Visual Program Synthesis

Divyat Mahajan
divyatm@iitk.ac.in

Hritvik Taneja
hritvikt@iitk.ac.in

Indian Institute of Technology Kanpur
Mentor: Prof. Vinay Namboodiri

November 22, 2018

Outline

Program Synthesis

- Program Synthesis is the process of automatically translating the given input specification into a program

Program Synthesis

- Program Synthesis is the process of automatically translating the given input specification into a program
- There are several ways to get around solving this problem of program generation:

Program Synthesis

- Program Synthesis is the process of automatically translating the given input specification into a program
- There are several ways to get around solving this problem of program generation:
  - **Traditional Rule based:** Translate using pre defined rule. Such approaches mostly require a formal, precise specification that can be hard to obtain and also cannot perform well under noisy input

Program Synthesis

- Program Synthesis is the process of automatically translating the given input specification into a program
- There are several ways to get around solving this problem of program generation:
  - **Traditional Rule based:** Translate using pre defined rule. Such approaches mostly require a formal, precise specification that can be hard to obtain and also cannot perform well under noisy input
  - **Neural Program Synthesis:** Model generates a program conditioned on the input specification, usually in the form of Input Output pairs

Program Synthesis

- Program Synthesis is the process of automatically translating the given input specification into a program
- There are several ways to get around solving this problem of program generation:
  - **Traditional Rule based:** Translate using pre defined rule. Such approaches mostly require a formal, precise specification that can be hard to obtain and also cannot perform well under noisy input
  - **Neural Program Synthesis:** Model generates a program conditioned on the input specification, usually in the form of Input Output pairs
  - **Neural Program Induction:** Models learns a latent representation of the program and generates output directly as per the input specifications

Problem Statement

- Aim is to learn a model that generates logo program which achieves the task of creating output(images) as specified in the given input text caption

Problem Statement

- Aim is to learn a model that generates logo program which achieves the task of creating output(images) as specified in the given input text caption
- It involves learning a program representation under the Neural Program Synthesis approach, it can also be interpreted as converting pseduo code(input text caption) to logo code that can be compiled to generate the required output

Problem Statement

- Aim is to learn a model that generates logo program which achieves the task of creating output(images) as specified in the given input text caption
- It involves learning a program representation under the Neural Program Synthesis approach, it can also be interpreted as converting pseduo code(input text caption) to logo code that can be compiled to generate the required output
- Ex: Given the query "This is an rectangle of side 10 and 20.", the model should output the logo code `repeat 2 [fd 10 lt 90 fd 20 lt 90]`

Problem Statement

- Aim is to learn a model that generates logo program which achieves the task of creating output(images) as specified in the given input text caption
- It involves learning a program representation under the Neural Program Synthesis approach, it can also be interpreted as converting pseduo code(input text caption) to logo code that can be compiled to generate the required output
- Ex: Given the query "This is an rectangle of side 10 and 20.", the model should output the logo code `repeat 2 [fd 10 lt 90 fd 20 lt 90]`
- We propose a generative model using Conditional GAN and attentional LSTM that learns to generate Logo code without explicitly depending on the input caption embedding

Generative Adversarial Networks (GANs)

- Generative Adversarial Network (GANs) are implicit generative models trained under the adversarial setting

Generative Adversarial Networks (GANs)

- Generative Adversarial Network (GANs) are implicit generative models trained under the adversarial setting
- Generator learns a distribution over data $p_g(x|z, \theta_g)$ and Discriminator learns the distribution $p_d(x|\theta_d)$ to predict the probability of the data x coming the real data distribution

Generative Adversarial Networks (GANs)

- Generative Adversarial Network (GANs) are implicit generative models trained under the adversarial setting
- Generator learns a distribution over data $p_g(x|z, \theta_g)$ and Discriminator learns the distribution $p_d(x|\theta_d)$ to predict the probability of the data x coming the real data distribution
- The framework is trained such that the Discriminator learns to differentiate between the actual data and the data samples from Generator's distribution while simultaneously Generator learns to generate samples that would be be hard for the Discriminator to differentiate from real data

Generative Adversarial Networks (GANs)

- Generative Adversarial Network (GANs) are implicit generative models trained under the adversarial setting
- Generator learns a distribution over data $p_g(x|z, \theta_g)$ and Discriminator learns the distribution $p_d(x|\theta_d)$ to predict the probability of the data x coming the real data distribution
- The framework is trained such that the Discriminator learns to differentiate between the actual data and the data samples from Generator's distribution while simultaneously Generator learns to generate samples that would be be hard for the Discriminator to differentiate from real data
- The training procedure can be stated formally as follows:

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

Generative Adversarial Text to Image Synthesis

- A model proposed by Reed et al. for automatic synthesis of images from their textual description

Generative Adversarial Text to Image Synthesis

- A model proposed by Reed et al. for automatic synthesis of images from their textual description
- The framework consists of GAN with the Generator conditioned on input text caption to generate images and Discriminator architecture modified to incorporate the text caption embedding

Generative Adversarial Text to Image Synthesis

- A model proposed by Reed et al. for automatic synthesis of images from their textual description
- The framework consists of GAN with the Generator conditioned on input text caption to generate images and Discriminator architecture modified to incorporate the text caption embedding
- It incorporates additional loss term over the loss function in GAN to make the fake images sampled from Generator correspond to the caption of the image as well

Generative Adversarial Text to Image Synthesis

- A model proposed by Reed et al. for automatic synthesis of images from their textual description
- The framework consists of GAN with the Generator conditioned on input text caption to generate images and Discriminator architecture modified to incorporate the text caption embedding
- It incorporates additional loss term over the loss function in GAN to make the fake images sampled from Generator correspond to the caption of the image as well
- The additional loss term corresponds to the real image with incorrect caption that the Discriminator is trained to predict as fake class

Generative Adversarial Text to Image Synthesis

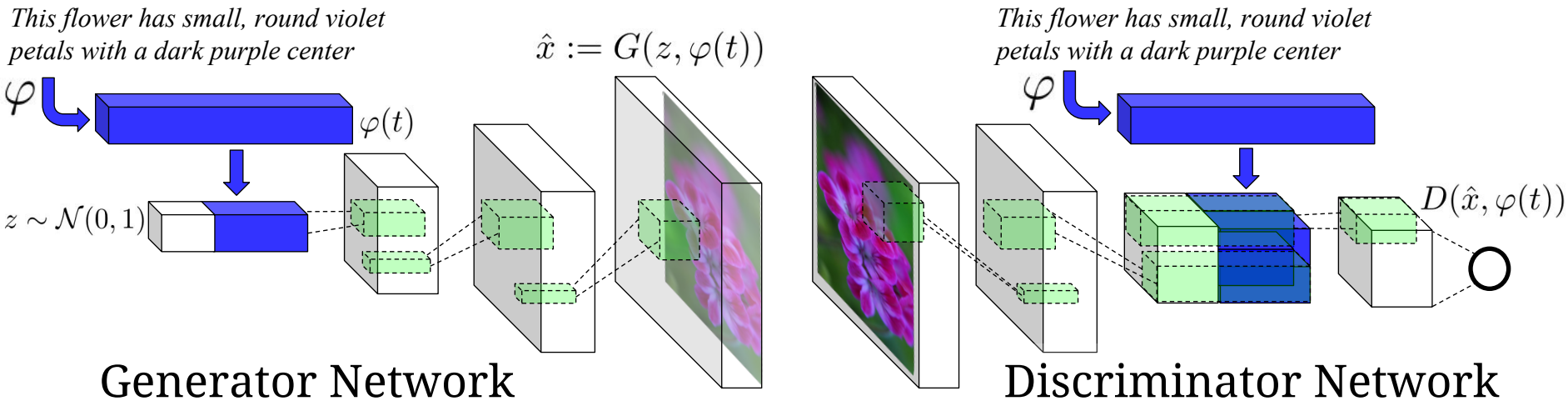
- A model proposed by Reed et al. for automatic synthesis of images from their textual description
- The framework consists of GAN with the Generator conditioned on input text caption to generate images and Discriminator architecture modified to incorporate the text caption embedding
- It incorporates additional loss term over the loss function in GAN to make the fake images sampled from Generator correspond to the caption of the image as well
- The additional loss term corresponds to the real image with incorrect caption that the Discriminator is trained to predict as fake class
- Loss function:

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

where h corresponds to the correct caption embedding corresponding the real image x and the

$\hat{h}$

Generative Adversarial Text to Image Synthesis



Show Attend Tell

- An attention based LSTM framework to generate captions for an image

Show Attend Tell

- An attention based LSTM framework to generate captions for an image
- It consists of a set of annotation feature vectors $\{a_i\}$ constructed from the image using CNN and an attention framework f_{att} that predicts the weight α_i for each annotation feature a_i. This is used compute the context vector $\hat{z}_t$

$$e_{ti} = f_{att}(a_i, h_{t-1})$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})}$$

$$\hat{z}_t = \psi(\{a_i\}, \{\alpha_i\})$$

Show Attend Tell

- An attention based LSTM framework to generate captions for an image
- It consists of a set of annotation feature vectors $\{a_i\}$ constructed from the image using CNN and an attention framework f_{att} that predicts the weight α_i for each annotation feature a_i. This is used compute the context vector $\hat{z}_t$

$$e_{ti} = f_{att}(a_i, h_{t-1})$$

$$\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{L} exp(e_{tk})}$$

$$\hat{z}_t = \psi(\{a_i\}, \{\alpha_i\})$$

- The CNN architecture for constructing the annotation vector $\{a_i\}$ serves as the Encoder and the Decoder LSTM uses attention framework to determine which annotation vectors to focus more for computing the hidden states at a particular time step

Show Attend Tell

- The hidden state and memory state of the LSTM are initialised by feeding the average of annotation vectors into Multi Layer Perceptron layers $f_{init,c}$ and $f_{init,h}$
  $c_o = f_{init,c}(\sum_i^L a_i/L)$
  $h_o = f_{init,h}(\sum_i^L a_i/L)$

Show Attend Tell

- The hidden state and memory state of the LSTM are initialised by feeding the average of annotation vectors into Multi Layer Perceptron layers $f_{init,c}$ and $f_{init,h}$
  $c_o = f_{init,c}(\sum_i^L a_i/L)$
  $h_o = f_{init,h}(\sum_i^L a_i/L)$
- The updates for the hidden states and memory states of LSTM for the further time step depend on the previous state and the context vector, defined below:

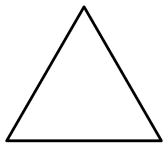
$$\begin{pmatrix} \boldsymbol{i}_t \\ \boldsymbol{f}_t \\ \boldsymbol{o}_t \\ \boldsymbol{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} T_{D+m+n,n} \begin{pmatrix} \boldsymbol{E}\boldsymbol{y}_{t-1} \\ \boldsymbol{h}_{t-1} \\ \hat{\boldsymbol{z}}_t \end{pmatrix}$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

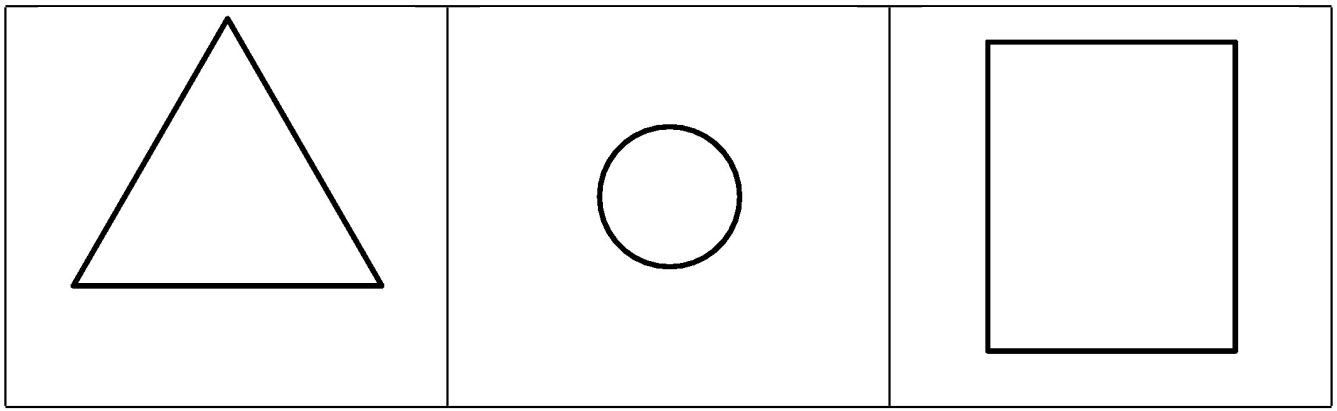
$$h_t = o_t * tanh(c_t)$$

Dataset

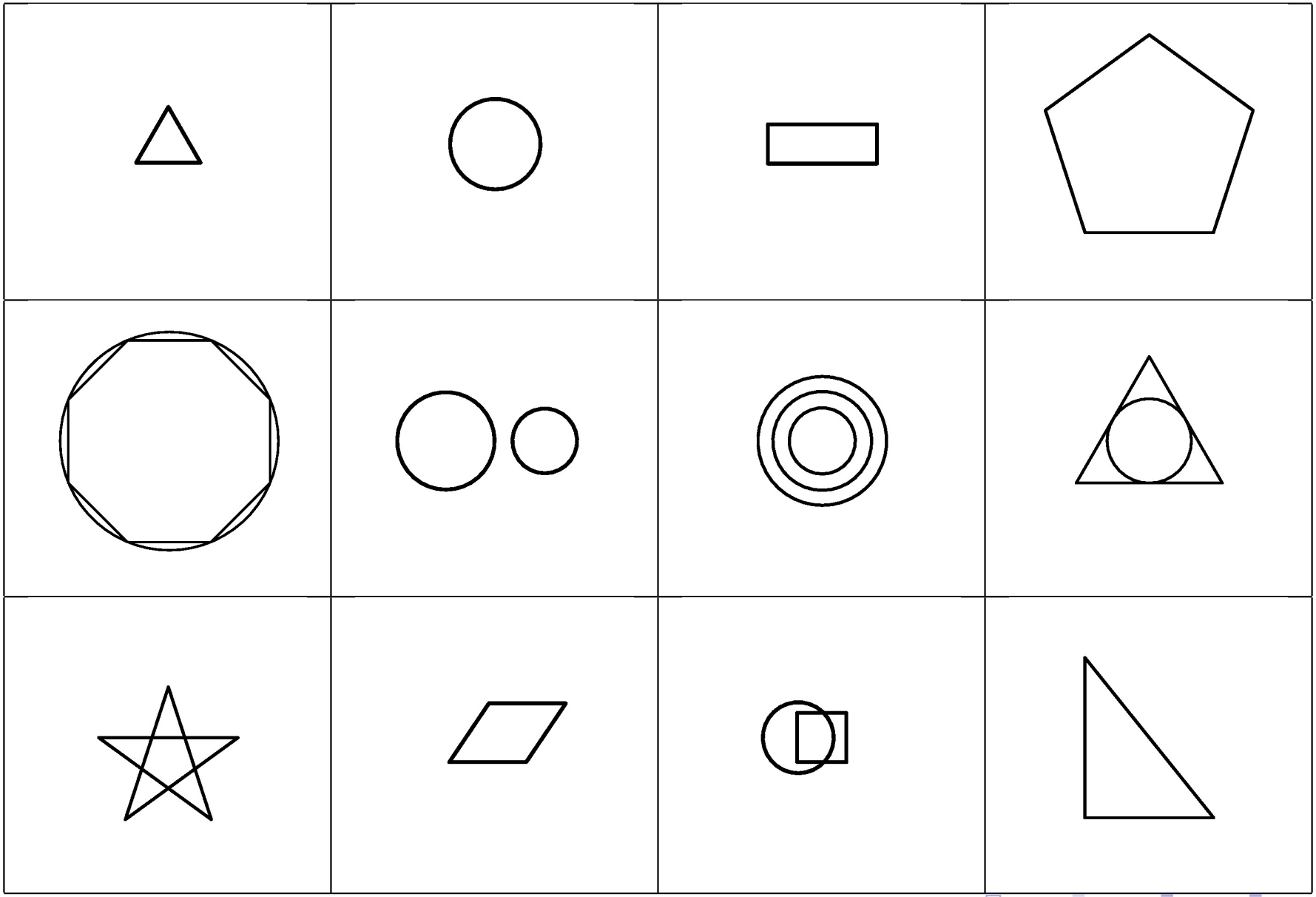
- Due to no standard datasets available for testing, we created two datasets Basic and Complex to test the task of generating Logo Code that create geometrical shapes
- Basic dataset has 10,000 images with 3 different categories
- Complex dataset has 20,000 images with 12 different categories
- There is a description and logo code corresponding to each image present in the dataset.

Description	Logo Code	Image
This is an equilateral triangle of side length: 656.	`repeat 3 [fd 656 lt 120]`	

Basic Dataset



Complex Dataset



Architecture

- We use a conditional DC-GAN architecture as proposed by Reed et al. to generate images (conditioned on text caption). The text caption is encoded using skipthoughts and then appended with the noise vector before feeding into the Generator

Architecture

- We use a conditional DC-GAN architecture as proposed by Reed et al. to generate images (conditioned on text caption). The text caption is encoded using skipthoughts and then appended with the noise vector before feeding into the Generator
- We augment the text to image architecture by adding $\mathcal{L}1$ loss term between the encoding of the real and fake image. This encoding is extracted out from the penultimate layer of discriminator.

Architecture

- We use a conditional DC-GAN architecture as proposed by Reed et al. to generate images (conditioned on text caption). The text caption is encoded using skipthoughts and then appended with the noise vector before feeding into the Generator
- We augment the text to image architecture by adding $\mathcal{L}1$ loss term between the encoding of the real and fake image. This encoding is extracted out from the penultimate layer of discriminator.
- We use a Decoder LSTM network to synthesize Logo code with attention framework similar to the paper Show Attend Tell.

Architecture

- We use a conditional DC-GAN architecture as proposed by Reed et al. to generate images (conditioned on text caption). The text caption is encoded using skipthoughts and then appended with the noise vector before feeding into the Generator
- We augment the text to image architecture by adding $\mathcal{L}1$ loss term between the encoding of the real and fake image. This encoding is extracted out from the penultimate layer of discriminator.
- We use a Decoder LSTM network to synthesize Logo code with attention framework similar to the paper Show Attend Tell.
- The annotation vectors comprises of features constructed using VGG network on image alongwith the encoding extracted out of the penultimate layer of the discriminator

Architecture

- We use a conditional DC-GAN architecture as proposed by Reed et al. to generate images (conditioned on text caption). The text caption is encoded using skipthoughts and then appended with the noise vector before feeding into the Generator
- We augment the text to image architecture by adding $\mathcal{L}1$ loss term between the encoding of the real and fake image. This encoding is extracted out from the penultimate layer of discriminator.
- We use a Decoder LSTM network to synthesize Logo code with attention framework similar to the paper Show Attend Tell.
- The annotation vectors comprises of features constructed using VGG network on image alongwith the encoding extracted out of the penultimate layer of the discriminator
- We can also incorporate additional annotation vector as the caption skipthoughts embedding

Architecture

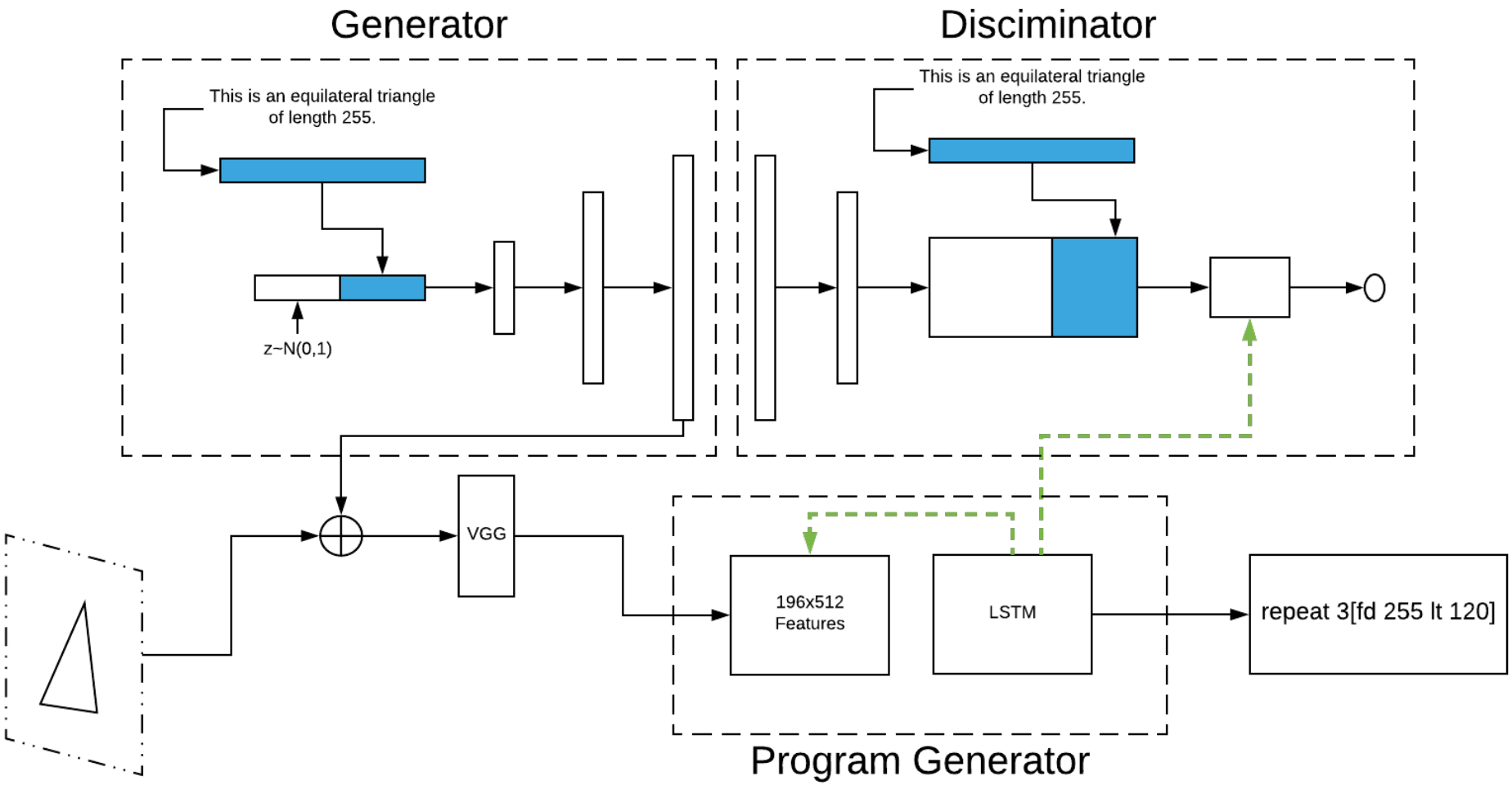


Figure: Overall Architecture without caption attention

Algorithm I

Algorithm 1: Model

Input: minibatch images x, matching code c, matching description t, different image x_1, number of training batch steps N

for $n = 1$ **to** N **do**

$h \leftarrow \phi(t)$ ▷ Encode matching text description

$z \leftarrow \mathcal{N}(0,1)^Z$ ▷ Draw sample of random noise

$\hat{x} \leftarrow G(z, h)$ ▷ Forward through generator

$s_r, pl_r \leftarrow D(x, h)$ ▷ real image, right description

$s_w, pl_w \leftarrow D(x_1, h)$ ▷ mis-matching image and description

$s_f, pl_f \leftarrow D(\hat{x}, h)$ ▷ fake image, right description

$c_r \leftarrow PG(x, pl_r)$ ▷ Forward through program generator, real image

$c_f \leftarrow PG(\hat{x}, pl_f)$ ▷ Forward through program generator, fake image

Algorithm II

$\mathcal{L}_E \leftarrow |pl_r - pl_f|$ ▷ L1 loss

$\mathcal{L}_D \leftarrow \log(s_r) + \frac{\log(1-s_w)+\log(1-s_f)}{2} + \mathcal{L}_E$

$\mathcal{L}_G \leftarrow \log(s_f)$

$\mathcal{L}_R \leftarrow seq2seq\_loss(c, c_r) + seq2seq\_loss(c, c_f)$ ▷ LSTM Loss

$D \leftarrow D - \frac{\partial \mathcal{L}_D}{\partial D}$

$G \leftarrow G - \frac{\partial \mathcal{L}_G}{\partial G}$

$R \leftarrow R - \frac{\partial \mathcal{L}_R}{\partial R}$

end for

Experiment Design

- The end to end training of the model is tricky and performing well on both the tasks of generating good synthesized images and good synthesized logo program is challenging
- After a lot of tuning and traning methodologies for LSTM we arrive at the following method to generate good resultls
- We add teacher forcing for program generation from both fake and real image during the training phase of the model to facilitate LSTM in tuning its parameters. Hence, the LSTM is updated on the loss contribution from Fake Image, Real Image, Fake Image Teacher Forcing and Real Image Teacher Forcing
- We learn the parameters by first updating the parameters of LSTM and Discriminator together and then updating the parameters of Generator
- The test dataset comprises of 1000 data points with the same structure and specifications as the train dataset: (input caption, image, logo code) triplet

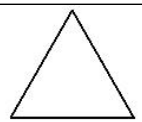
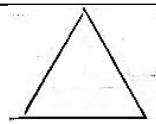
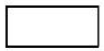
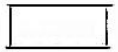
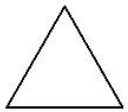
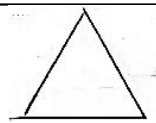
Basic Dataset with caption attention, Real Image while Training

Actual Code	repeat 3 [fd 375 lt 120]	repeat 2 [fd 586 lt 90 fd 524 lt 90]
Epoch 0	repeat 3 [fd 4 100	repeat [fd 5 9
Epoch 15	repeat 3 [fd 382 lt 120]	repeat 2 [fd 580 lt 90 fd 492 lt 90]
Epoch 30	repeat 3 [fd 362 lt 120]	repeat 2 [fd 585 lt 90 fd 522 lt 90]
Epoch 45	repeat 3 [fd 303 lt 120]	repeat 2 [fd 583 lt 90 fd 520 lt 90]
Epoch 60	repeat 3 [fd 301 lt 120]	repeat 2 [fd 500 lt 90 fd 500 lt 90]
Epoch 75	repeat 3 [fd 370 lt 120]	repeat 2 [fd 590 lt 90 fd 500 lt 90]
Epoch 90	repeat 3 [fd 372 lt 120]	repeat 2 [fd 583 lt 90 fd 520 lt 90]
Epoch 105	repeat 3 [fd 370 lt 120]	repeat 2 [fd 553 lt 90 fd 582 lt 90]
Epoch 120	repeat 3 [fd 370 lt 120]	repeat 2 [fd 580 lt 90 fd 500 lt 90]
Epoch 135	repeat 3 [fd 370 lt 120]	repeat 2 [fd 580 lt 90 fd 525 lt 90]
Epoch 150	repeat 3 [fd 370 lt 120]	repeat 2 [fd 581 lt 90 fd 511 lt 90]
Epoch 165	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 523 lt 90]
Epoch 180	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 500 lt 90]
Epoch 195	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 520 lt 90]
Epoch 210	repeat 3 [fd 375 lt 120]	repeat 2 [fd 585 lt 90 fd 526 lt 90]
Epoch 225	repeat 3 [fd 375 lt 120]	repeat 2 [fd 584 lt 90 fd 522 lt 90]
Epoch 240	repeat 3 [fd 375 lt 120]	repeat 2 [fd 581 lt 90 fd 523 lt 90]
Epoch 255	repeat 3 [fd 375 lt 120]	repeat 2 [fd 583 lt 90 fd 523 lt 90]
Epoch 270	repeat 3 [fd 374 lt 120]	repeat 2 [fd 599 lt 90 fd 512 lt 90]
Epoch 285	repeat 3 [fd 375 lt 120]	repeat 2 [fd 583 lt 90 fd 523 lt 90]

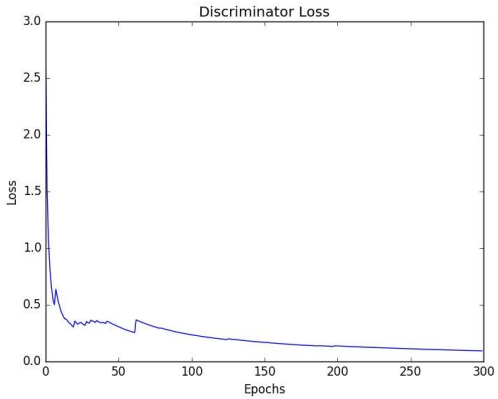
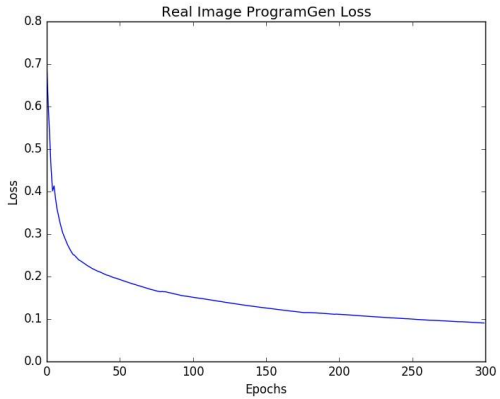
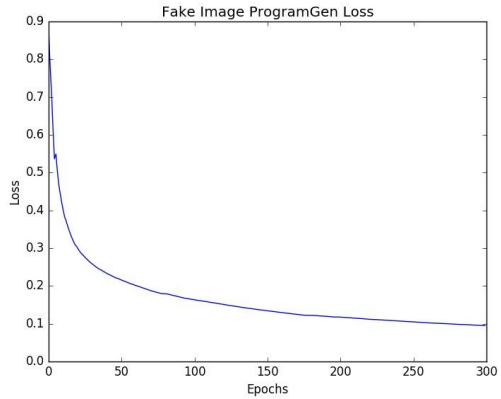
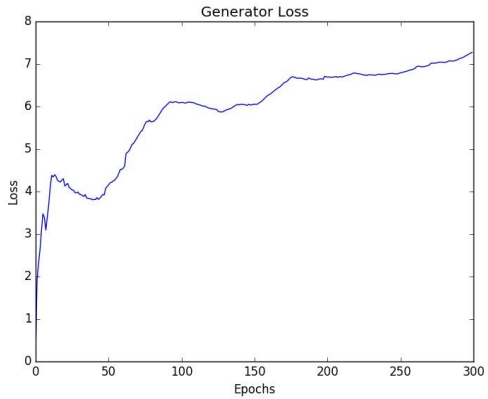
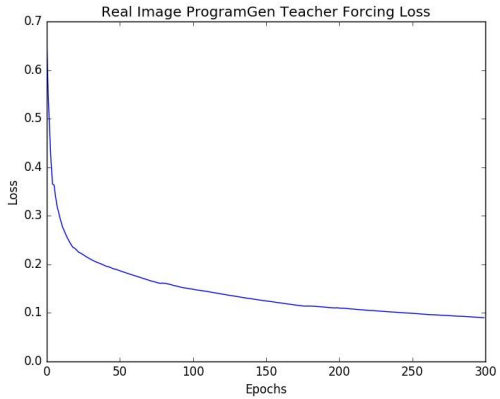
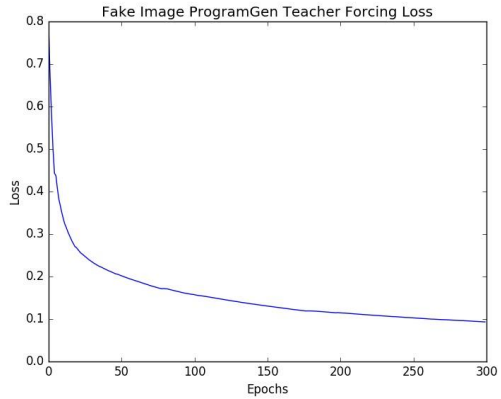
Basic Dataset with caption attention, Fake Image while Training

Actual Code	repeat 3 [fd 375 lt 120]	repeat 2 [fd 586 lt 90 fd 524 lt 90]
Epoch 0	repeat 3 [fd 4 200	repeat 2 [fd 5
Epoch 15	repeat 3 [fd 319 lt 120]	repeat 2 [fd 580 lt 90 fd 492 lt 90]
Epoch 30	repeat 3 [fd 362 lt 120]	repeat 2 [fd 585 lt 90 fd 522 lt 90]
Epoch 45	repeat 3 [fd 303 lt 120]	repeat 2 [fd 581 lt 90 fd 520 lt 90]
Epoch 60	repeat 3 [fd 301 lt 120]	repeat 2 [fd 500 lt 90 fd 500 lt 90]
Epoch 75	repeat 3 [fd 370 lt 120]	repeat 2 [fd 590 lt 90 fd 500 lt 90]
Epoch 90	repeat 3 [fd 372 lt 120]	repeat 2 [fd 583 lt 90 fd 520 lt 90]
Epoch 105	repeat 3 [fd 370 lt 120]	repeat 2 [fd 553 lt 90 fd 582 lt 90]
Epoch 120	repeat 3 [fd 370 lt 120]	repeat 2 [fd 580 lt 90 fd 500 lt 90]
Epoch 135	repeat 3 [fd 370 lt 120]	repeat 2 [fd 580 lt 90 fd 525 lt 90]
Epoch 150	repeat 3 [fd 370 lt 120]	repeat 2 [fd 581 lt 90 fd 511 lt 90]
Epoch 165	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 523 lt 90]
Epoch 180	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 500 lt 90]
Epoch 195	repeat 3 [fd 375 lt 120]	repeat 2 [fd 580 lt 90 fd 520 lt 90]
Epoch 210	repeat 3 [fd 375 lt 120]	repeat 2 [fd 585 lt 90 fd 526 lt 90]
Epoch 225	repeat 3 [fd 375 lt 120]	repeat 2 [fd 584 lt 90 fd 522 lt 90]
Epoch 240	repeat 3 [fd 375 lt 120]	repeat 2 [fd 583 lt 90 fd 523 lt 90]
Epoch 255	repeat 3 [fd 375 lt 120]	repeat 2 [fd 583 lt 90 fd 523 lt 90]
Epoch 270	repeat 3 [fd 374 lt 120]	repeat 2 [fd 599 lt 90 fd 512 lt 90]
Epoch 285	repeat 3 [fd 375 lt 120]	repeat 2 [fd 583 lt 90 fd 523 lt 90]

Basic Dataset with caption attention, while Testing

Real Image	Fake Image	Real Code	Fake Code
		repeat 3 [fd 653 lt 120]	repeat 3 [fd 653 lt 120]
		repeat 2 [fd 487 lt 90 fd 213 lt 90]	repeat 2 [fd 485 lt 90 fd 222 lt 90]
		repeat 2 [fd 210 lt 90 fd 461 lt 90]	repeat 2 [fd 210 lt 90 fd 461 lt 90]
		circle 161	circle 161
		repeat 3 [fd 616 lt 120]	repeat 3 [fd 612 lt 120]

Basic Dataset with caption attention Plots

Discriminator Loss	Real Image ProgramGen Loss	Fake Image ProgramGen Loss
		
Generator Loss	**Real Image ProgramGen Teacher Forcing Loss**	**Fake Image ProgramGen Teacher Forcing Loss**
		

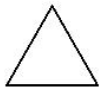
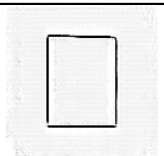
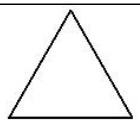
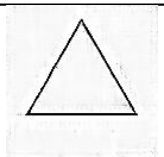
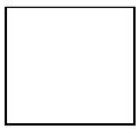
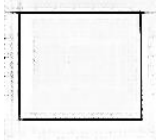
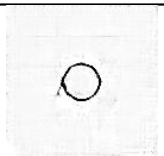
Basic Dataset without caption attention, Real Image while Training

Actual Code	repeat 2 [fd 595 lt 90 fd 371 lt 90]	repeat 2 [fd 586 lt 90 fd 524 lt 90]
Epoch 0	repeat 2 [fd 9	repeat 3 [fd 5 100
Epoch 15	repeat 2 [fd 688 lt 90 fd 349 lt 90]	repeat 3 [fd 669 lt 120]
Epoch 30	repeat 2 [fd 680 lt 90 fd 399 lt 90]	repeat 3 [fd 667 lt 120]
Epoch 45	repeat 2 [fd 590 lt 90 fd 369 lt 90]	repeat 3 [fd 666 lt 120]
Epoch 60	repeat 2 [fd 580 lt 90 fd 328 lt 90]	repeat 3 [fd 696 lt 120]
Epoch 75	repeat 2 [fd 620 lt 90 fd 311 lt 90]	repeat 3 [fd 679 lt 120]
Epoch 90	repeat 2 [fd 690 lt 90 fd 311 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 105	repeat 2 [fd 500 lt 90 fd 361 lt 90]	repeat 3 [fd 689 lt 120]
Epoch 120	repeat 2 [fd 590 lt 90 fd 361 lt 90]	repeat 3 [fd 689 lt 120]
Epoch 135	repeat 2 [fd 591 lt 90 fd 371 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 150	repeat 2 [fd 581 lt 90 fd 374 lt 90]	repeat 3 [fd 691 lt 120]
Epoch 165	repeat 2 [fd 590 lt 90 fd 370 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 180	repeat 2 [fd 599 lt 90 fd 370 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 195	repeat 2 [fd 591 lt 90 fd 370 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 210	repeat 2 [fd 599 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 225	repeat 2 [fd 599 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 240	repeat 2 [fd 599 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 255	repeat 2 [fd 590 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 270	repeat 2 [fd 596 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 285	repeat 2 [fd 590 lt 90 fd 371 lt 90]	repeat 3 [fd 690 lt 120]

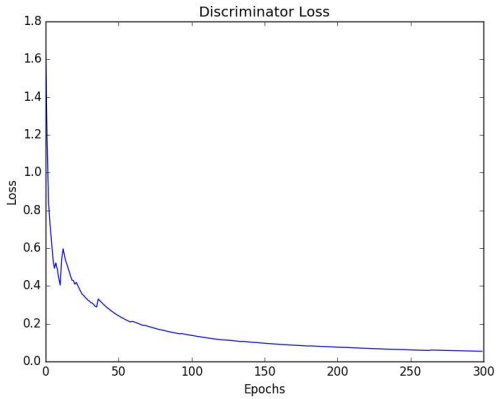
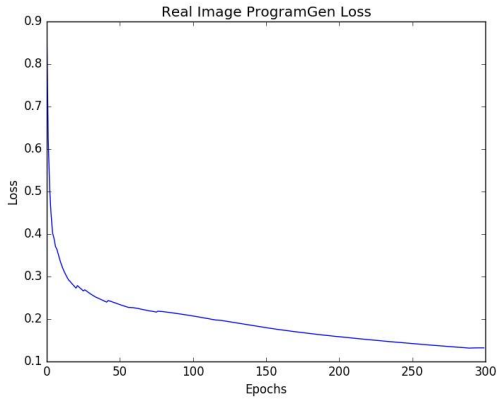
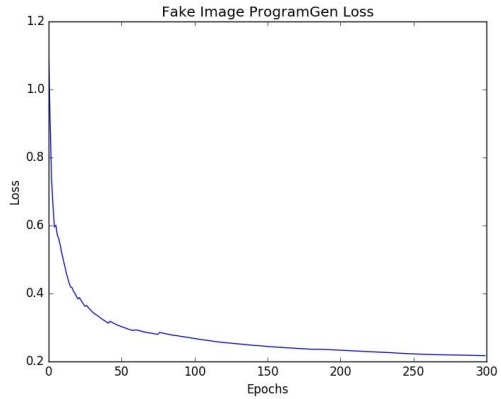
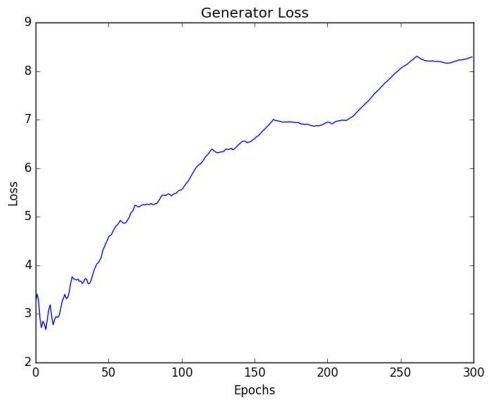
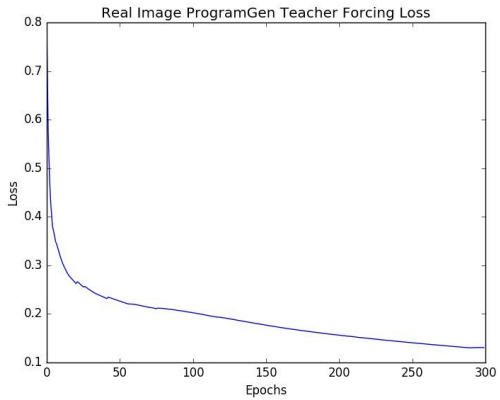
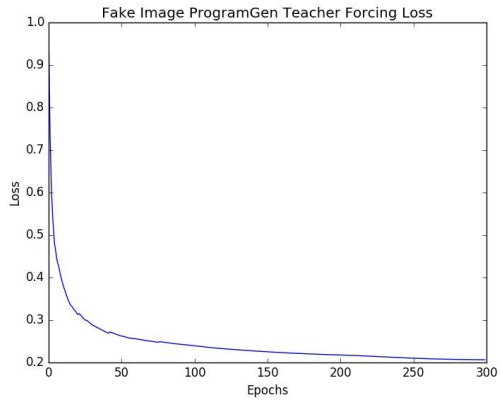
Basic Dataset without caption attention, Fake Image while Training

Actual Code	repeat 2 [fd 595 lt 90 fd 371 lt 90]	repeat 2 [fd 586 lt 90 fd 524 lt 90]
Epoch 0	repeat 2 [fd 4 9	repeat 3 [fd 5 100
Epoch 15	repeat 2 [fd 581 lt 90 fd 211 lt 90]	repeat 3 [fd 669 lt 120]
Epoch 30	repeat 2 [fd 681 lt 90 fd 311 lt 90]	repeat 3 [fd 677 lt 120]
Epoch 45	repeat 2 [fd 690 lt 90 fd 301 lt 90]	repeat 3 [fd 666 lt 120]
Epoch 60	repeat 2 [fd 680 lt 90 fd 428 lt 90]	repeat 3 [fd 596 lt 120]
Epoch 75	repeat 2 [fd 610 lt 90 fd 321 lt 90]	repeat 3 [fd 691 lt 120]
Epoch 90	repeat 2 [fd 520 lt 90 fd 211 lt 90]	repeat 3 [fd 688 lt 120]
Epoch 105	repeat 2 [fd 680 lt 90 fd 421 lt 90]	repeat 3 [fd 696 lt 120]
Epoch 120	repeat 2 [fd 681 lt 90 fd 321 lt 90]	repeat 3 [fd 676 lt 120]
Epoch 135	repeat 2 [fd 611 lt 90 fd 211 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 150	repeat 2 [fd 580 lt 90 fd 299 lt 90]	repeat 3 [fd 686 lt 120]
Epoch 165	repeat 2 [fd 680 lt 90 fd 300 lt 90]	repeat 3 [fd 696 lt 120]
Epoch 180	repeat 2 [fd 510 lt 90 fd 391 lt 90]	repeat 3 [fd 696 lt 120]
Epoch 195	repeat 2 [fd 689 lt 90 fd 321 lt 90]	repeat 3 [fd 690 lt 120]
Epoch 210	repeat 2 [fd 500 lt 90 fd 211 lt 90]	repeat 3 [fd 696 lt 120]
Epoch 225	repeat 2 [fd 510 lt 90 fd 399 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 240	repeat 2 [fd 610 lt 90 fd 391 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 255	repeat 2 [fd 610 lt 90 fd 321 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 270	repeat 2 [fd 500 lt 90 fd 261 lt 90]	repeat 3 [fd 699 lt 120]
Epoch 285	repeat 2 [fd 510 lt 90 fd 490 lt 90]	repeat 3 [fd 699 lt 120]

Basic Dataset without caption attention, while Testing

Real Image	Fake Image	Real Code	Fake Code
		repeat 3 [fd 483 lt 120]	repeat 3 [fd 471 lt 120]
		repeat 2 [fd 311 lt 90 fd 409 lt 90]	repeat 2 [fd 301 lt 90 fd 424 lt 90]
		repeat 3 [fd 653 lt 120]	repeat 3 [fd 619 lt 120]
		repeat 2 [fd 683 lt 90 fd 617 lt 90]	repeat 2 [fd 689 lt 90 fd 508 lt 90]
		circle 147	circle 144

Basic Dataset without caption attention Plots

Discriminator Loss	Real Image ProgramGen Loss	Fake Image ProgramGen Loss
		
Generator Loss	**Real Image ProgramGen Teacher Forcing Loss**	**Fake Image ProgramGen Teacher Forcing Loss**
		

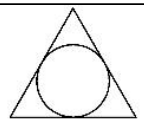
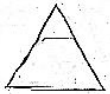
Complex Dataset with caption attention, Real Image while Training

Actual Code	circle 84 pu rt 90 fd 43 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 0	circle 1	repeat [4
Epoch 15	circle 92 pu rt 90 fd 33 rt 270 pd circle 116	repeat 6 [fd 236 rt 144]
Epoch 30	circle 80 pu rt 90 fd 44 rt 270 pd circle 124	repeat 6 [fd 390 rt 144]
Epoch 45	circle 85 pu rt 90 fd 47 rt 270 pd circle 127	repeat 6 [fd 303 rt 144]
Epoch 60	circle 88 pu rt 90 fd 48 rt 270 pd circle 127	repeat 6 [fd 304 rt 144]
Epoch 75	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 303 rt 144]
Epoch 90	circle 84 pu rt 90 fd 49 rt 270 pd circle 122	repeat 6 [fd 302 rt 144]
Epoch 105	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 120	circle 88 pu rt 90 fd 47 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 135	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 150	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 165	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 180	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 195	circle 82 pu rt 90 fd 45 rt 270 pd circle 124	repeat 6 [fd 302 rt 144]
Epoch 210	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 225	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 240	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]

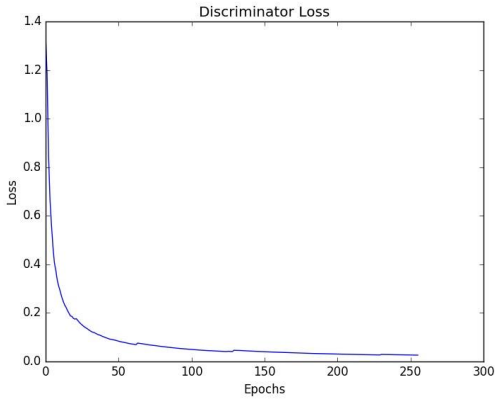
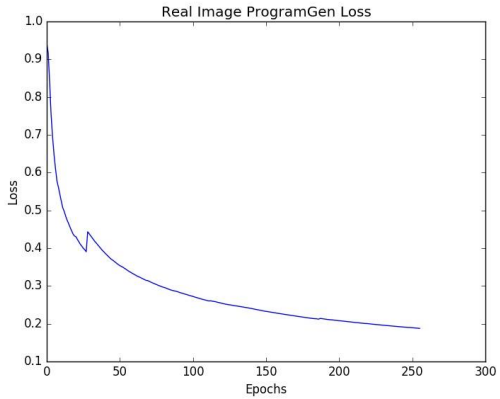
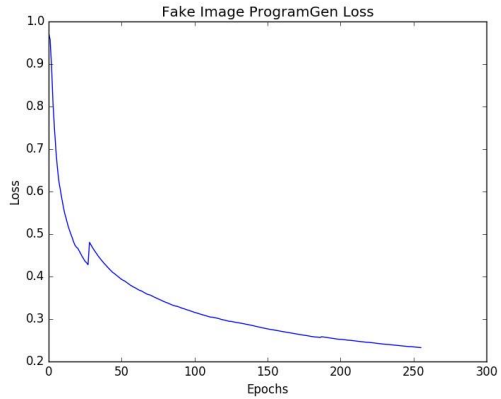
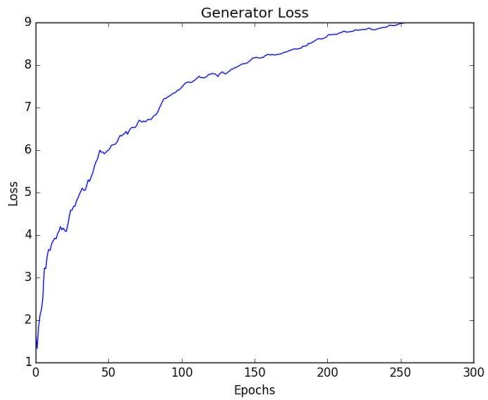
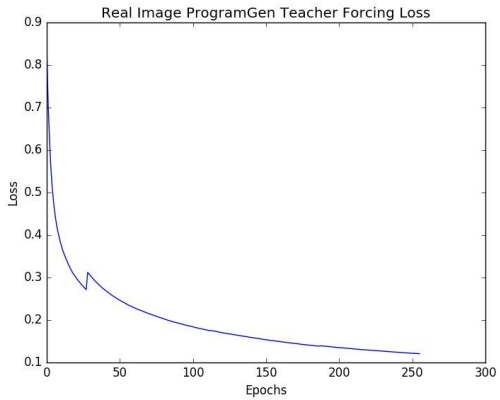
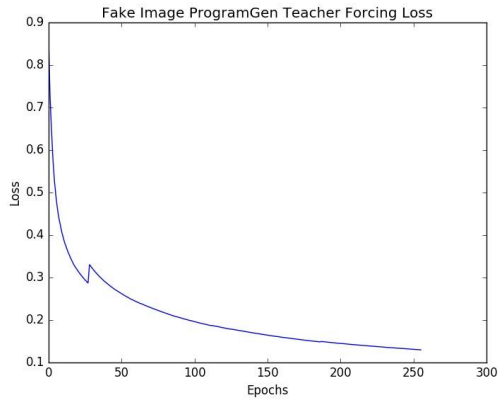
Complex Dataset with caption attention, Fake Image while Training

Actual Code	circle 84 pu rt 90 fd 43 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 0	circle 1	repeat [[
Epoch 15	circle 92 pu rt 90 fd 33 rt 270 pd circle 116	repeat 6 [fd 330 rt 144]
Epoch 30	circle 88 pu rt 90 fd 44 rt 270 pd circle 124	repeat 6 [fd 394 rt 144]
Epoch 45	circle 82 pu rt 90 fd 47 rt 270 pd circle 127	repeat 6 [fd 363 rt 144]
Epoch 60	circle 88 pu rt 90 fd 48 rt 270 pd circle 127	repeat 6 [fd 304 rt 144]
Epoch 75	circle 82 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 303 rt 144]
Epoch 90	circle 82 pu rt 90 fd 49 rt 270 pd circle 122	repeat 6 [fd 302 rt 144]
Epoch 105	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 120	circle 88 pu rt 90 fd 47 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 135	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 150	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 165	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 180	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 195	circle 82 pu rt 90 fd 45 rt 270 pd circle 124	repeat 6 [fd 302 rt 144]
Epoch 210	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 225	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]
Epoch 240	circle 88 pu rt 90 fd 49 rt 270 pd circle 127	repeat 6 [fd 302 rt 144]

Complex Dataset with caption attention, while Testing

Real Image	Fake Image	Real Code	Fake Code
		repeat 6 [fd 481 rt 144]	repeat 6 [fd 491 rt 144]
		fd 237 lt 47 fd 241 lt 133 fd 237 lt 47 fd 241	fd 231 lt 47 fd 231 lt 133 fd 239 lt 47 fd 23
		circle 62 pu rt 90 fd 33 rt 270 pd circle 95 pu rt 90 fd 47 rt 270 pd circle 142	circle 62 pu rt 90 fd 36 rt 270 pd circle 93 pu rt 90 fd 49 rt 270 pd circle 140
		circle 111 pu lt 90 fd 111 rt 90 fd 147 rt 90 fd 104 rt 270 pd circle 104	circle 110 pu lt 90 fd 111 rt 90 fd 143 rt 90 fd 101 rt 270 pd circle 104
		repeat 3 [fd 667 lt 120] pu fd 333.5 pd circle 385.09	repeat 3 [fd 675 lt 120] pu fd 334.5 pd circle 302.5

Complex Dataset with caption attention Plots

Discriminator Loss	Real Image ProgramGen Loss	Fake Image ProgramGen Loss
		
Generator Loss	**Real Image ProgramGen Teacher Forcing Loss**	**Fake Image ProgramGen Teacher Forcing Loss**
		

Conclusion

- The model is able to generate syntactically and semantically correct code for both basic and complex image

Conclusion

- The model is able to generate syntactically and semantically correct code for both basic and complex image
- The dimensions corresponding to the size in the image become less accurate upon the removal of caption embedding in annotation vector

Conclusion

- The model is able to generate syntactically and semantically correct code for both basic and complex image
- The dimensions corresponding to the size in the image become less accurate upon the removal of caption embedding in annotation vector
- The fake images generated in the case of Complex Dataset are not very accurate for overlapping regions

Future Work

- Compare the results without using the VGG network generated annotation vectors to determine their effect in generating accurate of dimensions in synthesized logo code

Future Work

- Compare the results without using the VGG network generated annotation vectors to determine their effect in generating accurate of dimensions in synthesized logo code
- Implement a Baseline model for the task like a sequence to sequence to model that takes text caption as input and generates logo program as output

Future Work

- Compare the results without using the VGG network generated annotation vectors to determine their effect in generating accurate of dimensions in synthesized logo code
- Implement a Baseline model for the task like a sequence to sequence to model that takes text caption as input and generates logo program as output
- Quantify the results using some metric like L2 loss between the images created by synthesized code and actual logo code

The End