HOMEWORK 6

Dario Placencio - 907 284 6018

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. Answers to the questions that are not within the pdf are not accepted. This includes external links or answers attached to the code implementation. Late submissions may not be accepted. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework. It is ok to share the results of the experiments and compare them with each other.

1 Implementation: GAN (50 pts)

In this part, you are expected to implement GAN with MNIST dataset. We have provided a base jupyter notebook (gan-base.ipynb) for you to start with, which provides a model setup and training configurations to train GAN with MNIST dataset.

(a) Implement training loop and report learning curves and generated images in epoch 1, 50, 100. Note that drawing learning curves and visualization of images are already implemented in provided jupyter notebook. (20 pts)

Procedure 1 Training GAN, modified from Goodfellow et al. (2014)

```
Input: $m$: real data batch size, $n_z$: fake data batch size
Output: Discriminator $D$, Generator $G$
  for number of training iterations do
      # Training discriminator
      Sample minibatch of $n_z$ noise samples $\{z^{(1)}, z^{(2)}, \cdots, z^{(n_z)}\}$ from noise prior $p_g(z)$
      Sample minibatch of $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$
      Update the discriminator by ascending its stochastic gradient:
          $$\nabla_{\theta_d}\Big(\frac{1}{m}\sum_{i=1}^{m}\log D(x^{(i)}) + \frac{1}{n_z}\sum_{i=1}^{n_z}\log(1 - D(G(z^{(i)})))\Big)$$
      # Training generator
      Sample minibatch of $n_z$ noise samples $\{z^{(1)}, z^{(2)}, \cdots, z^{(n_z)}\}$ from noise prior $p_g(z)$
      Update the generator by ascending its stochastic gradient:
          $$\nabla_{\theta_g}\frac{1}{n_z}\sum_{i=1}^{n_z}\log D(G(z^{(i)})))$$
  end for
  # The gradient-based updates can use any standard gradient-based learning rule. In the base code, we are using Adam optimizer (Kingma and Ba, 2014)
```

Expected results are as follows.

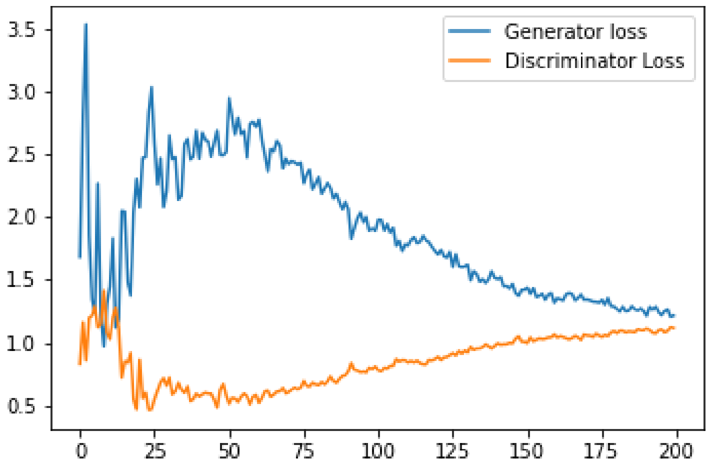
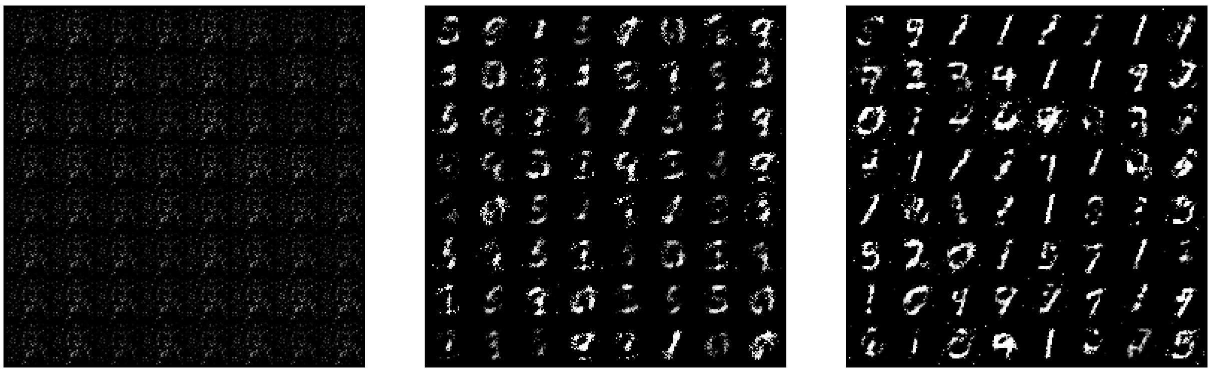


Figure 1: Learning curve



(a) epoch 1 (b) epoch 50 (c) epoch 100

Figure 2: Generated images by G

These are the results of executing the first GAN, code attached below.

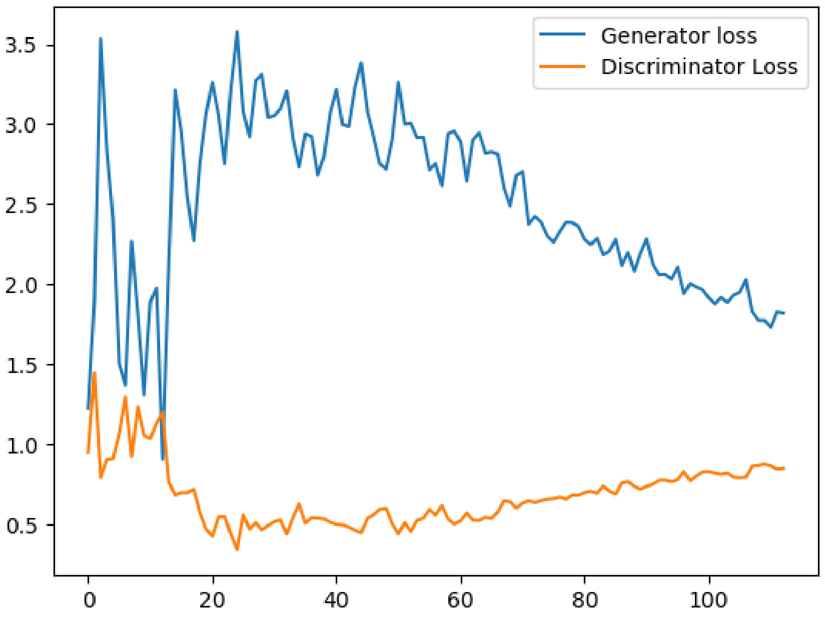
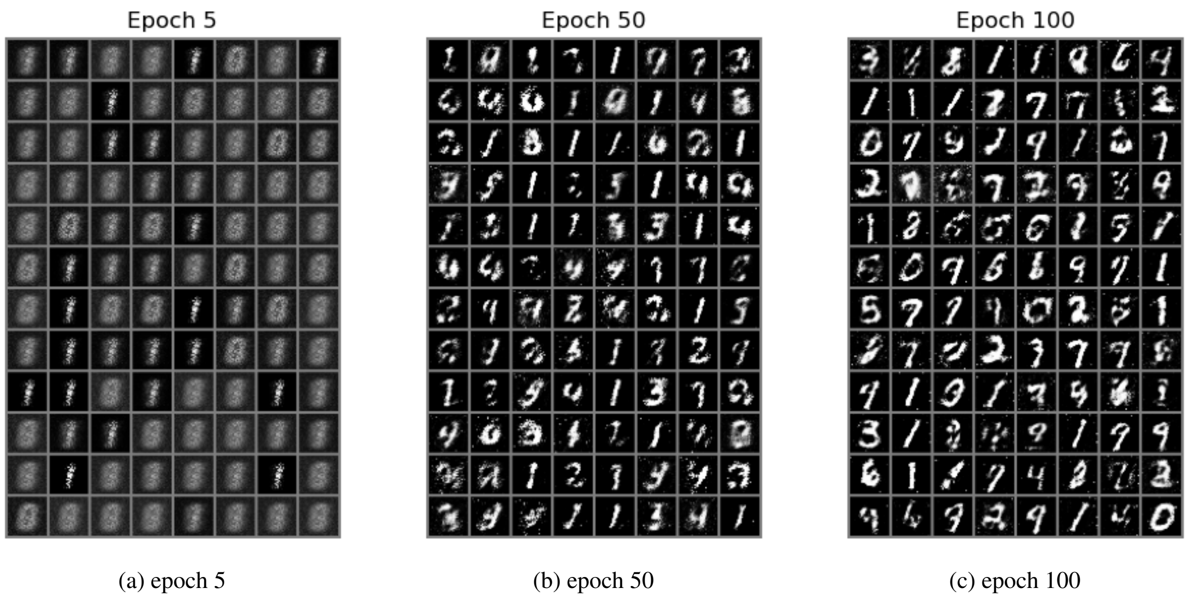


Figure 3: Learning curve for GAN (a)



(a) epoch 5 (b) epoch 50 (c) epoch 100

Figure 4: Epochs for GAN (a)

(b) Replace the generator update rule as the original one in the slide,
"Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{n_z} \sum_{i=1}^{n_z} \log(1 - D(G(z^{(i)})))$$

" , and report learning curves and generated images in epoch 1, 50, 100. Compare the result with (a). Note that it may not work. If training does not work, explain why it doesn't work.
You may find this helpful: https://jonathan-hui.medium.com/gan-what-is-wrong-with-the-gan-cost-function-6f594162ce01 (10 pts)

These are the results of executing the second GAN, code attached below.

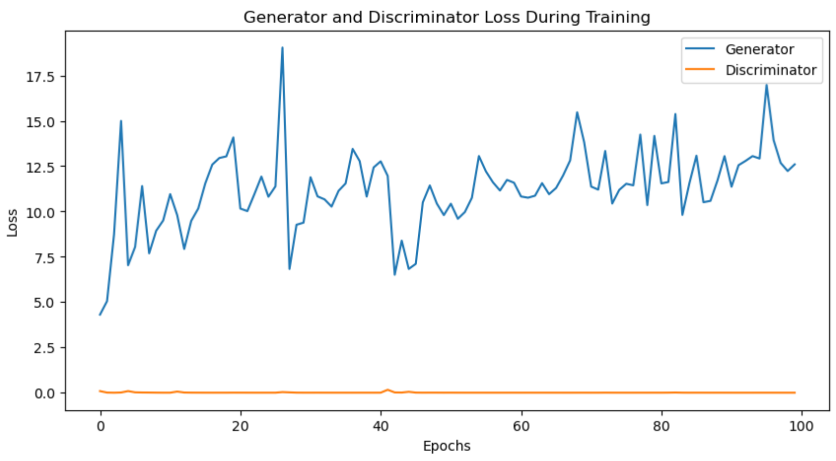
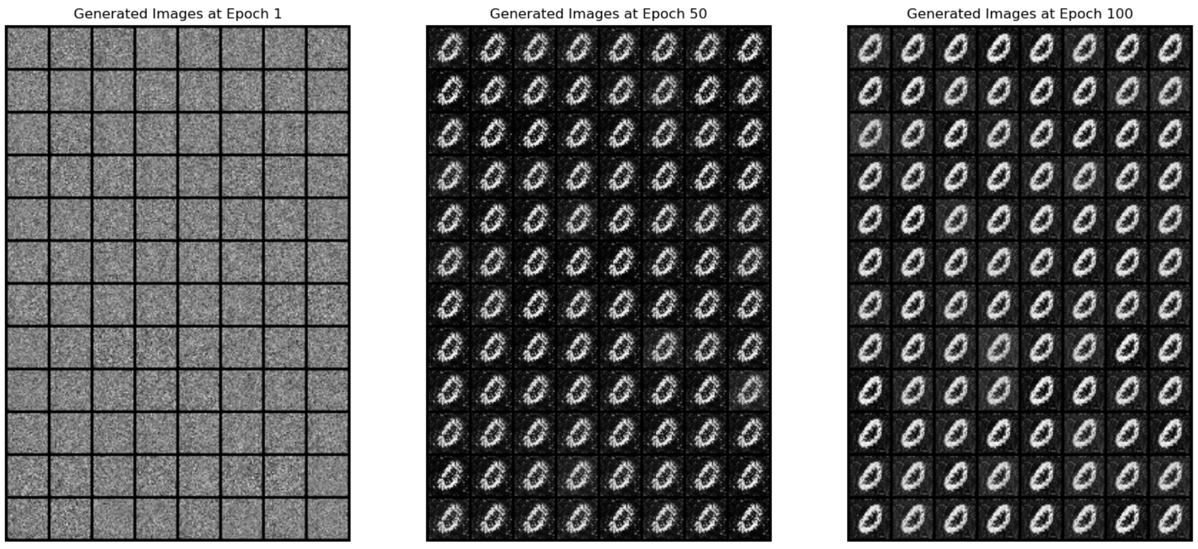


Figure 5: Learning curve for GAN (b)


Generated Images at Epoch 1
Generated Images at Epoch 50
Generated Images at Epoch 100

(a) epoch 5 (b) epoch 50 (c) epoch 100

Figure 6: Epochs for GAN (b)

In the comparative analysis of the GAN models, the substitution of Adam with SGD for the generator has led to notably different results.

With Adam as the optimizer, the generator consistently produced images that gradually increased in quality and recognizability across epochs. This improvement is likely due to Adam's adaptive learning rate mechanism, which adjusts based on the model's performance.

Conversely, when using SGD, the images generated did not exhibit the same level of progression and remained less distinct. This can be attributed to the constant learning rate of SGD, which does not account for the nuanced needs of the generator's training process. The learning curve with SGD also showed greater fluctuations in the generator's loss, indicating challenges in optimizing the model effectively.

These observations suggest that the adaptive qualities of Adam are better suited for the intricate dynamics of training GANs, a hypothesis that is supported by the more stable and successful learning trajectory observed when Adam is employed.

(c) Except the method that we used in (a), how can we improve training for GAN? Implement that and report your setup, learning curves, and generated images in epoch 1, 50, 100. This question is an open-ended question and you can choose whichever method you want. (20 pts)

These are the results of executing the third GAN, code attached below.

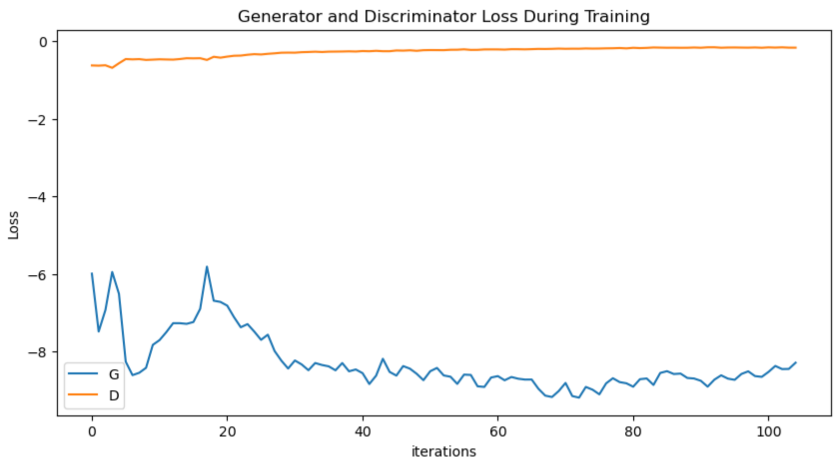
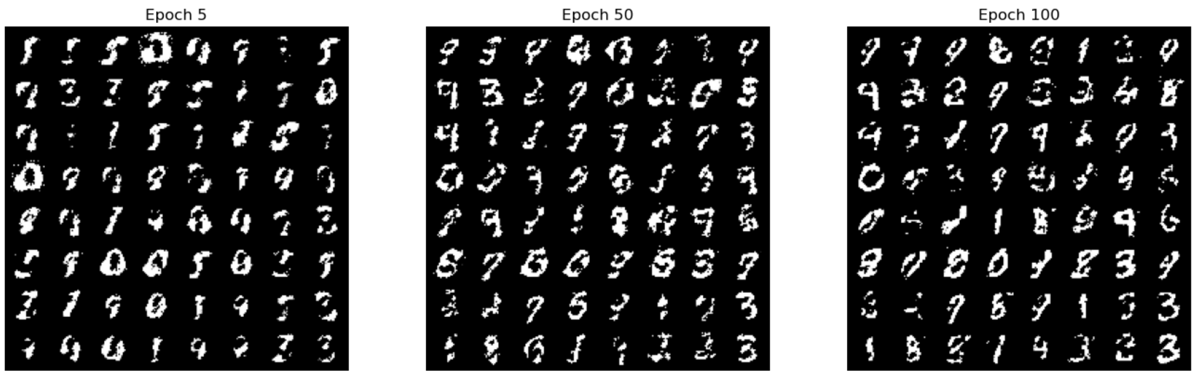


Figure 7: Learning curve for GAN (c)



(a) epoch 5

(b) epoch 50

(c) epoch 100

Figure 8: Epochs for GAN (c)

The following improvements were made in the GAN model compared to the first original model:

(a) **Spectral Normalization in Discriminator:** Spectral normalization was applied in the discriminator's layers. This method normalizes the weights, controlling the Lipschitz constant to stabilize the training by preventing the discriminator from overpowering the generator.

(b) **Wasserstein Loss with Gradient Penalty (WGAN-GP):** The model employs the Wasserstein loss function, offering greater stability than Binary Cross Entropy loss. It minimizes an estimate of the Earth Mover's distance between real and generated distributions. The gradient penalty enforces a 1-Lipschitz constraint, enhancing training stability and mitigating mode collapse.

(c) **Differentiated Learning Rates and Adam Optimizer Parameters:** Distinct learning rates and beta parameters for the generator and discriminator optimize each network's training efficiency, addressing the adversarial setup's unique challenges.

(d) **Learning Rate Schedulers:** These schedulers gradually reduce the learning rates during training, aiding in fine-tuning the optimization process for better convergence and stability.

(e) **Noise Addition to Discriminator Inputs:** Adding noise to the discriminator's inputs serves as regularization, preventing overfitting to the training data and improving the diversity of the generated images.

These enhancements focus on increasing the stability and quality of the GAN training process, addressing common challenges such as training instability and mode collapse, and improving the diversity and realism of the generated images.

2 Directed Graphical Model [25 points]

Consider the directed graphical model (aka Bayesian network) in Figure 9.

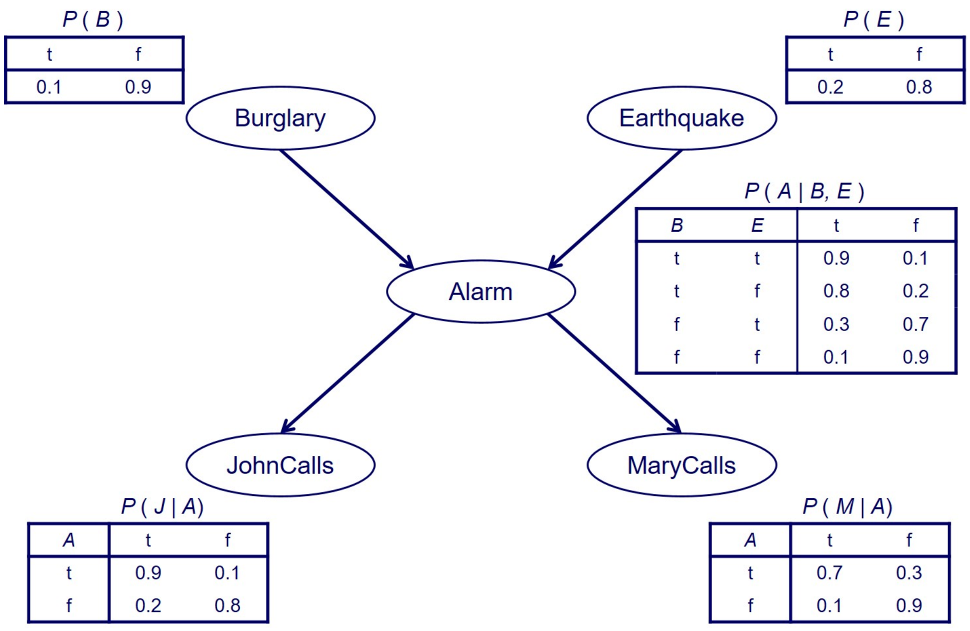


Figure 9: A Bayesian Network example.

Compute $P(B = t \mid E = f, J = t, M = t)$ and $P(B = t \mid E = t, J = t, M = t)$. (10 points for each) These are the conditional probabilities of a burglar in your house (yikes!) when both of your neighbors John and Mary call you and say they hear an alarm in your house, but without or with an earthquake also going on in that area (what a busy day), respectively.

The conditional probabilities can be calculated as follows:
Given that $P(B) = 0.1$, $P(E) = 0.2$, and the conditional probability tables for $P(A|B, E)$, $P(J|A)$, and $P(M|A)$, we can calculate:

$$\begin{aligned} P(A = t) = {} & P(B = t, E = t)P(A = t|B = t, E = t) \\ & + P(B = t, E = f)P(A = t|B = t, E = f) \\ & + P(B = f, E = t)P(A = t|B = f, E = t) \\ & + P(B = f, E = f)P(A = t|B = f, E = f) \end{aligned}$$

After computing $P(A = t)$, we have $P(A = f) = 1 - P(A = t)$.
We can then find $P(J = t, M = t)$ which is the marginal likelihood of the evidence:

$$P(J = t, M = t) = P(J = t, M = t|A = t)P(A = t) + P(J = t, M = t|A = f)P(A = f)$$

Using Bayes' theorem, the posterior probabilities for B given J, M, and E are calculated as:

$$P(B = t|E = f, J = t, M = t) = \frac{P(B = t)P(A = t|B = t, E = f)P(J = t, M = t|A = t)}{P(J = t, M = t)}$$

$$P(B = t|E = t, J = t, M = t) = \frac{P(B = t)P(A = t|B = t, E = t)P(J = t, M = t|A = t)}{P(J = t, M = t)}$$

Substituting the values and solving gives us:

$$P(B = t|E = f, J = t, M = t) \approx 0.411$$

$$P(B = t|E = t, J = t, M = t) \approx 0.237$$

Thus, the conditional probabilities of a burglary given the evidence of neighbors' calls with and without an earthquake are approximately 41.1% and 23.7% respectively.

3 Chow-Liu Algorithm [25 pts]

Suppose we wish to construct a directed graphical model for 3 features X, Y, and Z using the Chow-Liu algorithm. We are given data from 100 independent experiments where each feature is binary and takes value T or F. Below is a table summarizing the observations of the experiment:

X	Y	Z	Count
T	T	T	36
T	T	F	4
T	F	T	2
T	F	F	8
F	T	T	9
F	T	F	1
F	F	T	8
F	F	F	32

1. Compute the mutual information $I(X, Y)$ based on the frequencies observed in the data. (5 pts)
2. Compute the mutual information $I(X, Z)$ based on the frequencies observed in the data. (5 pts)
3. Compute the mutual information $I(Z, Y)$ based on the frequencies observed in the data. (5 pts)
4. Which undirected edges will be selected by the Chow-Liu algorithm as the maximum spanning tree? (5 pts)
5. Root your tree at node X, assign directions to the selected edges. (5 pts)

1. Mutual Information $I(X, Y)$: The mutual information between X and Y is calculated using their joint and individual probabilities. The result is:

$$I(X, Y) \approx 0.278$$

2. Mutual Information $I(X, Z)$: Similarly, the mutual information between X and Z is computed, yielding:

$$I(X, Z) \approx 0.133$$

3. Mutual Information $I(Z, Y)$: The mutual information between Z and Y is the highest among the three pairs, calculated as:

$$I(Z, Y) \approx 0.397$$

4. Maximum Spanning Tree (Edges Selected by the Chow-Liu Algorithm): Based on the calculated mutual information, the Chow-Liu algorithm selects the edges with the highest mutual information to form the maximum spanning tree.
The selected edges are:

- Edge $Y - Z$ (highest mutual information)
- Edge $X - Y$ (second highest mutual information)

The edge $X - Z$ is not selected as it has the lowest mutual information.

5. Rooting the Tree at Node X with Directed Edges: After forming the tree, it is rooted at node X, and directions are assigned to the edges. The final directed tree structure is: - $X \to Y$ - $Y \to Z$

This assignment follows from the convention of directing edges away from the root in a directed tree.

References

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.