Neural Networks Final Project

Objective: Using StyleGAN2-ADA-Pytorch implementation to create synthetic x-ray images using the MURA database to try to increase the accuracy of the densenet169 CNN created last semester

Aileen Dugan - Neural Networks - Spring 2023

What is StyleGAN2-ADA?

GAN = Generative Adversarial Network consisting of a generator and a discriminator

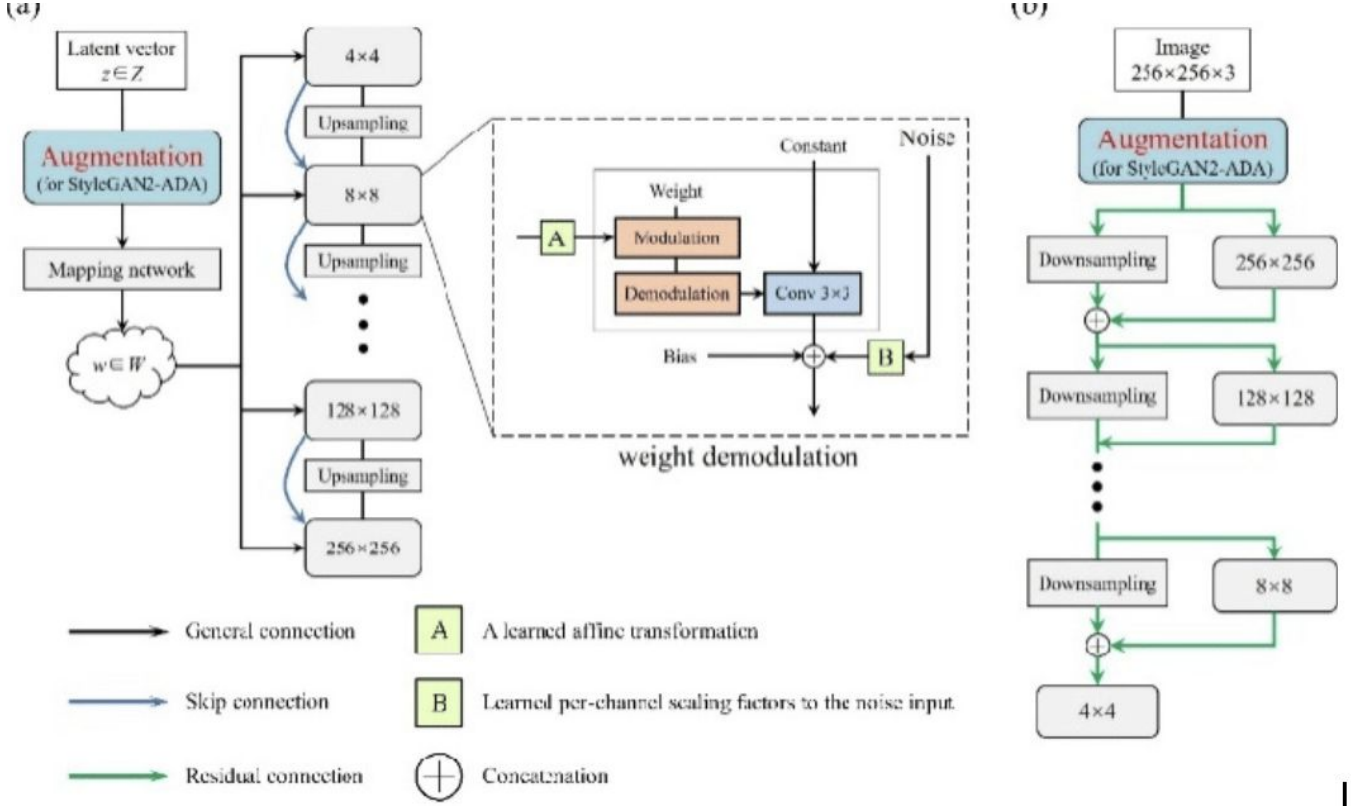
Generator = trains on image dataset (in my case MURA dataset) and creates increasingly more realistic fake images that resemble images from the training dataset and feeds them to the discriminator.

Discriminator = is fed synthetic images and original real images and must figure out which is real and which is fake. When the discriminator can no longer detect a synthetic image as fake, the synthetic images are useable.

ADA = adaptive Discriminator augmentation - the discriminator for styleGAN2-ADA which stabilizes training with limited data regimes - meaning that the number of training images needed for the generator is much less. (good for smaller training datasets)

StyleGAN2 specifically was created by nvidia and is pretrained on faces. It can be trained on images in a custom dataset (like the MURA dataset) that are not faces and molds to the contents of the images you use to train it.

StyleGAN2-ADA architecture



MURA Database (database used for CNN and StyleGAN2-ADA models)

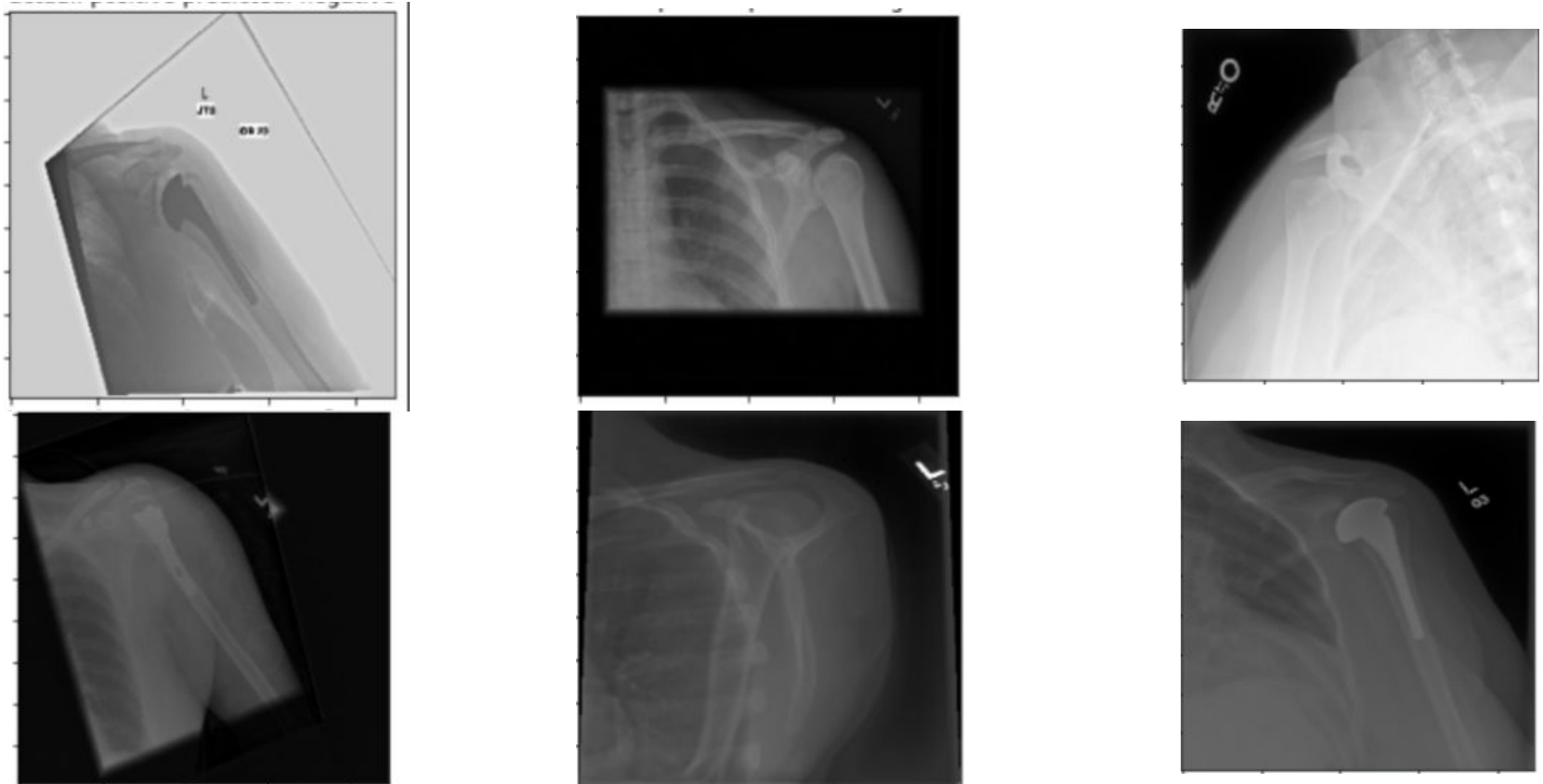
MURA is a database made by the Stanford Machine Learning Group that consists of x-ray images of Shoulders, Wrists, Elbows, Hands, Fingers, Forearms, and humerus. (I used only the shoulder x-rays)

They are labeled as positive for having abnormalities in the bone structure or negative for being normal.

The dataset is large however still challenging to use due to the inconsistencies of images and amount of variation across images

Specs: 14,863 studies from 12,173 patients, with a total of 40,561 multi-view radiographic images

View of some images in MURA dataset (notice variation and inconsistent formatting)



Implementing StyleGAN2-ADA-Pytorch

To do this in google colab, you need to have python version 3.9 and not the new updated python 3.10 that google colab has adopted. Google colab allows you to use the fallback python 3.9 still currently if desired which is available until mid-may.

To use it, I had to set up my training data from the MURA database as two seperate pandas dataframes - one with all positive training x-ray images of shoulders and one with all negative x-ray training images of shoulders.

In total there are around 8000 training images for shoulders but when split into positive and negative, there are less of each seperately

Implementing StyleGAN2-ADA-Pytorch pt2

The reason for splitting into positive and negative dataset data frames is to train styleGAN2-ADA once on only positive x-ray images and once on only negative x-ray images so the I knew which resulting grid of fake images produced from each was positive and which was negative (as I am not a doctor and cannot decipher that myself). The dataframes look like this:

	filename	class
19	MURA-v1.1/train/XR_SHOULDER/patient00007/study...	negative
20	MURA-v1.1/train/XR_SHOULDER/patient00007/study...	negative
21	MURA-v1.1/train/XR_SHOULDER/patient00007/study...	negative
22	MURA-v1.1/train/XR_SHOULDER/patient00007/study...	negative
155	MURA-v1.1/train/XR_SHOULDER/patient00051/study...	negative

Implementation of StyleGAN2-ADA-Pytorch continued

Then from the dataframes, you take each image path and copy the image with a new name into a folder structure that works with StyleGAN2-ADA.

- The folder structure should start as “data” with 3 sub folders “images” “datasets” and “experiments”.
  - The images are copied and renamed into the “images” subfolder.
  - The datasets subfolder has the data paths
  - the experiments subfolder is meant for holding the created synthetic data images and other model information.

StyleGAN2-ADA-Pytorch implementation

Each image needs to be resized to (256, 256) and have 3 color channels (RGB) to be fed into the GAN model.

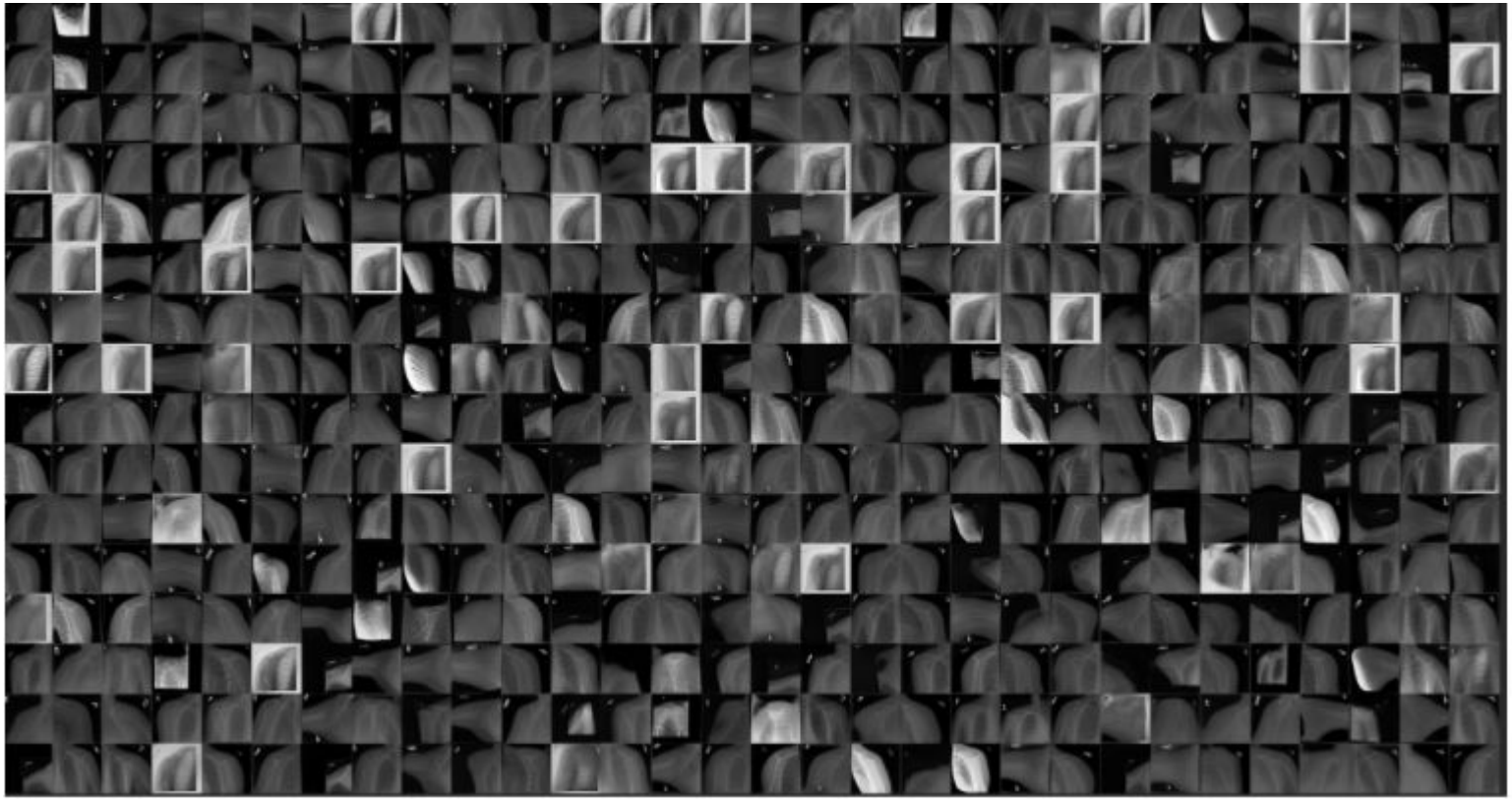
The model then works by using the datasets folder of the images that are properly sized and have 3 channels and are compatible with Pytorch.

You need to specify “SNAPS” for this model which is like taking a “snapshot” of the information the model has and the place in training it is. If SNAPS = 10 then the model will save all information and its place in training every 10 images. Snapshots take a lot of time however it is important to have this as the information saved/recorded is what you use to resume training where you left off if google colab stops the training.

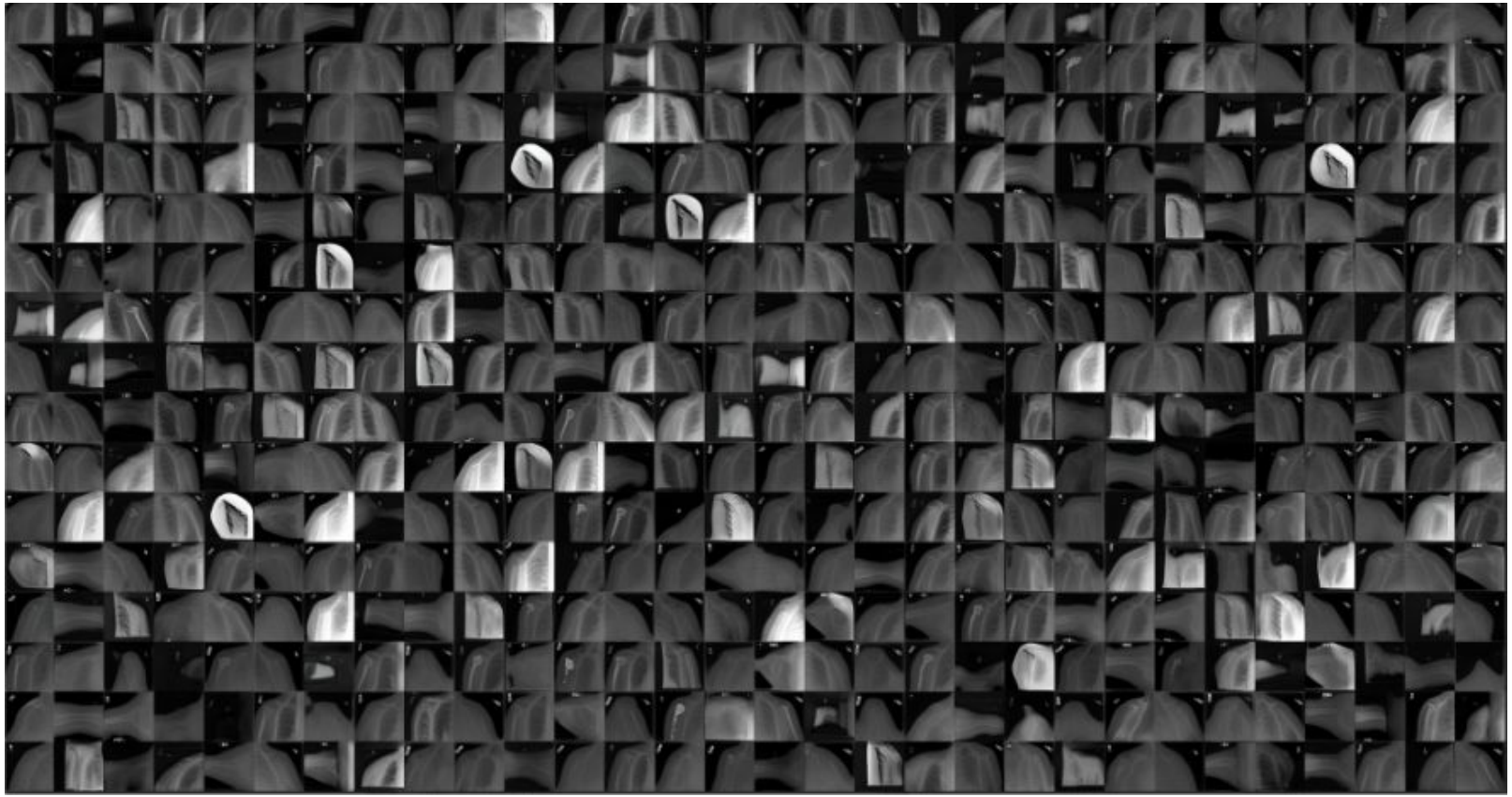
GRIDS OF SYNTHETIC IMAGES

The final grids of the synthetic images that I used from StyleGAN2-ADA training look like this..

Negative Shoulder synthetic images grid

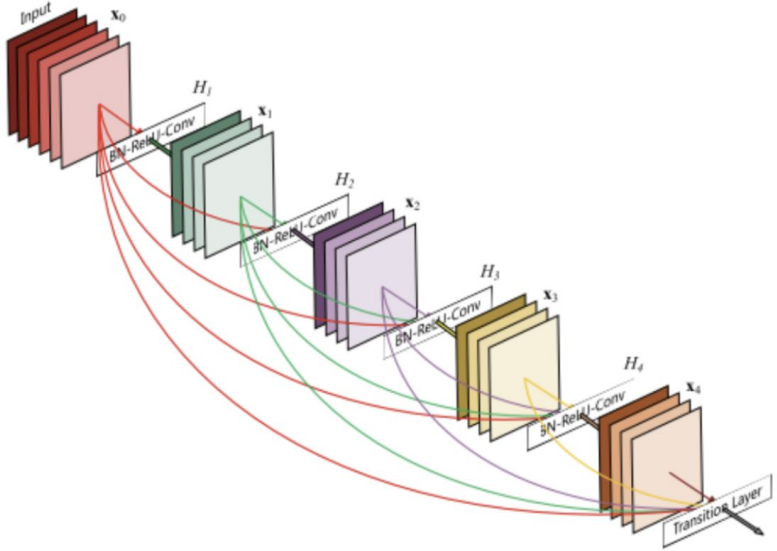


Positive Shoulder Synthetic images grid



Old CNN with densenet169 architecture

The CNN that I made last semester with densenet169 architecture pre-trained on imageNet that I am trying to improve the accuracy of has an architecture as follows:



Why densenet169 is good

When researching, I saw many that used the MURA dataset using this densenet 169 architecture.

One of the great things about this CNN architecture is that it handles the vanishing gradient problem well and is pretrained on imagenet so it is more capable of understanding new datasets that it is trained on with a bit less data.

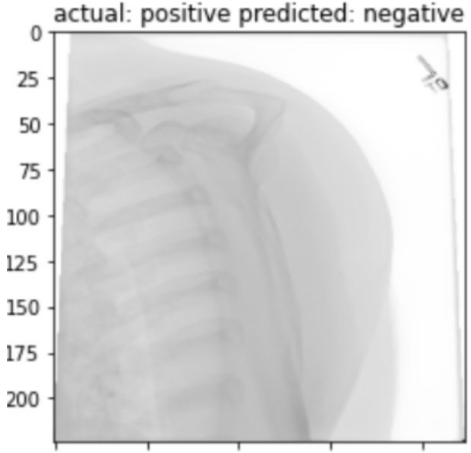
The issue with this CNN is that still, the 8000 shoulder x-ray images in the MURA dataset have so much variation with not enough data to get high accuracy from training the model and testing it on the test data of MURA.

Accuracy of densenet169 CNN model before additional Synthetic training data

The training accuracy of the CNN was about 83%

The testing accuracy of the CNN was about 67%

- specifically noticeable that the CNN failed to recognize/predict a positive testing x-ray image and tended towards always predicting an x-ray image as negative for having no abnormalities.



Adding the Synthetic images to old CNN

To add the synthetic images each image in the grid of synthetic images for both positive and negative x-rays needed to be cropped, sharpened, and resaved.

Each image in the grid was already 256x256 so it was easy to run through the grid and get each image and save it as its own individual image.

To do this, i used the Image import from PIL (pillow) as it is very easy to use to manipulate images and re-save them.

Saved under new subfolder in data folder called Augmented_Fakes with corresponding positive and negative subsub folders.

Then the new images were appended onto the dataframe of all the training images which were then resized to 224x224 before being fed into the CNN model.

Accuracy Results of adding new Synthetic data to CNN

The training set accuracy increased from 83% to 88.04%

The testing set accuracy decreased from 67% to 64.42%

- The model still tended to classify positive x-ray images as negative like before

Why?

- This is likely due to the synthetic data images not being realistic enough as training was cut short. The training got better but picked up bad information from the new data that caused less accurate testing.
- Also likely due to needing more than just 960 total new synthetic images in the training dataset it increase the data to a sufficient size for the amount of variation. It would likely be best to have at least 10,000 training images or more.

In the Future…

To increase accuracy of the CNN model there are many things that could be improved:

1. Train styleGAN2-ADA fully for more realistic synthetic data
2. Increase amount of synthetic data made to increase training dataset for CNN to have 10,000+ training images
3. Do more data augmentation on the training and testing datasets to eliminate poorly formatted images that cause extra variation and to make important information in images more prominent (cropping, hue enhancement, sharpening, etc)
4. Trying different CNN architectures such as ResNet for summation of feature maps instead of concatenation.