

Final Project

CSCI E-80A Intro to Artificial Intelligence

Alyssa Blanco

As stated in my Final Project Proposal, I created a web interface that can execute a GAN model and display the outcoming images.

To do this, I first took the code from the GAN assignment (week 10) and improved it. Then, I created a Flask app (for the first time ever) and locally hosted a webpage that could successfully run the improved model and display the results.

Improving the GAN model from week 10

The first thing I did to improve the GAN model was adding more layers to the discriminator and generator models. I added three Leaky ReLU layers with an alpha of 0.02 to both models. I did this to prevent gradient death, making sure the algorithm continues improving over time. I also added three Dropout layers to the discriminator to prevent the model from overfitting the data.

https://keras.io/api/layers/activation_layers/leaky_relu/
https://keras.io/api/layers/regularization_layers/dropout/

In assignment ten, I used the `.imshow` function from matplotlib library to display the resulting data into an image. To improve it, I removed the interpolation parameter and changed the `cmap` value from 'gray' to `plt.cm.binary`. I found this solution at:

(<https://stackoverflow.com/questions/58631088/why-are-my-neural-network-predictions-correct-when-applied-to-mnist-hand-drawn-i>) while researching a way to improve the clarity of generated images.

To try this new model, I wanted to run a really big test. The largest test I did, 1 million epochs, gave the best results I have been able to achieve since starting with this model in week 10. The results of this test are laid out in a later section of this report. After this I was happy with how the model performed and moved on to developing the Flask application.

Developing a Flask app

When researching how to accomplish my goal of running a model through a webpage, Flask was the popular answer. I was easily able to install flask to my environment using a pip command and learned how to create a simple app from the Flask documentation.

<https://flask.palletsprojects.com/en/2.2.x/quickstart/>

After setting up the environment, I created a simple web page with a button to start the model and a container to house all incoming images that are generated from the model. As the model

runs, the image container periodically reloads and new results appear on the page. To see examples of this webpage running, see sections below.

After creating this simple web page, I was able to import my improved model into the app and run it locally. I had never worked with Flask before but I am glad I had the chance during this project. I now believe this is one of the best ways to publish models for general use.

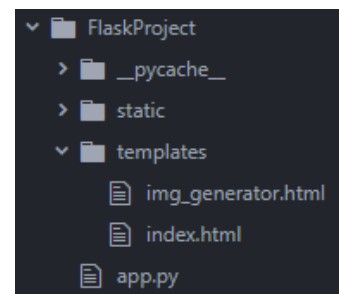
Running the Project & Documentation Links

https://www.dropbox.com/sh/80z18osarihiz5z/AAD41A1yv_kwBngTaFKCickGa?dl=0
https://github.com/alyssakblanco/MNIST_Img-Generator

Please see the Dropbox linked above for the video recordings of the finished project running on my local machine.

To run it locally yourself, clone the repo linked above and create an empty folder named “static” within the directory (on the same level as templates, see structure on the right). This will hold the resulting images. I couldn’t upload the empty folder to GitHub.

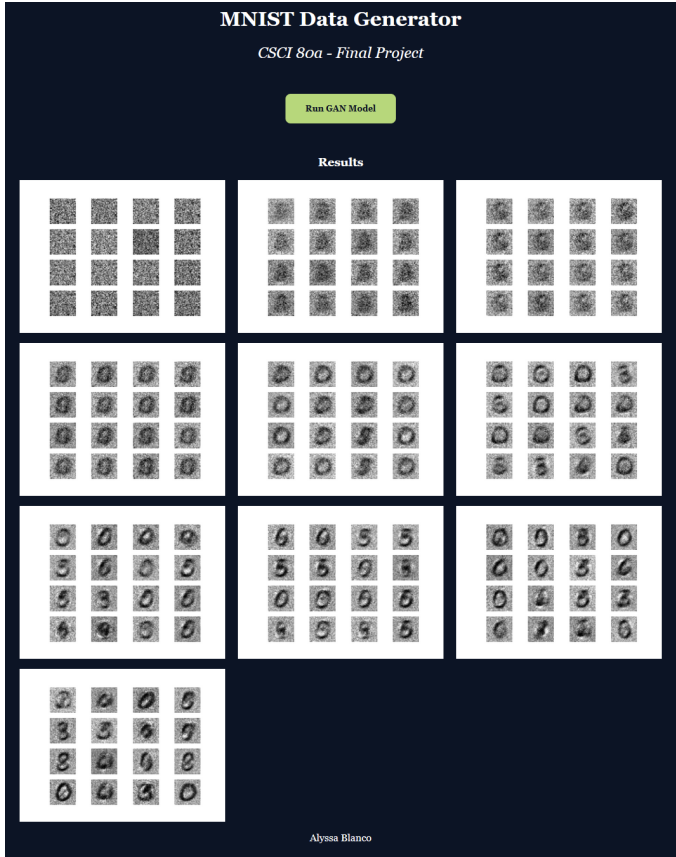
You will also need to install flask in the environment. Other than that, the environment configured from Week 1’s requirement.txt is where I developed this project.



Website Test Runs

Test Run 1	Test Run 2
Video https://www.dropbox.com/s/74bqsh8ng2okbm9/TestRun1_Video.mp4?dl=0	Video https://www.dropbox.com/s/uqqchmsdzrqu3ne/TestRun2_Video.mp4?dl=0

Screenshot



Console output:

Discriminator loss: 0.04372000694274902, Generator loss: 5.370907306671143
Discriminator acc.: 1.0, Generator acc.: 0.0

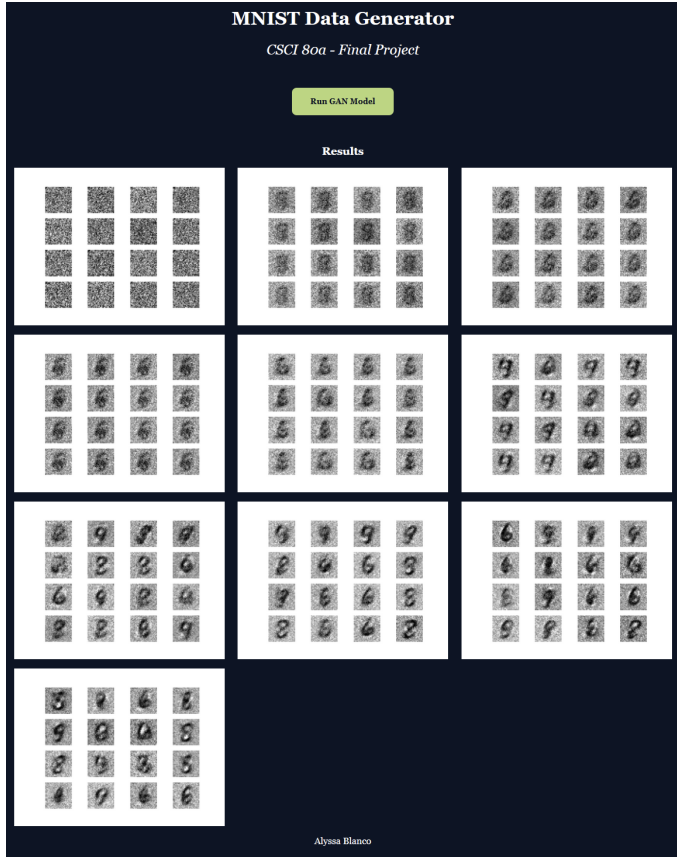
Discriminator loss: 0.14267945289611816, Generator loss: 4.643519878387451
Discriminator acc.: 0.9609375, Generator acc.: 0.0

Discriminator loss: 0.03422815352678299, Generator loss: 7.106220245361328
Discriminator acc.: 0.9921875, Generator acc.: 0.0

Discriminator loss: 0.029306022450327873, Generator loss: 4.962342739105225
Discriminator acc.: 1.0, Generator acc.: 0.0

Discriminator loss: 0.07719862461090088, Generator loss: 7.898516654968262
Discriminator acc.: 0.9609375, Generator acc.: 0.0

Screenshot



Console output:

Discriminator loss: 0.07407703995704651, Generator loss: 4.136655807495117
Discriminator acc.: 0.9921875, Generator acc.: 0.0

Discriminator loss: 0.041494689881801605, Generator loss: 3.6977193355560303
Discriminator acc.: 0.9921875, Generator acc.: 0.0

Discriminator loss: 0.30167654156684875, Generator loss: 16.607999801635742
Discriminator acc.: 0.875, Generator acc.: 0.0

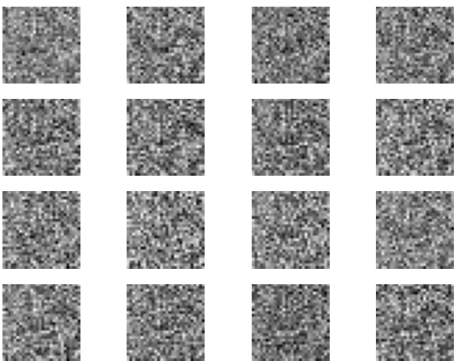
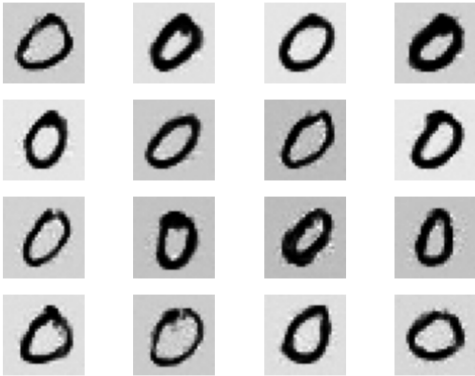

Discriminator loss: 0.11118004471063614, Generator loss: 5.399774551391602
Discriminator acc.: 0.9609375, Generator acc.: 0.0

Discriminator loss: 0.040356721729040146, Generator loss: 8.861858367919922
Discriminator acc.: 0.9765625, Generator acc.: 0.0

<p>Discriminator loss: 0.12389317154884338, Generator loss: 4.812526702880859 Discriminator acc.: 0.96875, Generator acc.: 0.0</p> <p>Discriminator loss: 0.06150910630822182, Generator loss: 5.8210601806640625 Discriminator acc.: 0.984375, Generator acc.: 0.0</p> <p>Discriminator loss: 0.14649926126003265, Generator loss: 6.886724472045898 Discriminator acc.: 0.9453125, Generator acc.: 0.0</p> <p>Discriminator loss: 0.07686930149793625, Generator loss: 5.941143035888672 Discriminator acc.: 0.96875, Generator acc.: 0.0</p> <p>Discriminator loss: 0.0763295590877533, Generator loss: 5.336113929748535 Discriminator acc.: 0.96875, Generator acc.: 0.0078125</p>	<p>Discriminator loss: 0.020295217633247375, Generator loss: 5.266066551208496 Discriminator acc.: 1.0, Generator acc.: 0.0</p> <p>Discriminator loss: 0.038718290627002716, Generator loss: 6.815834045410156 Discriminator acc.: 0.9921875, Generator acc.: 0.0</p> <p>Discriminator loss: 0.09321552515029907, Generator loss: 6.241243362426758 Discriminator acc.: 0.9765625, Generator acc.: 0.0</p> <p>Discriminator loss: 0.1219346672296524, Generator loss: 6.431574821472168 Discriminator acc.: 0.9453125, Generator acc.: 0.0</p> <p>Discriminator loss: 0.045609693974256516, Generator loss: 7.8395538330078125 Discriminator acc.: 0.96875, Generator acc.: 0.0085147</p>
---	---

1,000,000 epoch Test

Screenshots:

<p>Epoch 1</p>  <p>10:33 pm</p>	<p>Epoch 500,000</p>  <p>5:01 am</p>	<p>Epoch 1,000,000</p>  <p>11:57 am</p>
---	--	--

Console Output:

Epoch 1

Discriminator loss: 0.7368070483207703, Generator loss: 1.0907936096191406

Discriminator acc.: 0.4375, Generator acc.: 0.0

Epoch 500,000

Discriminator loss: 0.021076707169413567, Generator loss: 5.85882043838501

Discriminator acc.: 0.9921875, Generator acc.: 0.0

Epoch 1,000,000

Discriminator loss: 0.020910276100039482, Generator loss: 5.434567451477051

Discriminator acc.: 1.0, Generator acc.: 0.0234375

Summary of Tests

For both tests run through the website, I ran 1000 epochs and saved an image every 100 runs. I found this to be a good test to see how the webpage works, with visible results, without having to wait a long time. From the console output, we can see that the generated data is not extremely accurate yet, but the images produced show improvement. We can see in the last run of both tests, where the images are the clearest, that the generator gains that first bit of plausible accuracy.

For Test Run 3, I waited 13 hours for 1 million epochs to run (saving the output every 100,00 runs) and train the improved model. Unexpectedly, after epoch 850,000, the test images start to deteriorate and look like the final image output above. I expected the console output to show a decrease in accuracy here, but the final epoch had the highest generator accuracy (~0.02). I think this means either I did not do enough to avoid gradient death or I am mapping my images in a way that doesn't match what the model is discriminating.

Final Note

I really enjoyed working on this final project. I was able to combine all the knowledge I learned from this course with my experience in web development. I believe that in order for the amazing results of today's algorithms to be helpful for a wider variety of people, it needs to be hosted online. Going forward, I hope to continue improving this project and maintaining my newfound skills.

Thanks for a great semester!