My chosen classifier was for a CNN. I used DenseNet169 trained on Imagenet and trained on my x-ray shoulder training images from MURA(set up as described in previous part). The model I used was sequential and compiled with Adam optimizer and binary cross entropy loss function. I chose this because a CNN is necessary for such a detailed image as x-rays and used DenseNet169 in this fashion because it was suggested by all MURA dataset users and when tried, gave better accuracy on the training data than any of my manually made CNN model setups. This is likely due to the fact that the images had to stay in a 3 BGR channels instead of going to grayscale, and denseNet169 eliminates the vanishing gradient problem which is how a model steps through the process of changing pixels to find its correlations. A vanishing gradient seems to be a major problem with x-ray data especially when the MURA dataset images are not all consistent enough in their formatting where certain things should be ignored.

While the training accuracy with DenseNet169 was around 83%, the test accuracy was about 67%. However, with the manually made CNN, the testing and training accuracy stayed about the same at 50%. The results can be seen below in figures 1 and 2.

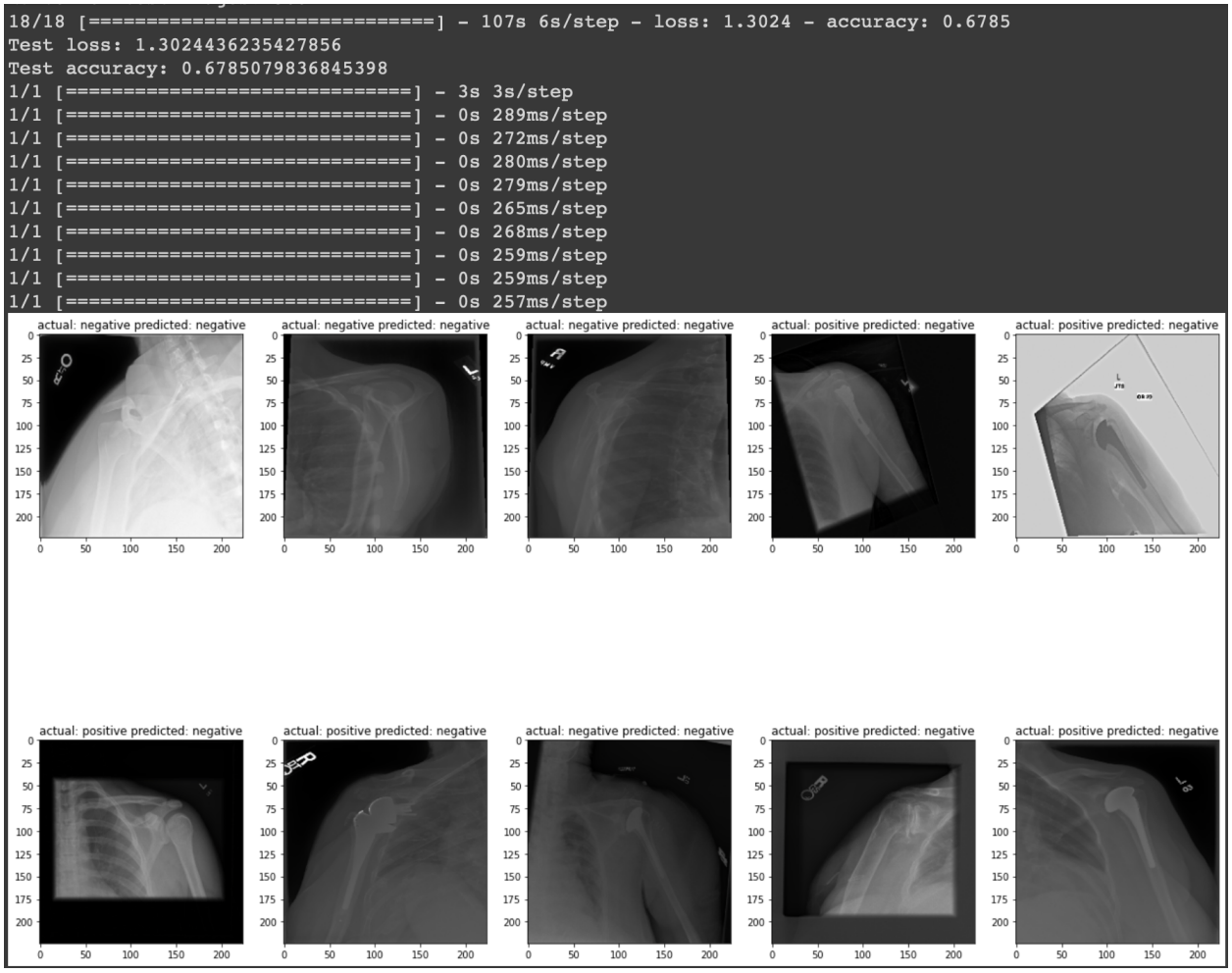


Figure 1: DenseNet169 testing data accuracy and example images with actual and predicted labels.

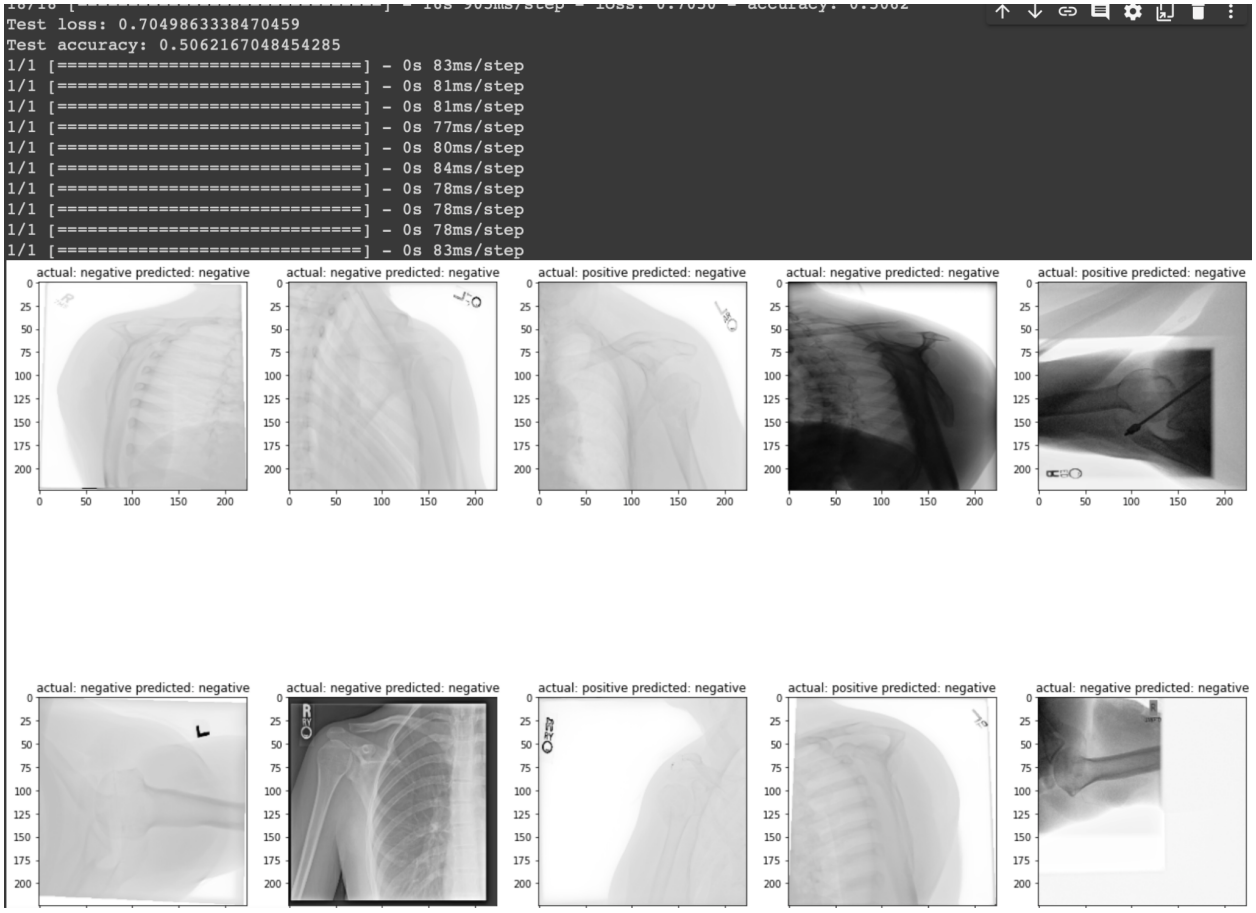

Figure 2: Manually made CNN model testing results with actual and predicted labels.

As you can see, the model seems to struggle to find positive testing results and generally tends to predict all testing images as negative in both approaches. This is likely due to the fact that either the images are not consistent enough, or the model itself is not detail oriented enough to pick up on what a positive result would look like.. Also, due to most bones being different per person, iit tends to get tricky to tell even as a regular human, which x-rays are abnormal or have broken bones especially in such a complicated bone structure such as a shoulder. I noticed that most people that used DenseNet also used the Keras ImageDataGenerator.flow_from_dataframe to create generators for the training and testing data instead of the ndarrays that I used. I think this might be important as it breaks up each image a little bit more. ImageDataGenerator class ensures that the model receives new variations of the image at each epoch and only returns the transformed images. I think this is useful because the dataset also, when only using one body part''s x-rays is a smaller dataset and thus needs more variation in the data. If I were to do this, then likely, the testing would become a little bit smarter and possibly more capable of predicting negative images.