Aileen Dugan

Neural Networks Final Project

Part 4: Final Solution

For the final solution, I took the images generated and created copies such that each image in the grids of images was cropped to be its own image, sharpened by a factor of 2, and then saved to a new folder either positive/SHOULDER or negative/SHOULDER to keep track of the positive new images and negative new images within the folder schema /content/drive/MyDrive/data/Augmented_Fakes. Then, each image was added to the pandas training dataframe by their new filename paths and given their corresponding label of "positive" or "negative" as their class in the dataframe. Then the training data frame was turned into the proper ndarrays format for x_train being the images from the filenames in the dataframe and y_train being their corresponding positive/negative class names for their labels. Then each image was resized when the images in the rest of the MURA database were resized as well to fit the proper format for the densenet169 CNN used. Tse synthetic images were added to the training data used to train my CNN from last semester to see if a higher accuracy could be achieved. My assumption was that they would not increase the accuracy just because some of the generated images are not that great. Especially the positive (abnormal) synthetic images as I stopped the StyleGAN2-ADA training for the positive images early in order to finish this project in time. In the future, I would of course finish the training for StyleGAN2-ADA in order to get better synthetic images to add to the training data. I also think that in the future I would change from densenet169 architecture to a ResNet architecture for my CNN to see if that performs better. The architecture of the CNN I made and all other information about what I did for densenet169 architecture can be found in my computervision xray project on github. Therefore, I will not re-explain it here but rather provide a link to the github repo that contains all this information(https://github.com/aileendugan/CompVisionBoneXrayProject).

I will, however, explain why I think ResNet would be a good change. ResNet, as opposed to any densenet architecture, adopts a summation of all the proceeding feature maps while densenet concatenates them. The reason that I think this switch to summing the feature maps would be useful is because of all of the unimportant variation in the data. The MURA database, as explained before, has many images where the x-ray is not fit to the entire image and blank space is left. Also, there are many weird things such as some are rotated and some are brighter than others. While this should be fixed with augmenting the data images (another thing to implement/fix in the future), it also seems that summing up the feature maps could help put more importance on the important and consistent features rather than concatenating all of the feature maps and therefore not releasing any unimportant information picked up that could cause more confusion. Most of the research papers that I looked at of others using the MURA database did use densenet architectures such as densenet169 like I did or densenet121 which is why I originally chose densenet169. However, without augmenting the data or cleaning it up as much as I should, it seems that ResNet could make a slight difference in a higher accuracy and seems worth trying. Besides switching to ResNet, I could, as mentioned before, try to fix all of the

MURA database images through augmentation. The only issue with trying to do this is that not every image in the database needs to be fixed or fixed in the same way and for shoulder x-rays alone there are over 8000 images. Therefore, manually fixing each would be a hassle and I cannot think of a great way to speed up that process. One easy thing to do would be to normalize the coloration, saturation, and hues of each image. I tried last semester to change each x-ray image to grayscale but that did not really do anything but since some of the images are much brighter than others, changing/normalizing the hues across all images could be helpful. This would also likely be helpful for making the synthetic data using the StyleGAN2-ADA.

One of the great things about StyleGAN2-ADA or any StyleGAN is that it is an addition to the GAN architecture that introduces significant modifications to the generator model. GANs or Generative Adversarial Networks have both a generator and a discriminator. The generator takes simple random variables as inputs and generates new data aka the new synthetic images. The discriminator then takes original data and the new generated data and tries to discriminate them, aka tell which one is the synthetic data image, building a classifier. So, the generator tries to learn and train and make more and more realistic fake data to trick the discriminator until the discriminator can no longer correctly pick which data is synthetic and which is real. The ADA addition of StyleGAN2 is an adaptive discriminator augmentation mechanism that significantly stabilizes training in limited data regimes.This makes it helpful for creating synthetic data when training on smaller training datasets. The approach does not require changes to loss functions or network architectures.

For my accuracy results with adding the synthetic data images to the training data for my CNN, I got a 0.8804 or 88.04% accuracy on the training data and 0.6412 or 64.12% accuracy on the testing. Ofcourse, the accuracy from training to testing goes down as to be expected but the same problem for testing is coming up that came up for my CNN results last semester. As shown in figure 2, the model tends to classify x-rays that are actually positive for abnormalities as negative or normal which is the same issue as before. The accuracy is 64% I think mainly because the model classifies the negative x-ray images properly as being negative but cannot detect positive x-rays well at all even with the additional synthetic data. The results can be seen in figures 1 and 2 below.

```
epochs = 5
print('-TRAINING----------------------------')
print('Input shape:', x_train.shape)
print('Number of training images: ', x_train.shape[0])

model.fit(x=x_train, y=y_train_enc, epochs=epochs)
```

```
-TRAINING----------------------------
Input shape: (9339, 224, 224, 3)
Number of training images:  9339
Epoch 1/5
292/292 [==============================] - 471s 2s/step - loss: 1.1942 - accuracy: 0.6553
Epoch 2/5
292/292 [==============================] - 463s 2s/step - loss: 0.6656 - accuracy: 0.7756
Epoch 3/5
292/292 [==============================] - 462s 2s/step - loss: 0.6360 - accuracy: 0.7911
Epoch 4/5
292/292 [==============================] - 468s 2s/step - loss: 0.4749 - accuracy: 0.8396
Epoch 5/5
292/292 [==============================] - 460s 2s/step - loss: 0.3170 - accuracy: 0.8804
<keras.callbacks.History at 0x7f08d7d3fcd0>
```

Figure 1. Training Accuracy of CNN with synthetic data added to training data

```
-TESTING----------------------------
Number of test images: 563
18/18 [==============================] - 30s 2s/step - loss: 1.4751 - accuracy: 0.6412
Test loss: 1.4751086235046387
Test accuracy: 0.6412078142166138
1/1 [==============================] - 3s 3s/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 123ms/step
1/1 [==============================] - 0s 122ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 127ms/step
1/1 [==============================] - 0s 125ms/step
1/1 [==============================] - 0s 130ms/step
```

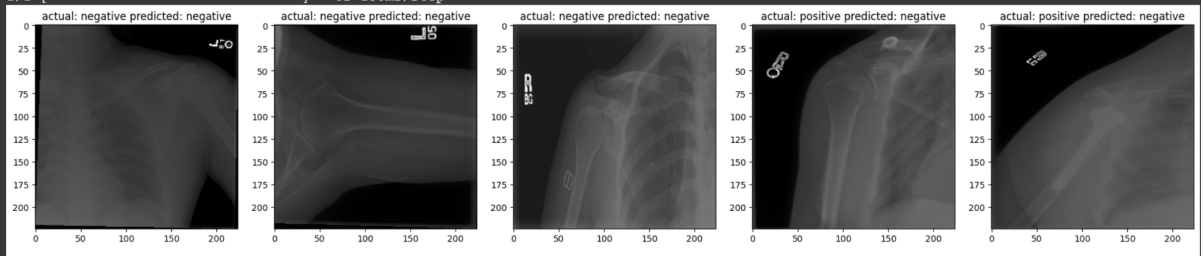

Figure 2. Testing Accuracy of CNN with synthetic data added to training data

When comparing the training and testing accuracy of the CNN with the additional synthetic data to the training and testing accuracy of the CNN from before without the additional synthetic data, the training accuracy improved from about 83% to 88.04% but the testing accuracy decreased from about 67% accuracy to 64.42% accuracy. What I believe caused this is the fact that I trained the CNN on synthetic images that are not that great or "real" looking and so the training accuracy went up because the model looked at those images and made it work however the testing accuracy went down because the model during training took into consideration things about the synthetic images that would never be found in the real images in

the test set so the model actually understood less about negative vs positive x-rays from the synthetic images.

It is important to note that my synthetic data is not great as you can see from the figures as part of the part 3 write up. The negative synthetic images are okay but some of the images have curved bones that are not normal and thus obviously fake and do not work as part of the negative (normal) x-ray images and the positive synthetic images are not great because I had to stop the training for them after 13 hours before it was fully complete in order to move on and finish this project. The positive (abnormal) synthetic x-ray images are all a little bit blurrier than desired and do not have clearly pictured bones even compared to the quality of the negative synthetic x-ray images. This likely was the reason for the accuracy of the CNN model not getting better from last year because these new synthetic images are not up to par with the real images from the database. I assume that if I trained the StyleGAN2-ADA to completion for both positive and negative x-ray images that this would not be the case and that the accuracy would get better as long as the final synthetic images that fake out the discriminator are more realistic the way I would expect them to be. Also, it is important to note that the number of training images for shoulder x-rays in the MURA database is around 8000 and with the addition of 480 positive synthetic x-ray images and 480 negative synthetic x-ray images of shoulders created and added to the training data, there becomes about 9330 training images. It is great that the training data was increased by 960 x-ray images however, it is likely that generating more than this to possibly double the training data instead would be beneficial for increasing the accuracy of the CNN.