Logistic Regressions and Generative Adversarial Networks

Roger Kutyna

Clark University School of Professional Studies

MSIT 3103: Generative AI Model Development

Onur Barut

September 12, 2025

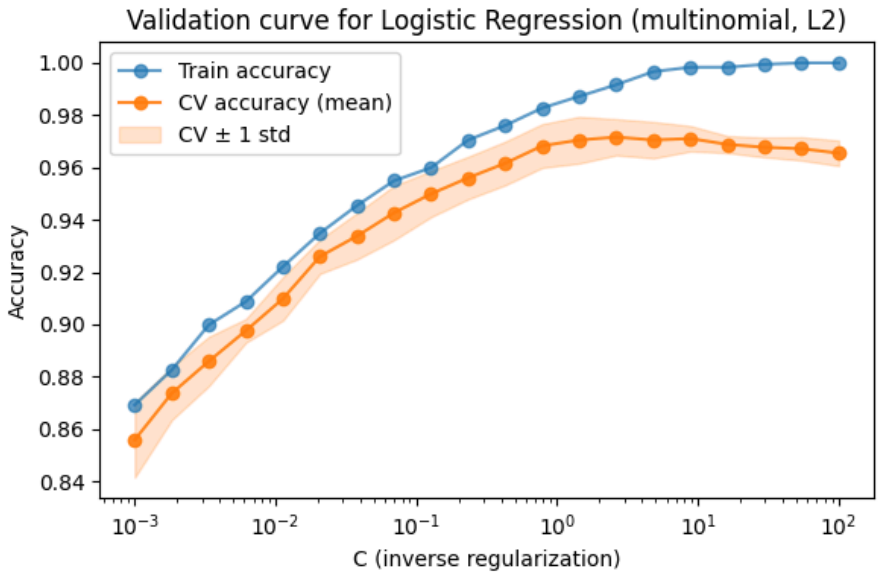
Introduction

For this assignment, I chose to work with the MNIST dataset that was included in the assignment file. For models, I chose to use the logistic regression model that is built into scikit-learn as well as the GAN that was included in the starting file.

In order to determine the best parameters of the logistic regression, I tested both the l1 and l2 penalties using the saga solver to ensure that both models were the same besides the penalty. The l1 penalty uses the absolute value of the coefficients to apply penalties while the l2 coefficient uses the square of the coefficient to apply penalties.

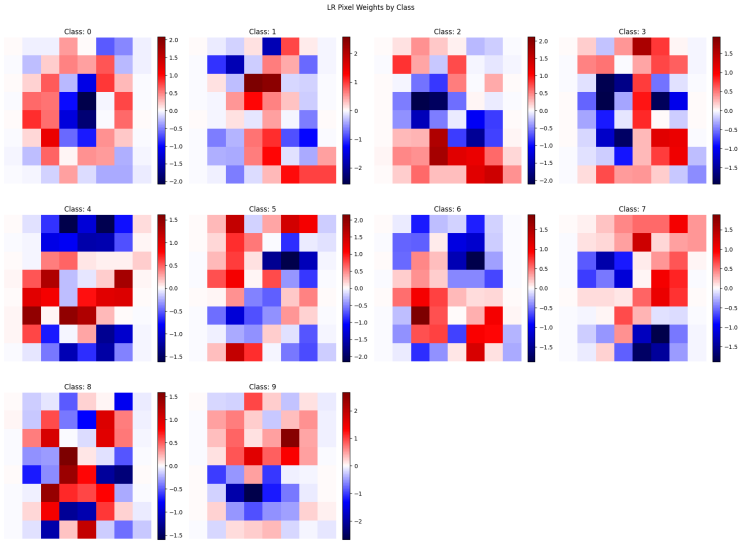
Results

Through my testing, both l1 and l2 scored exactly the same, both with 95.83% accuracy. I then tested each function across different C values. C determines how strict the regularization is when these penalties are applied. As can be seen in the below figure of c value vs accuracy, the accuracy increases with the C value until it reaches a plateau and slowly drops off. This indicates that the accuracy increases as the model is fit closer to the training examples, until it reaches a point where it starts to overfit and can no longer generalize to the test data. The best C value was 2.6367 and resulted in an accuracy of 97.16%.

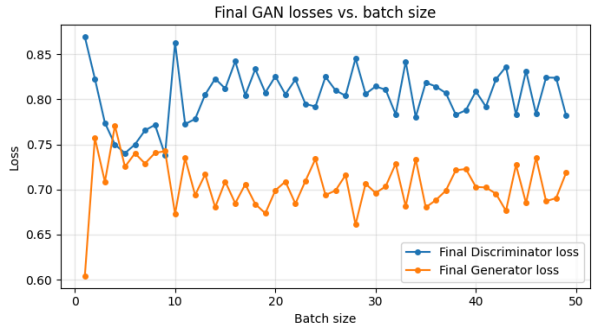


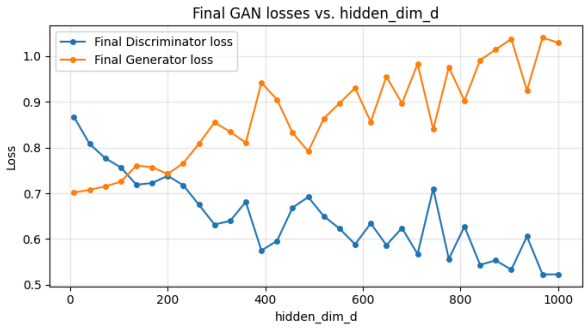
I tested this with both l1 and l2 and found that l2 generalizes better, while l1 performed better in the train accuracy but appeared to overfit more easily.

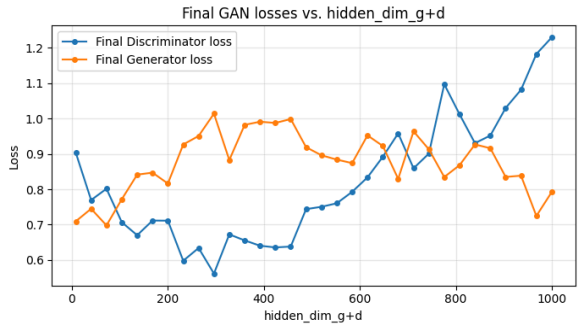
After this, I utilized GPT-5 (low reasoning) to generate a function to view the heatmaps of the pixels for each class in order to understand which weights/pixels had the most influence for each class. As can be seen below, the pixels that make up the shape of the number have a high weight. However, the part that I find most interesting are the negative pixel weights. These seem to help to differentiate between numbers that have a similar shape. For example, the classes 6 and 8 look very similar when written. When looking at the weights for 6, the most negative pixels are those to the upper right of the image, which happens to be where the 8 has very high weights. This is the only part of the shapes that do not overlap and likely indicate that they play a large role in differentiating between the two.

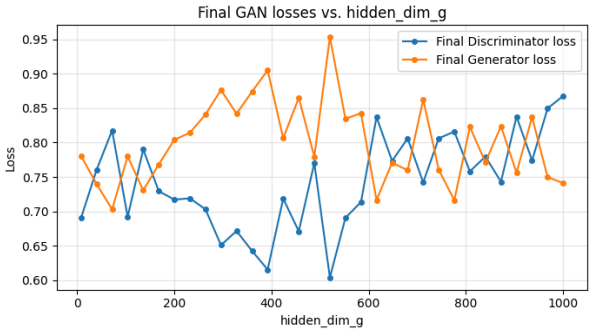


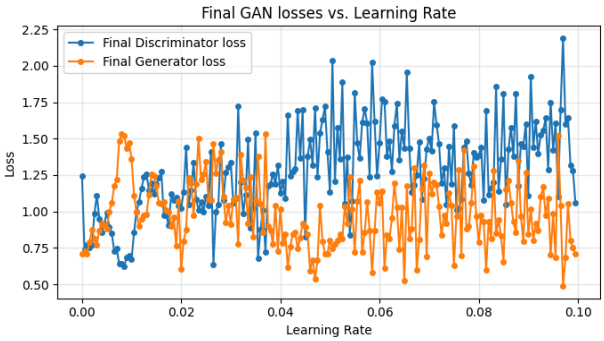
After testing the logistic regression, I began testing the Generative Adversarial Network. I first used GPT-5 (medium reasoning) to turn the included GAN model into a function with parameters that I could tweak as I tested. This included the additional control of the number of neurons within the hidden layers of the generator and discriminator, along with the number of hidden layers. I then did several experiments which plotted the final loss after training of the generator and discriminator against the batch size, learning rate, number of training iterations, size of the generator hidden layer, size of the discriminator hidden layer, and the size of both the discriminator and generator hidden layers.

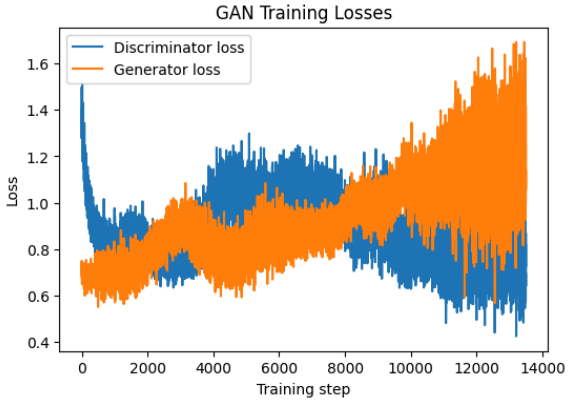






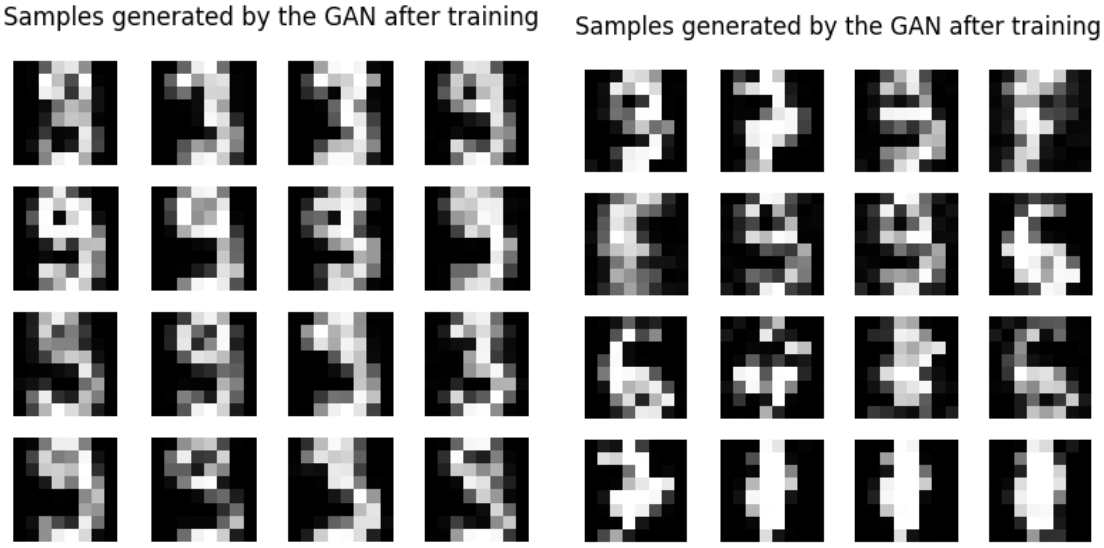






The unfortunate realization that came along with this testing, and subsequent testing based on the results shown above, is that these loss graphs do not indicate performance. They indicate model collapse, but even when they remain close, the results produced by the GAN are not necessarily going to be better. The best looking numbers that I was able to produce actually occurred due to what Goodfellow et al. described as the Helvetica Scenario, in which the generator learns that it can score relatively well while only producing a small subset of the classes in the original data. For me, this manifested in the model producing solely 3's and 9's.

This result can be seen below, along with the output of the closest I was able to get to achieving an equilibrium of losses between the generator and discriminator:



Conclusion

Based on this experimentation, I would be much more comfortable deploying a logistic regression to a real world scenario as it is substantially more consistent and easier to train. It also ran much much faster than the GANs. Logistic regressions could be reliably used for any sort of classification in industry. That being said, I experimented with logistic regressions and GANs on the CIFAR-10 dataset, in which the logistic regression was only able to get about 33% accuracy (viewable in the assignment1_extra.ipynb file), so I would imaging that this would struggle in more complex situations. For GANs, I would not want to work with them in any real-world application unless I was given substantial time and compute to test many combinations of parameters and review the results. Like the logistic regression, I tested a GAN with the CIFAR-10 dataset and was only able to get it to generate static. This was true even when training only on a single class, so this would also be much more difficult in more complex situations.

Works Cited:

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June). *Generative adversarial nets*. arXiv. https://arxiv.org/abs/1406.2661