COSC 499: Peer Testing #1 Report

Medical Image Matching (1)

Marieke Gutter-Spence, Emily Medema, Alex Rogov, Niklas Tecklenburg

Activities Done this Week:

- Restructuring of the git repo file structure.
    - Split into training and testing folders.
    - Added folder per model to store test scripts.
- Setting up of the environment on BC Cancer GPU so our model can run.
- Added additional unit tests to scripts.
- CNN model is running, optimization is needed.
- `preprocess_data.py` has been implemented and tested.
- Images are separated into folders based on mammogram type.

Issues/Problems Discovered:

1. Our test files are unorganized: separate model_unit_test into test folders.
2. Mammograms are not separated by type.
3. Function in the Prediction component still needs to be completed.
4. The script that will resize the image and structure it in a way that is ready for use in the model/augmentation is missing. We can also have it separate the images into their own mammogram type.
5. Packages were not installed on the remote server
6. The area of interest in mammograms is only about 50% of the image. The other half of the image is just black, this may make the input size of the model bigger than needed (more parameters and slower training time)

Solutions to Issues/Problems:

1. Reorganized our file, specifically: separate model_unit_test into test folders.

2. Create a file or folder to store all the image files into their own mammogram type. Or if all images are added to one file, then create that file.

3. In an exchange with the client, we decided to prioritize the building and training of the model first, before coming up with a deliverable function to return a proper similarity score. For now, we will work with the distance of the embeddings, to evaluate the performance of our model. Therefore we have a simple function in place, which can be expanded to the desired similarity score, later in the project.

4. `preprocess_data.py` needs to be implemented, this is the script that will resize the image and structure it in a way that it is ready for use in the model/augmentation. We can also have it separate the images into their own mammogram type.

5. Used a virtual environment and installed the packages needed.

6. As of now, we have not experienced any issues with the training times, but we are aware that this problem could come up. Therefore we are experimenting in parallel, with some detection stuff to properly get the relevant areas of the image, to be prepared once the issue occurs.

Testing:

Additional Unit Tests to Create:

Preprocessing:

1. `data_cleaning.py`
    a. Add testing to ensure that we are reading the correct files.
    b. Add testing to ensure that we are getting the output we are expecting when we read the files.
    c. Ensuring that the split between non-numeric and numeric works.
    d. Using dummy data, ensure that the missing values are found and dealt with.
2. `preprocess_data.py`
    a. Once this is completed, we will need to implement testing on the resizing of images, getting the images, getting the right images, and saving the now processed images.

src:

1. `models/cnn_tripletloss`
    a. `buildDataSet` - test that it returns a list of length 10 containing the type of values that it should be.
    b. `get_batch_random` - test that it creates a batch of APN triplets with a completely random strategy
    c. `build_network` - make sure that this method returns a neural net.
    d. `TripletLossLayer` - make sure that when a TripletLossLayer object is made, it has the correct parameters and type.
        i. `triplet_loss` - test that this returns the value and types that we expect.
        ii. `call` - test that this method returns what we expect.
2. `training/data_augmentation`
    a. Test the augmentation methods.
3. `training/models/ORB.py`
    a. `orb_sim` - ensure that the method returns what we expect based on some dummy data.
4. `training/models/SSIM.py`
    a. `get_sim` - ensure that this method returns a value that makes sense in accordance with the dummy data and that it returns the right type of data.
5. `training/models/siamese.py`
    a. `make_pairs` - ensure that this method returns the values we expect based on dummy data and that it returns the right type of data.
    b. `euclidean_distance` - ensure that the result makes sense in accordance with the dummy data.

    c. `loss` - ensure that the return type is what we expect.
6. `training/models/siamese2.py`
    a. `preprocess_image` - test that it returns the file type we expect.
    b. `preprocess_triplets` - ensure that the method returns what we expect.
    c. `euclid_distance` - ensure that the return type is what we expect and that based on the dummy data we pass in the calculation is correct.
    d. `DistanceLayer/call` - ensure that the return type is what is expected.
    e. `SiameseModel`
        i. Ensure that all methods and attributes associated with this class return the type we expect and if applicable perform as expected with dummy data.
    f. `cosine_similarity` - ensure that the return type is what we expect and that based on the dummy data we pass in the calculation is correct.

All of the mentioned unit tests have been implemented as of December 3rd.

Discussion with Medical Image Matching (0):

During the peer testing session, we had a discussion with the other team, in which each team detailed their progress and then answered any questions the other team had. MIM team 0 started off the discussion by giving our team an overview of their dataflow and steps taken for their project so far. They led us through their thinking process and explained the decision to wait to implement any models until further research was done. They feel that there are a lot of unknowns that are tough to account for at this point which may inhibit the accuracy of the model. Instead they have been focusing on curating their database, populating their csv(s) with details of the database, and connecting their csvs and images with opencv. They also have their unit tests running. We questioned them on how the database is arranged, how the CSV file would look, and how were they accounting for different dimension sizes?

Our team began the discussion by describing our progress through the diagrams Niklas created. It was expressed that these graphics were very helpful in understanding our progress so far. We then went over the pre-processing sections, as well as how the augmentation of images works and which libraries were used. We also spoke about our models, which ones we have as ‘backups’, and the connections we still need to make within our project. Lastly, we went over the list of tests that still need to be implemented.We were asked questions regarding our CNN model (ie., why we were training only one part), if we have been able to curate datasets from the GPU, and if we had been performing unit testing. This discussion gave both of our teams a good understanding of the work we have accomplished so far and the direction in which our projects are headed for the next semester.

Following the in-class peer review discussion, our team met over zoom with Gema in order to demonstrate our progress so far. Niklas showed the graphics he made in order to describe the work we have been doing. Emily, then clarified the achievements we have made in comparison to what we had projected to have finished from the first milestone. You can find the graphics as

well as the planned completion chart described below. This meeting helped us to see that we are on track with progress for this semester.

Completion of Planned Peer Testing #1 Features

During our Requirements milestone we planned to have completed the implementation of the preprocessing; the organization and structure of our repo, models, and storage; building of the training test dataset; the implementation and beginnings of the optimization of the first model and some basic models; and the testing of the preprocessing and model (See Figure 2).

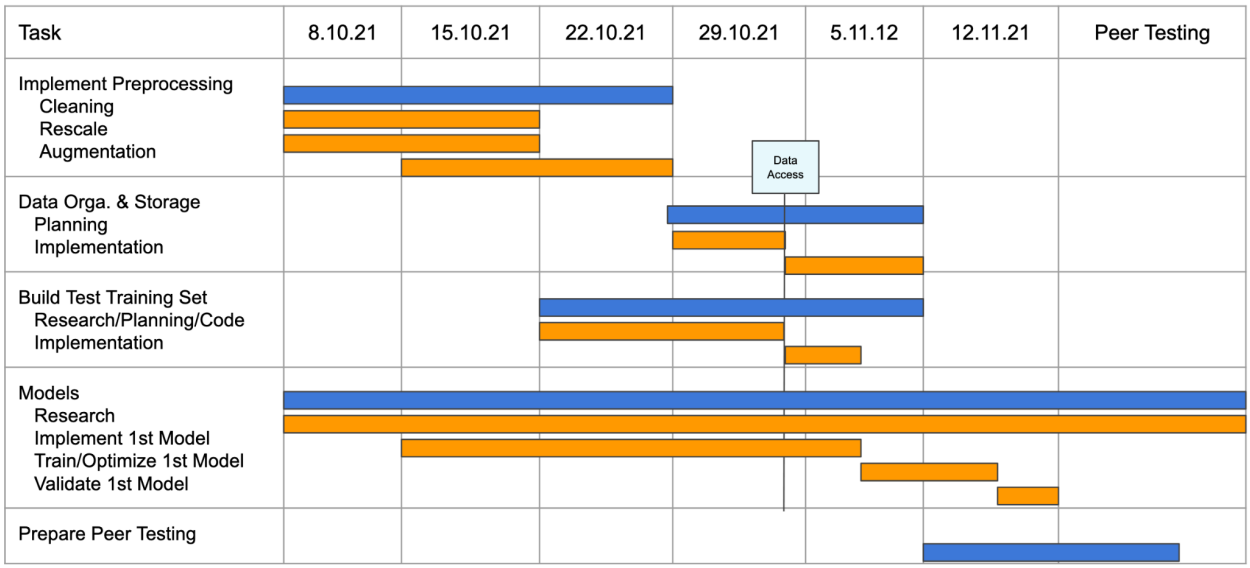


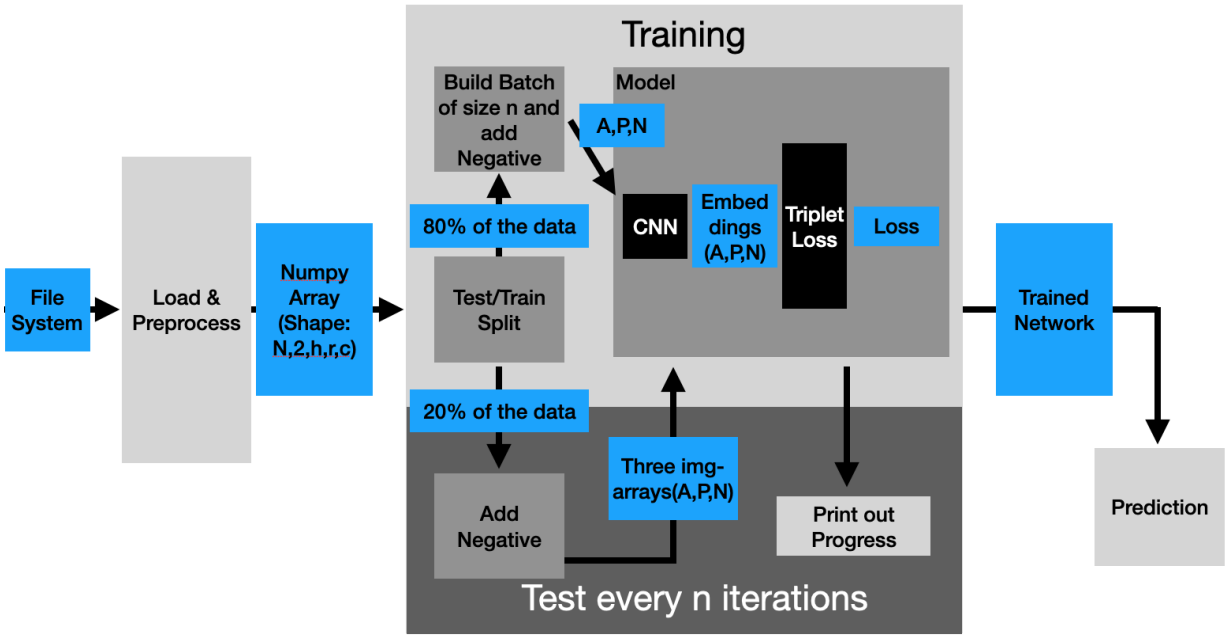
Figure 2

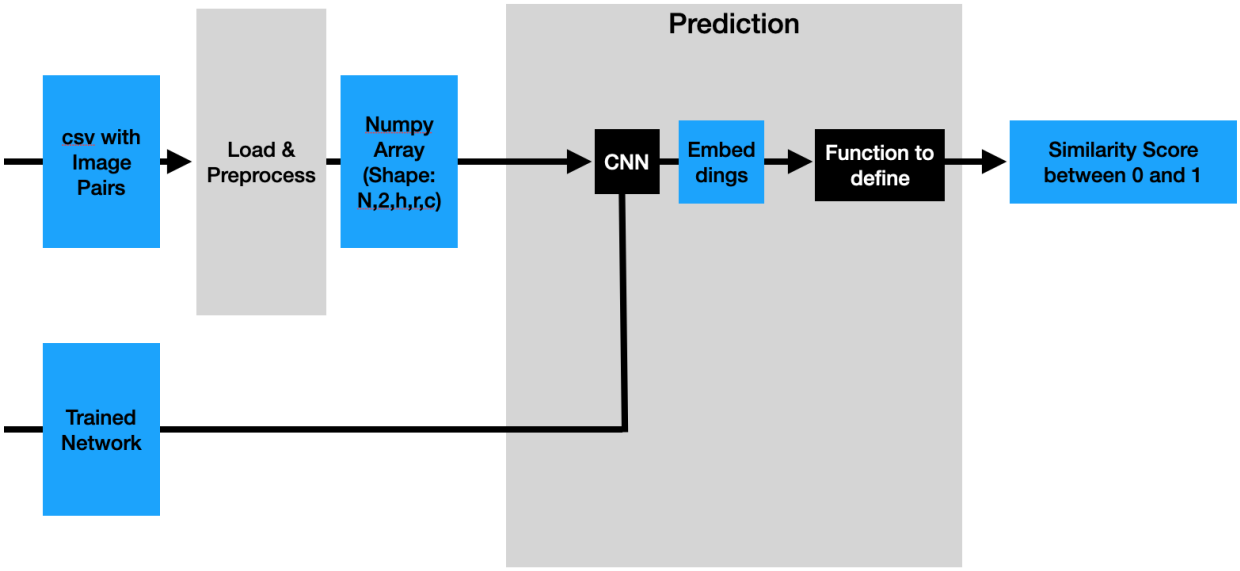
We have completed all of the above and are prepared to move on to the Peer Testing #2 features, which are primarily optimizing and continuing development on the model(s). For more details, see Table 1.

Task	**Completion**	**Details**
Implement Preprocessing	Completed	Preprocessing has been implemented, we now clean the data, preprocess it (resize and save as an numpy array in a specific location based on the mammogram type), and augment the images if necessary. We are now ready to optimize it if necessary which is a goal for Peer Testing #2.

Data Org. & Storage	Completed	We planned how we wanted to implement and organize our structures for files and data. We are now ready to focus solely on the model.
Build Test Training Set	Completed	We have built up the training and testing sets with data augmentation. Our current split is 60-40.
Models	Completed	Our research is completed and we have a few models implemented and running that we can now focus on optimizing for Peer Testing #2.
Prepare Peer Testing	Completed	We have discussed with the other team, met with the Professor in lieu of a video, and have completed the report.

Table 1




Prediction
csv with Image Pairs
Load & Preprocess
Numpy Array (Shape: N,2,h,r,c)
CNN
Embed dings
Function to define
Similarity Score between 0 and 1
Trained Network