figure 3 describes the model.

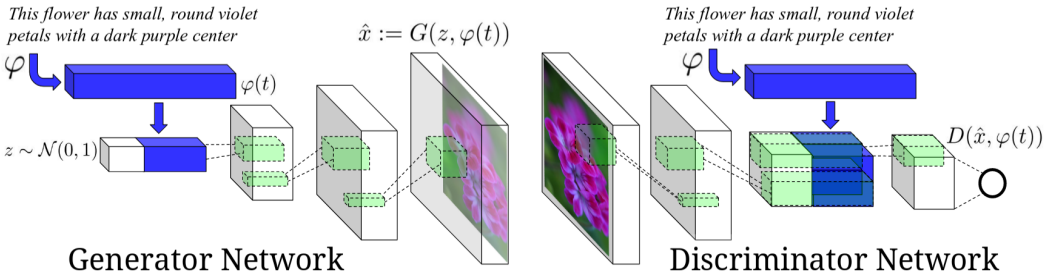

Figure 3: Conditional GAN model for image-generation task. Figure architecture from Reed et al. 2016 (19).

In the generator G_m, we first sample z from the noise $N(0, 1)$ and encode the text query t using text encoder ψ. The embedded text is then compressed to a small dimension using a fc layer and then concatenated to the sampled noise vector z. Following this, we feed-forward it through the deconvolutional and batch normalization layers of the generator network G_m. The **generator** is composed of one linear layer and five convolution layers. Each layer is normalized and activated by a ReLU function. The text embedding is first projected onto a 128 by 1 embedding space using the linear layer. We then concatenate a sampled noise z, which has a size of 100 by 1, to the embedding space to form the latent space. Then the convolution layers deconvolve the latent space into a 64 by 64 by 3 image space.

In the discriminator D_m, we have multiple layers of convolution with batch normalization and leaky ReLU. In a separate fc layer, we again decompress the text into $\psi(t)$ using the text encoder. This text embedding is depth-concatenated with image feature maps in the one of the convolution layers in the D_m. A sigmoid is performed at the last layer to get the final score from D_m that distinguishes between generated and real image-caption pair.

This classification objective $L(G_m, D_m)$ can be formulated as a mini-max game, similar to the original GAN:

$$\min_{G_m} \max_{D_m} L(D_m, G_m) = \mathbb{E}_{x \sim p_{data \cup S}(x)}[log D_m(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D_m(G_m(z)))] \tag{8}$$

The text embeddings are generated using LSTM model which is trained as mentioned in 2. We use the model to produce a 1 by 1024 vector that carried the text representation of a given caption.

5.3 Stage 3: Train Image-Caption model

We model the task of caption generation given an image by an encoder-decoder architecture with an additional attention mechanism from Kelvin et al. 2015 (29). It is based on work in machine translation where an "encoder" RNN reads the source sentence and transforms it into a rich fixed-length vector representation, which in turn in used as the initial hidden state of a "decoder" RNN that generates the target sentence. Here, given that our input is an image , we will be using a CNN as the image encoder to get a rich representation of it in a fixed-length vector. For our encoder CNN, we choose to use ResNet-101, a 101-layer pre-trained network.

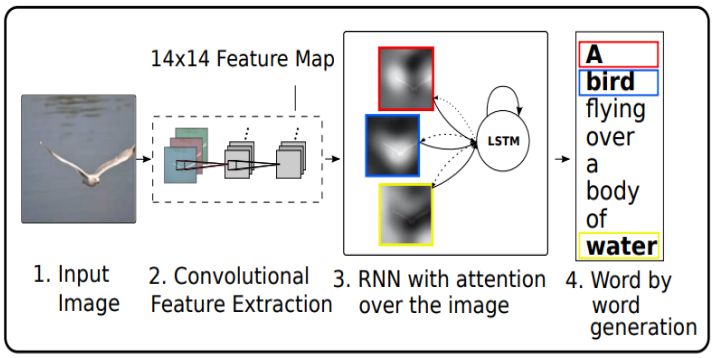

Figure 4: CNN-LSTM with Attention model for image-to-caption generation task. Figure architecture from Kelvin et al. 2015 (29).

We then use the embedding from the last layer in CNN as an input to the decoder recurrent network to generate a caption. We use LSTM with an Attention network as our decoder recurrent network. The Attention Network computes the importance of each pixel (attention weights) from the given generated sequence and LSTM uses the weighted pixels to generate the next word. The architecture of our caption generation model is shown in Figure 4.

The LSTM model is trained to predict each word of the caption S after it has seen the image as well as all preceding words ($S_0, S_1, S_2, ..S_{N-1}$) where N is the length of the caption. The image I is only input once, at t = -1 (Figure 5).

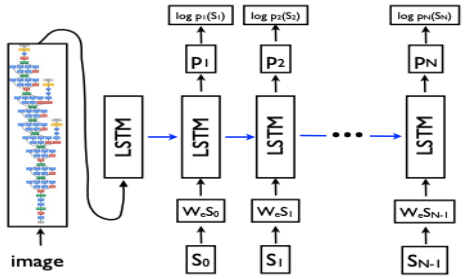

Figure 5: CNN-LSTM model for caption generation task. Figure architecture from Vinyals et al. 2014 (8).

The loss for each training image-caption pair is defined as the sum of the negative log likelihood of the correct word at each step as follows:

$$L(I, S) = -\sum_{t=1}^{N} \log p_t(S_t) \tag{9}$$

The images are resized to (256,256) and the captions are one-hot encoded. We add <start> and <end> tokens to each caption to signal start and end of each sentence and we add <pad> tokens fix the length of each sentences. For the **encoder** ResNet-101, we discard its final two layers, and we fine-tune its convolutional blocks 2 through 4 when we train it with our images. Keeping the rest of the convolutional blocks the same allows us to retain the model's fundamental skill sets like edge detection, line detection, etc. For the **Attention network**, we have three linear layers. The first two layers take in the encoder outputs and decoder hidden layers. The last layer connects the sum of the first two layers and outputs the attention weights α, which have the same dimension as the encoder output. We use soft attention for this task, which means all the attention weights sum to one. This is achieved by using the softmax function on the last linear layer.

LSTM then uses the attention weights, in addition to the image representation, to generate captions. The initial hidden state and the initial cell state of the LSTM cells are initialized by two separate linear layers. A gate is placed onto the attention weights that is activated by the previous hidden state. This gate is to allow the RNN to place more emphasis on the objects in the images (29). In each batch, we sort the image-caption pairs by the length of the captions in descending order. This way we are able to batch process the image-caption pairs at each timestep once we align the start of each sentence. The **loss** function for the training process is reformulated to have additional term on attention weights along with cross-entropy loss as defined previously.

$$L = -\sum_{t=1}^{N} \log p_t(S_t) + \lambda \sum_{i}^{L} (1 - \sum_{t}^{C} \alpha_{ti})^2 \tag{10}$$

The first term is the cross-entropy loss with the paddings excluded. The second term introduces double stochastic regularization, which encourages $\sum_t \alpha$ to be close to 1. Intuitively, this is to encourage to the model to focus on every pixel equally.

6 Experiments and Results

6.1 Stage 1

In stage 1, we use a pre-trained GPT to generate captions from noise without any modifications. Some of the generated captions are listed in Table 1.

1	A young man sits on a bench in the middle of an empty street.
2	Two constructions workers sit on the ground in front of a building.
3	Three people hang out on the street, and they are not going to be able get a job.
4	The black dog runs through the streets of New York City.
5	A person climbs a tall building in the middle of an industrial park.

Table 1: Example generated captions at Stage 1.

6.2 Stage 2

For the training task, we set the learning rate of the generator to be 1e-4 and that of the discriminator to be 1e-5. For each epoch, we update the parameters of the discriminator once for every five generator update. While training the discriminator, the weights for the generator is not updated. Likewise, the weights for the discriminator stays the same while training the generator. Both networks use Adam optimizer with beta-1 equals 0.5 and beta-2 equals 0.999. Batch size is set to be 64. The data are processed prior to training. The images are reshaped to 64 by 64 and the captions are tokenized.

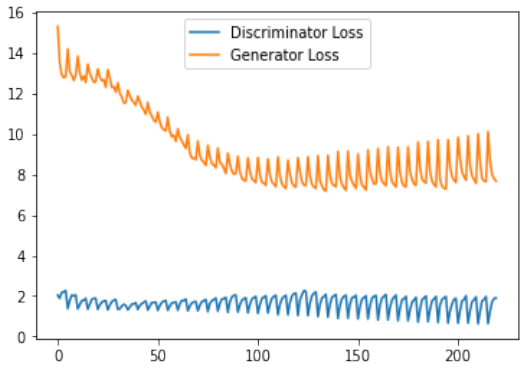

Figure 6: Generator and Discriminator loss during the training.

We trained the model for 200 epochs on Flickr8k dataset. The generator loss reduces from 15.8 and then reduces to 8.6 whereas discriminator loss keeps oscillating during the complete training between 1 to 3 as shown in Figure 6. Sample images generated by Stage2GAN are in Figure 7.

(a) Generated Images from Stage2GAN model.

(b) Images from Flickr8k dataset.

Figure 7: GAN generated images and corresponding data

Comparison with other studies: Our conditional GAN model is trained by creating representation vector of the complete sentence, which lacks fine-grained word-level information and prevents generating high quality images. To compare images generated by our Stage2GAN model, we used a pre-trained attention based GAN model called **AttnGAN** on MSCOCO dataset to generate images from synthetic captions retrieved from stage 1, by Xu et al. 2018 (32). The attention GAN model has two novel components to generate high resolution images, the attentional generative network and

deep attentional multi-modal similarity model. We generated sample images from this pre-trained model as shown in8, which is more refined and has higher resolution than cGAN model although the accuracy and the alignment of the images is questionable.

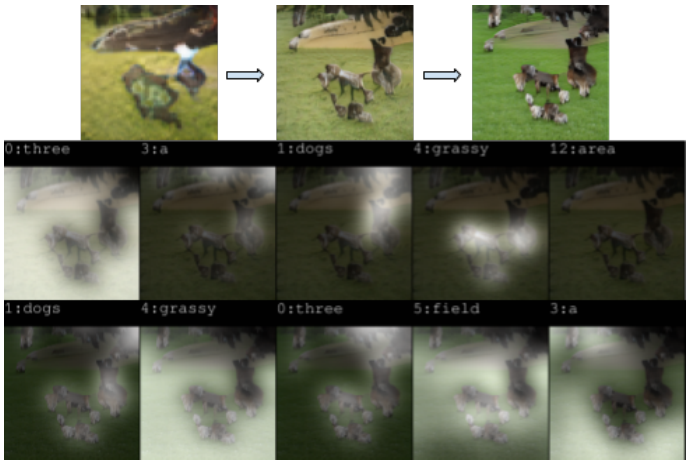

Figure 8: A sample image and corresponding attention maps generated by AttnGAN trained on MSCOCO when given the input "Three dogs on a grassy field in the middle of an urban area".

6.3 Stage 3

The image-caption model is trained on the two given datasets: Flickr8k and MSCOCO as two separate models and later on the synthetic dataset from previous stage.

We first train our model on Flickr8k/MSCOCO image-caption dataset for 10 epochs. The training loss for the MSCOCO dataset is shown in Figure 9. We use 1e-4 as the learning rate for encoder and 4e-4 learning rate for the decoder. It achieves a BLEU-4 score of 0.24 on the validation dataset which is close to the score other studies have achieved with similar computational power. The evolution of cross-entropy loss on the validation data set, BLEU-4 score and top-5 accuracy during the training on MSCOCO is provided in Figure 10.

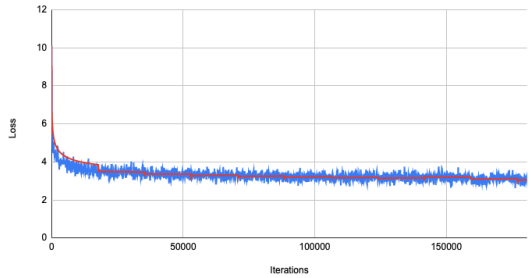

Figure 9: Cross-entropy loss during training on MSCOCO dataset.

Few sample captions generated by the two models, one trained on Flickr8k and another on MSCOCO, are given in Figure 11 along with the corresponding attention maps.

Then, we take the model trained on MSCOCO dataset and train it further on our synthetic image-caption dataset generated by Stage2GAN for 20 epochs. To compare the results, we also train it separately on dataset generated by AttnGAN for the same number of epochs. The validation loss and the BLEU-4 scores as this training progressed are plotted in Figure 12. While the loss during this training seemed to converge, the BLEU-4 scores dropped drastically from 0.24 to 0.04 in both cases and did not achieve the same level indicating the relatively poor quality of synthetic images and captions.

Beam Search: We also analyzed the effects of using beam search on the results on the model trained on the given dataset. The results for both Flickr8k and MSCOCO are compiled in Table 2. With

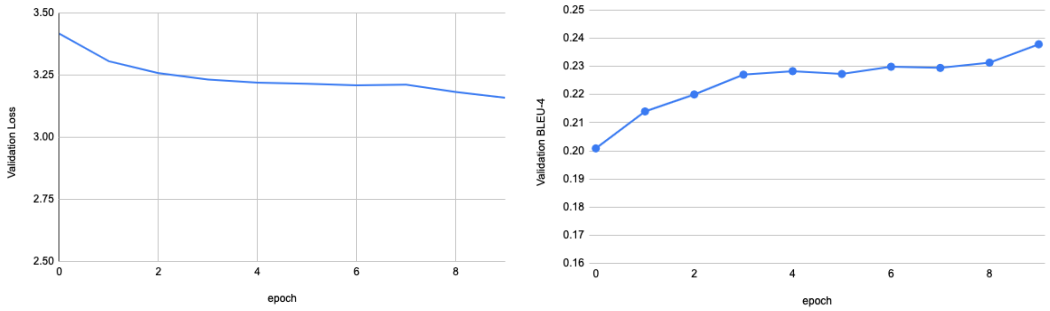

Figure 10: Cross-entropy loss and BLEU-4 score on the validation data set after each epoch of training on MSCOCO dataset.

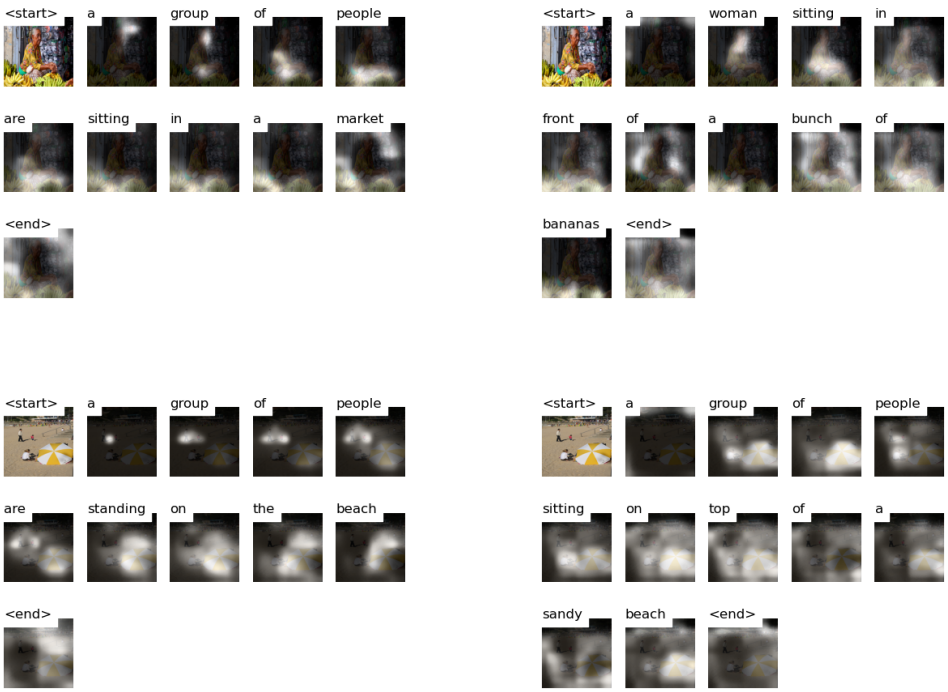

Figure 11: Captions and corresponding attention maps generated by Image-Captioning model at Stage 3 when trained on Flickr8k (left column) and MSCOCO (right column) dataset. The input image is the image under the <start> tag.

higher beam sizes, the model is able to consider more choices for each word in the caption which helps getting more optimal sequence.

7 Limitations of the framework

Though using this framework we can improve the performance of exiting models and given data, there are a few downsides too.

The image-to-caption generation procedure is based on maximum likelihood principle, as a result, the generated captions tend to follow patterns observed in the training data (Figure 13). So it lacks variability in the generated data. Objectives based on diversity and naturalness of the captions could be favored as proposed in study by Dai et al. 2017 (11).

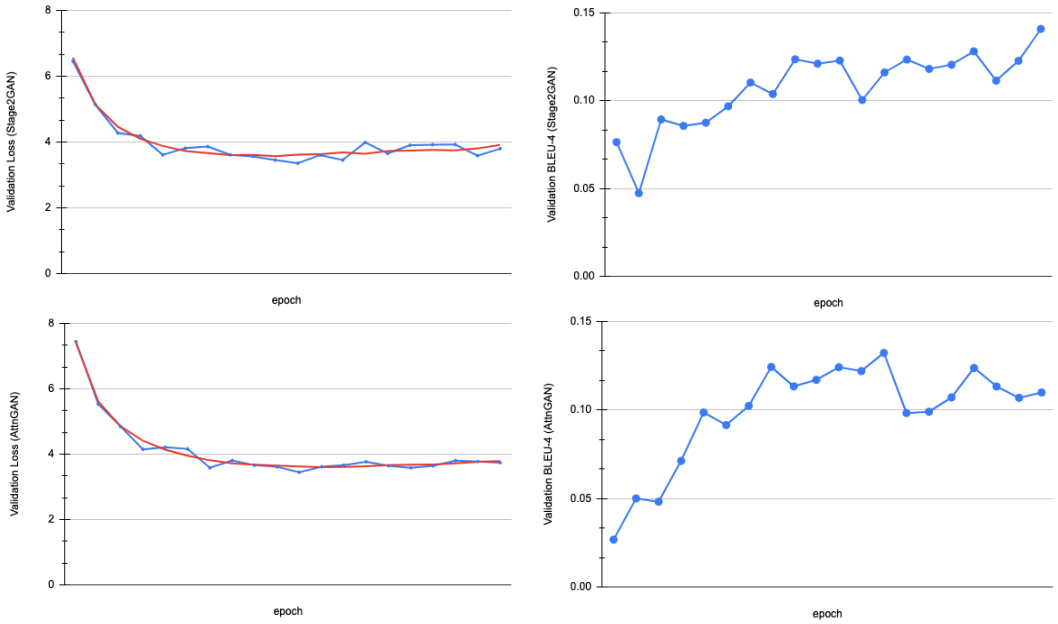

Figure 12: Once the image-caption model was trained on MSCOCO dataset and achieved a BLEU-4 score of 0.24, we started training it on synthetic data generated by our Stage2GAN (top row) and AttnGAN (bottom row). The plots above show cross-entropy loss and BLEU-4 score on the validation data set after each epoch of training.

Beam Size	Flickr8k	MSCOCO
1	0.2019	0.2667
3	0.2289	0.3089
5	0.2213	0.3120

Table 2: BLEU-4 scores on Flickr8k and MSCOCO test datasets for different beam sizes.

Our conditional GAN model is trained by creating representation vector of the complete sentence, which lacks fine-grained word-level information and prevents generating high quality images, as seen in the results in Figure 7.

Additionally, our sequential pipeline is not parallelizable, hence require a huge amount of time to train. A large portion of the time is spent on training generating synthetic data as well as training the image captioning model. Nevertheless, the low quality images can potentially improve the robustness of the image-captioning model.

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2016.

[2] M. Z. Hossain, F. Sohel, M. F. Shiratuddin, and H. Laga, "A comprehensive survey of deep learning for image captioning," *ACM Comput. Surv.*, vol. 51, feb 2019.

[3] 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," 2016.

[4] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195, 2017.

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, pp. 1–48, 2019.

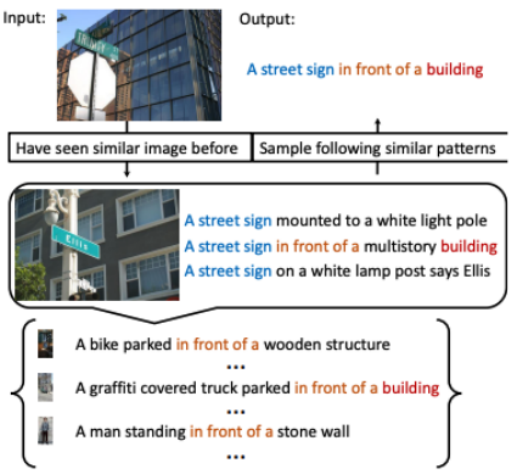

Figure 13: Figure taken from Dai et al. 2017 (11).

[6] A. Farhadi, M. Hejrati, M. Sadeghi, P. Young, C. Rashtchian, J. Hockenmaier, and D. Forsyth, "Every picture tells a story: Generating sentences from images," in *Computer Vision, ECCV 2010 - 11th European Conference on Computer Vision, Proceedings*, (Germany), pp. 15–29, Springer, 2010.

[7] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille, "Deep captioning with multimodal recurrent neural networks (m-rnn)," *arXiv preprint arXiv:1412.6632*, 2014.

[8] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," *CoRR*, vol. abs/1411.4555, 2014.

[9] C. Wang, H. Yang, C. Bartz, and C. Meinel, "Image captioning with deep bidirectional lstms," *CoRR*, vol. abs/1604.00790, 2016.

[10] J. Chen, H. Guo, K. Yi, B. Li, and M. Elhoseiny, "Visualgpt: Data-efficient image captioning by balancing visual input and linguistic knowledge from pretraining," *CoRR*, vol. abs/2102.10407, 2021.

[11] B. Dai, S. Fidler, R. Urtasun, and D. Lin, "Towards diverse and natural image descriptions via a conditional gan," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[12] C. Chen, S. Mu, W. Xiao, Z. Ye, L. Wu, and Q. Ju, "Improving image captioning with conditional generative adversarial nets," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, p. 8142–8150, Jul 2019.

[13] M. Hittmeir, A. Ekelhart, and R. Mayer, "On the utility of synthetic data: An empirical evaluation on machine learning tasks," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[14] S. I. Nikolenko, "Synthetic data for deep learning," *CoRR*, vol. abs/1909.11512, 2019.

[15] W. Qiu and A. L. Yuille, "Unrealcv: Connecting computer vision to unreal engine," *CoRR*, vol. abs/1609.01326, 2016.

[16] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," 2017.

[17] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014.

[18] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[19] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International conference on machine learning*, pp. 1060–1069, PMLR, 2016.

[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[21] Y. Koizumi, Y. Ohishi, D. Niizumi, D. Takeuchi, and M. Yasuda, "Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval," 2020.

[22] J. Huang, L. Murn, M. Mrak, and M. Worring, "GPT2MVS: generative pre-trained transformer-2 for multi-modal video summarization," *CoRR*, vol. abs/2104.12465, 2021.

[23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[24] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[25] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, pp. 4780–4789, 2019.

[26] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[27] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[28] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 49–58, 2016.

[29] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 2048–2057, PMLR, 07–09 Jul 2015.

[30] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.

[31] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), (Cham), pp. 740–755, Springer International Publishing, 2014.

[32] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," pp. 1316–1324, 06 2018.

Appendices

A Optimization Equation derivation

$$\begin{aligned}
&\min_{A,B} \quad L\left(D_{\text{val}}; W^*\left(G^*(A), E^*(B)\right)\right) \\
&\text{s.t} \quad\quad W^*\left(G^*(A), E^*(B)\right) = \operatorname{argmin}_W L\left(\{(x_i, y_i)\}_{i=1}^N; W\right) + \lambda L\left(S\left(G^*(A), E^*(B)\right); W\right) \\
&\qquad\quad E^*(B) = \operatorname{argmin}_F \operatorname{argmax}_E L\left(\{(x_i, y_i)\}_{i=1}^N; E, F, B\right) \\
&\qquad\quad G^*(A) = \operatorname{argmin}_G \sum_{i=1}^{N_i} a_i l\left(y_i; G\right)
\end{aligned} \tag{11}$$

One step gradient update of $E\left(B\right), F, G\left(A\right)$, and W.

$$\begin{aligned}
E^{'} &= E + \eta_E \nabla_E L\left(\{(x_i, y_i)\}_{i=1}^N; E, F, B\right) \\
F^{'} &= F - \eta_F \nabla_F L\left(\{(x_i, y_i)\}_{i=1}^N; E, F, B\right)
\end{aligned} \tag{12}$$

$$G^{'} = G - \eta_g \nabla_G \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i G\}\right)\right] \tag{13}$$

$$W^{'} = W - \eta_W \nabla_W \left(L\left(\{x_i, y_i\}_{i=1}^N; W\right)\right) - \eta_W \lambda \nabla_W L\left(S\left(G^*\left(A\right), E^*\left(B\right)\right), W\right) \tag{14}$$

$$\min_{A,B} L(D_{val}; W^*\left(G^*\left(A\right), E^*\left(B\right)\right))$$

Part 1. minimize w.r.t A and replacing $W^*\left(G^*\left(A\right), E^*\left(B\right)\right)$ with equation 14.

$$\nabla_A L\left(D_{val}; W - \eta_W \nabla_W \left(L\left(\{x_i, y_i\}_{i=1}^N; W\right)\right) - \eta_W \lambda \nabla_W L\left(S\left(G^*\left(A\right), E^*\left(B\right)\right), W\right)\right) \tag{15}$$

$$= -\eta_W \lambda \left(\nabla^2_{A,W} L\left(S\left(G^*\left(A\right), E^*\left(B\right)\right), W\right)\right)^* \nabla_{W'} L\left(D_{val}; W'\right) \tag{16}$$

$$\nabla^2_{A,W} L\left(S\left(G^*\left(A\right), E^*\left(B\right)\right), W\right) = \nabla_A \left(G^{'}\left(A\right)\right)^* \nabla^2_{G',W} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right), W\right)) \tag{17}$$

Now expression 16 becomes:

$$-\eta_W \lambda \left(\nabla_A \left(G^{'}\left(A\right)\right)^* \nabla^2_{G',W} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right), W\right))\right)^* \nabla_{W'} L\left(D_{val}; W'\right) \tag{18}$$

Using Finite difference method to approximate $\nabla^2_{G',W} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right), W\right))^* \nabla_{W'} L\left(D_{val}; W'\right)$ expression

$$\frac{1}{2\alpha_g}\left(\nabla_{G'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^+\right) - \nabla_{G'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^-\right)\right) \tag{19}$$

$$W^{\pm} = W \pm \alpha_g \nabla_{W'} L\left(D_{val}; W^{'}\right)$$

$$\alpha_g = \frac{0.01}{|\nabla_{W'} L\left(D, W'\right)|_2}$$

$$\nabla_A G^{'}(A) = \nabla_A \left(G - \eta_g \nabla_G \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i G\}\right) \right] \right)$$
$$= -\eta_g \nabla^2_{G,A} \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i G\}\right) \right] \tag{20}$$

Now Equation 18:

$$\frac{-1}{2\alpha_g} \xi_g \nabla^2_{G,A} \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i G\}\right) \right] * \left(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{+} \right) - \nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{-} \right) \right) \tag{21}$$

Simplifying $\nabla^2_{G,A} \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G\}\right) \right] * \left(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{+} \right) \right)$ equation using finite difference method

$$\frac{1}{2\alpha_a} \left(\nabla_A \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G^{+}\}\right) \right] - \nabla_A \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G^{-}\}\right) \right] \right) \tag{22}$$
$$G^{\pm} = G \pm \alpha_a * \left(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{+} \right) \right)$$
$$\alpha_a = \frac{0.01}{|(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B)\right); W^{+}\right))|_2}$$

Similarly simplifying $\nabla^2_{G,A} \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G\}\right) \right] * \left(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{-} \right) \right)$ equation using finite difference method.

$$\frac{1}{2\alpha_a} \left(\nabla_A \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G^{+}\}\right) \right] - \nabla_A \left[\sum_{i=1}^{N} \alpha_i l\left(\{y_i, G^{-}\}\right) \right] \right) \tag{23}$$
$$G^{\pm} = G \pm \alpha_a * \left(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{-} \right) \right)$$
$$\alpha_a = \frac{0.01}{|(\nabla_{G'} L\left(S\left(G^{'}(A), E^{'}(B)\right); W^{-}\right))|_2}$$

$$\min_{A,B} L(D_{val}; W^{*}(G^{*}(A), E^{*}(B)))$$

Part 2. minimize w.r.t B and replacing $W^{*}(G^{*}(A), E^{*}(B))$ with equation 14.

$$\nabla_B L\left(D_{val}; W - \xi_W \nabla_W \left(L\left(\{x_i, y_i\}_{i=1}^{N}; W \right) \right) \right) - \xi_w \lambda \nabla_W L\left(S\left(G^{*}(A), E^{*}(B) \right), W \right)) \tag{24}$$

$$-\xi_w \lambda \left(\nabla^2_{B,W} L\left(S\left(G^{*}(A), E^{*}(B) \right), W \right) \right)^{*} \nabla_{W'} L\left(D_{val}; W' \right) \tag{25}$$

$$\nabla^2_{B,W} L\left(S\left(G^{*}(A), E^{*}(B) \right), W \right)) = \nabla_B \left(E^{'}(B) \right)^{*} \nabla^2_{E',W} L\left(S\left(G^{'}(A), E^{'}(B) \right), W \right)) \tag{26}$$

Now equation 25 becomes:

$$-\xi_w \lambda \left(\nabla_B \left(E^{'}(B) \right)^{*} \nabla^2_{E',W} L\left(S\left(G^{'}(A), E^{'}(B) \right), W \right)) \right)^{*} \nabla_{W'} L\left(D_{val}; W' \right) \tag{27}$$

Using Finite difference method to approximate $\nabla^2_{E',W} L\left(S\left(G^{'}(A), E^{'}(B) \right), W \right)) * \nabla_{W'} L\left(D_{val}; W' \right)$ equation

$$\frac{1}{2\alpha_e} \left(\nabla_{E'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{+} \right) - \nabla_{E'} L\left(S\left(G^{'}(A), E^{'}(B) \right); W^{-} \right) \right) \tag{28}$$

$$W^{\pm} = W \pm \alpha_e \nabla_{W'} L\left(D_{val}; W^{'}\right)$$
$$\alpha_e = \frac{0.01}{|\nabla_{W'} L\left(D, W'\right)|_2}$$

$$\begin{aligned}\nabla_B\left(E^{'}\left(B\right)\right) &= \nabla_B\left(E - \xi_E \nabla_F L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right) + \xi_E \nabla_E L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right)\right) \\ &= \xi_E\left(-\nabla^2_{F,B} L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right) + \nabla^2_{E,B} L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right)\right)\end{aligned} \tag{29}$$

Now Equation 25 is:

$$\begin{aligned}&\frac{1}{2\alpha_e}\xi_E\left(\nabla^2_{E,B} L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right)\right) * \\ &\left(\nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{+}\right) - \nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{-}\right)\right)\end{aligned} \tag{30}$$

Simplifying $\nabla^2_{E,B} L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right) * \nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{+}\right)$ equation using finite difference method

$$\frac{1}{2\alpha_b}\left(\nabla_B L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E^{+}, F, B\right) - \nabla_B L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E^{-}, F, B\right)\right) \tag{31}$$
$$E^{\pm} = E \pm \alpha_b * \left(\nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{+}\right)\right)$$
$$\alpha_b = \frac{0.01}{|(\nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{+}\right))|_2}$$

Simplifying $\nabla^2_{E,B} L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E, F, B\right) * \nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{-}\right)$ equation using finite difference method

$$\frac{1}{2\alpha_b}\left(\nabla_B L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E^{+}, F, B\right) - \nabla_B L\left(\left\{\left(x_i, y_i\right)\right\}_{i=1}^{N}; E^{-}, F, B\right)\right) \tag{32}$$
$$E^{\pm} = E \pm \alpha_b * \left(\nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{-}\right)\right)$$
$$\alpha_b = \frac{0.01}{|(\nabla_{E'} L\left(S\left(G^{'}\left(A\right), E^{'}\left(B\right)\right); W^{-}\right))|_2}$$