Goals and discussion (16 points)

Essential goals

List all essential goals that you proposed either in your second proposal or project update. For each, (a) include one or two sentences on how you completed it, what progress you made, or why you abandoned it, and (b) include one to two sentences on what was the most interesting or difficult part.

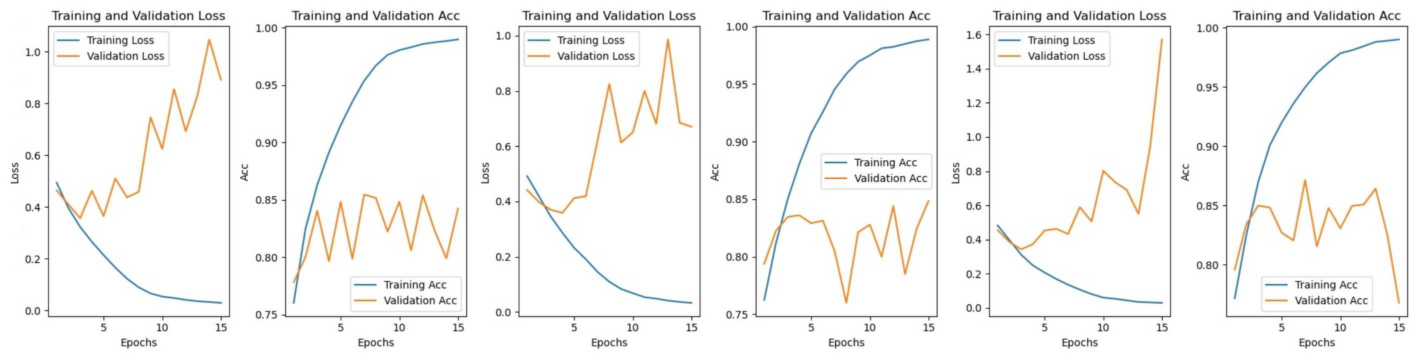
- We will design a CNN and use it to generate feature vectors for the x-ray scans.

  We used Pytorch's Conv-2D to design a CNN that has 4 2D-convolutional layers, and a kernel size of 3 for all layers. The initial channel size can be specified using a keyword argument with the channel size being doubled 3 times leading to a final channel size of 8 * initial_channel_size. The most interesting part was understanding how the CNN learns information from the image at each layer as this enabled us to know what output size we expected at each layer. This was particularly useful as the output of the final conv-2D layer served as input to the rest of our models.

- We will then vary the output network between an RNN, LSTM and a fully connected feedforward neural network and benchmark the performances of all of them.

  We used Pytorch's RNNCell, LSTMCell and Linear respectively to implement our 3 output models. For the RNN and LSTM, we used the cell implementations as we wanted to represent a single time step of the input as the sequential pixels and then concatenate the output of all the cells to make a prediction. We benchmarked the performance of the three models by training each set on 75000 data points and validating on 7500 data points. The most interesting part was understanding how the RNN and LSTM actually work and the intricate differences between both. As per theory, the LSTM outperformed the RNN.

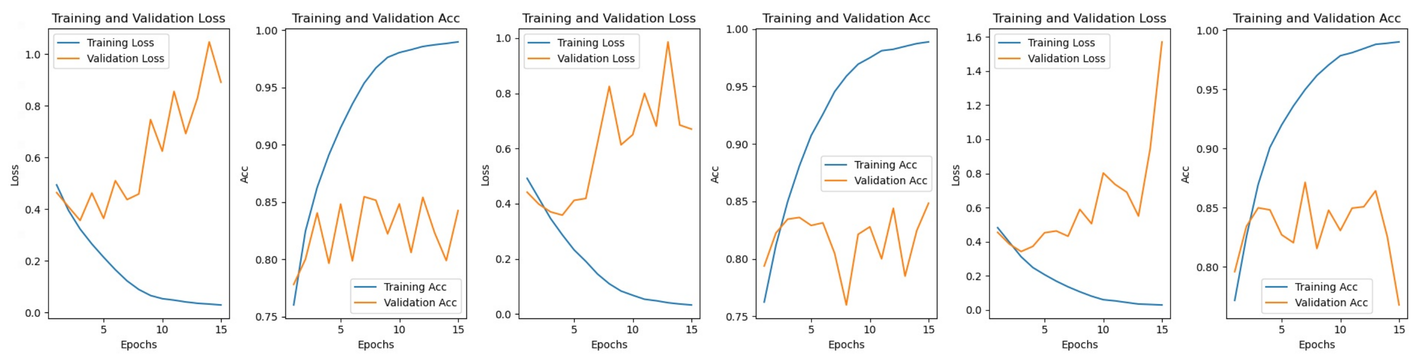
Images of loss / accuracy curves for FFNN, RNN, and LSTM respectivly. (Given large dataset and the smaller models we think the model learned the dataset within the first few epochs, leading to the validation curves to be the way they are)



- Achieve at least 85% accuracy with one of our models

We then computed and plotted the average best accuracies for the CNN-FC, CNN-RNN and CNN-LSTM models with accuracies ranging from 83.7% - 87.1%. The interesting and difficult part was hypothesizing the subtle differences between the performances of the three models. For example, we initially thought that the RNN would outperform the fully connected network but this was not the case and we came to the conclusion that the vanishing gradient problem of the RNN factored into this.

Plots for best accuricies for FFNN, RNN, and LSTM respectivly



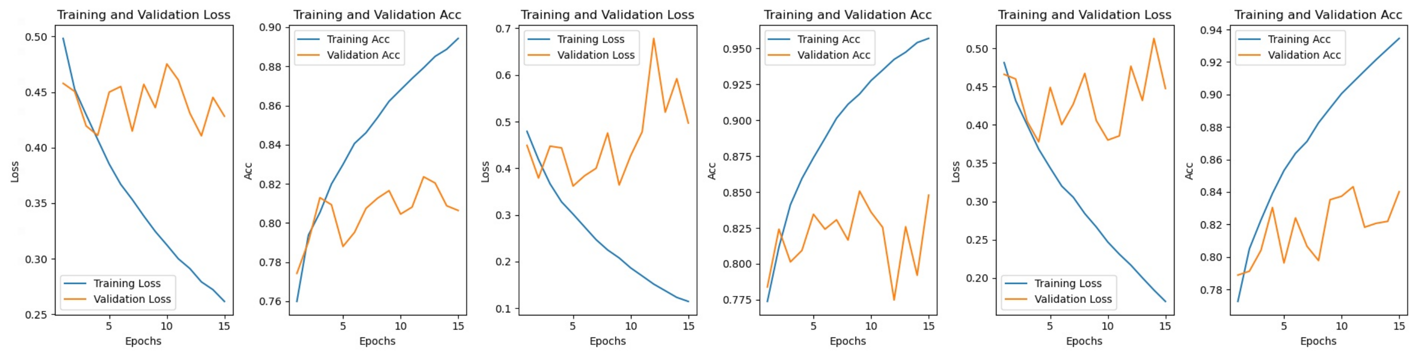
Desired goals

List all desired goals that you proposed either in your second proposal or project update. For each, (a) include one or two sentences on how you completed it, what progress you made, or why you abandoned it, and (b) include one to two sentences on what was the most interesting or difficult part.

- We will conduct a hyperparameter search for the best performing architecture combinations and see how changes in these parameters affect the performance, with the intention of improving performance even more.

  For our hyperparameter search, we varied three parameters:initial channel size, learning rate and dropout, that we identified as having the most impact on our train and validation loss and accuracy. For each parameter that we varied, we plotted the losses and accuracies and compared the performance across all. The most interesting part was that the hyperparameters that worked one for one architecture seemed to work well for the other architectures as well. The difficult part was inferring what the effect of changing multiple hyperparameters while only having the resources to change one at a time.

Plots show performance of LSTM with best settigns for dropout, learning rate, and initial channel size respectivly.



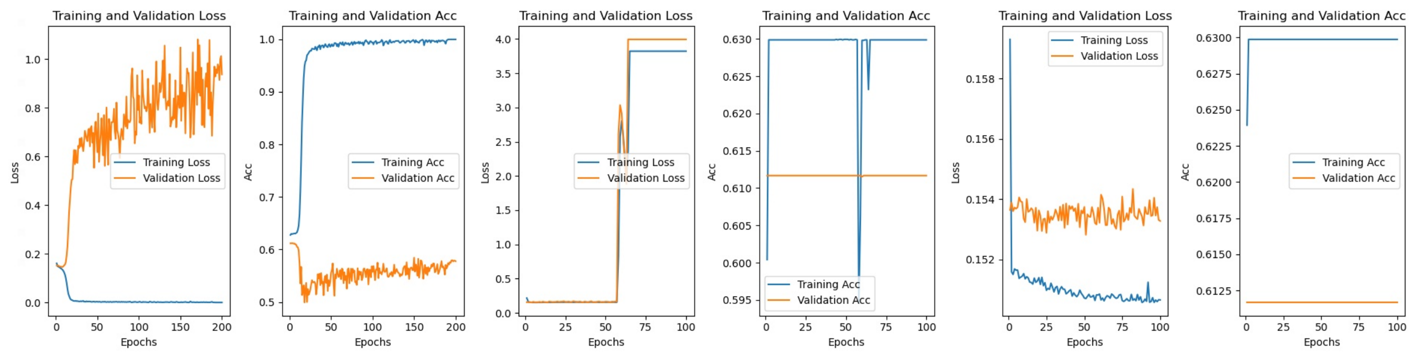
- Achieve at least 90% accuracy with one of our models

  After identifying the best hyperparameters for the binary classification task : (batch_size = 256, initial_channel_size = 80, lr=1e-4 ,epochs=15, dropout = 0.3), we trained and validated our models with these hyperparameters. However, we were not able to meet our goal of at least 90% accuracy. Our best performing model was the CNN-LSTM which got to a highest validation accuracy of 87.13%. The most difficult part while trying to achieve a 90% accuracy was while doing the hyperparameter search as discussed above. Additionally, the team hypothesized that a larger model would be able to achieve a higher accuracy, however given lack of compute power and time we were not able to test this theory.

- We will test our architecture on a multi-class classification task to see whether it yields the same results.

  We then trained our CNN-FC, CNN-RNN, CNN-LSTM best performing architectures on the NIH-Chest-Xray-dataset that has 15 possible classes. While we were able to modify our existing architecture to perform computation on the new dataset, we were only able to train a zero rate classifier with the exception of the FFNN which was able to memorize the dataset. However it was not able to perform better than a zero rate classifier. We think this is due to the complexity of the problem. The dataset is a multilabel, highly skewed dataset with high resolution images. While we were able to decrease the image resolution from 1024x1024 to 256x256 the other complications remained. We were able to train a model to memorize ~500 training examples over the course of 200 epochs. Despite this, we believe that our model did not have enough capacity to learn how to classify this dataset, moreover increasing the number of convolutional layers and increasing initial channel size was still insufficient. It may be the case that the team does not have adequate resources at this time to train a model that can perform well with the NIH-Chest-Xray-dataset.

Shows performance of FFNN, RNN, and LSTM respectivly.



Stretch goals

List all stretch goals that you proposed either in your second proposal or project update. For each, (a) include one or two sentences on how you completed it, what progress you made, or why you abandoned it, and (b) include one to two sentences on what was the most interesting or difficult part.

-Since a ResNet solves the vanishing gradient problem in deep CNNs, we will switch out the CNN for a ResNet (either design our own or use a pre-trained one) to generate the feature vectors.

We abandoned this goal due to lack of applicability. ResNet does indeed solve the vanishing gradient problem, however after spending well over 45 hours of compute time on networks that were not large enough to warrant the use of a skip connection the lack of applicability in our case became clear. In the theoretical case where we are able to go further with our multiclass/multilabel goal, the ResNet could be applicable. However with regards to the work we were able to accomplish, the ResNet approach is not applicable.

- We will then vary the output network the same way we did with the CNN and benchmark the performance.

As discussed above, the limitation on training time did not allow us to implement and train the Resnet-{} architectures. As a result, we could not benchmark the performance.

- Achieve at least 95% accuracy with one of our models (For reference)

  Similar to the above goal, we did not end up designing the Resnet-{} architectures because of the limitation on training time.

Everything else

Discuss the additional work you did as part of this project that was not covered by the goals you set for yourself. Try to organize this into "goals" that you could have set for yourself if you had known what needed to be done. For each of these goals, write one or two sentences on why you focused on it, and one or two sentences on what was the most interesting or difficult part. This can be something practical like "we couldn't load our data/model on Colab" or something conceptual like "we tried to understand the fancy model from this new paper but couldn't figure out how to implement it."

- We had to learn how to use Kaggle for GPU training since the team determined it would be a better fit for our project.

  We focused on this goal because we needed to train our learning model on a large dataset, which was taking too much time on Google Colab. By using Kaggle, we were able to access faster GPU processing, which helped to speed up the training process. The most difficult part was getting used to the new interface and understanding how to navigate the different options available to us there.

- We had to find ways to load large data files onto Google Colab and Kaggle by uploading to Google Drive, as well as making custom datasets on Kaggle.

  We focused on this goal because we had large data files (~8GB) that were difficult to load onto Google Colab. It was not feasible to upload the data everytime we started a new colab/kaggle runtime and as a result, we had to think of ways to work around this. By uploading the files to Google Drive and then connecting to Google Colab, we were able to access the data more easily. The most interesting part was discovering the different methods available for uploading and accessing files on Google Colab, such as mounting Google Drive.

Code and documentation (10 points)

While you should not include code in this report, all your code should be uploaded to your GitHub repository. In this part of your report, you should list all .py or .ipynb files that contain your code and a one or two sentence description of what that file contains. Then, in each of those files, you should make sure that there is enough documentation that we can read through the repository and understand what each part of the code does. This code does not necessarily have to be in immediately working order (e.g., if you cannot upload your data or model files to GitHub), but we should be able to read through your code and notebooks to understand how you implemented things. You absolutely do not need to include a comment for every line of code, but you should include docstrings for (most of) the functions you wrote and descriptions for the coding cells within any .ipynb files.

File	Description
CNN.ipynb	Ran binary classification experiments on the CNN-fully-connected, CNN-RNN and CNN-LSTM
CNN_multiclass.ipynb	Ran multiclass classification experiments on the CNN-fully-connected, CNN-RNN and CNN-LSTM
Pre-process_X-ray.ipynb	For preprocessing the Chest X-ray images for the multi-label classification task

Reflections (6 points)

This section is your chance to reflect on this project. You should write at least a few sentences for each of these questions. However, try to be concise; a longer answer is not necessarily a better answer. If you want to write several paragraphs about something you're excited about, that's great! On the other hand, don't just write several paragraphs listing all the hyperparameter values you tried; if we fall asleep reading your answers, you might lose points.

What was interesting?

What did you learn from this project that wasn't covered by other parts of the class?

- We learned how we could combine CNNs and RNNs to solve complex tasks. We also understood the importance of hyperparameter tuning and how making changes to these parameters can significantly affect the performance of the models. Lastly, exploring the limitations of the models was an eye-opening experience that highlighted the fact that sometimes, despite our best efforts, achieving the desired accuracy is not always possible.

What concepts from the lectures and readings were most relevant to your project?

- We found that the concepts we learned in the lectures and readings on Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) models, and the paper on Batch Normalization were highly relevant. The CNNs helped us to effectively classify images by detecting relevant features, while the RNNs and LSTMs enabled us to model sequential data and make predictions based on previous inputs. Additionally, the use of Batch Normalization helped to improve the overall performance and convergence of the models during training by normalizing the input data. These concepts were integral to our project and helped us to achieve accurate and meaningful results.

What was difficult?

What were the hardest or most frustrating parts of this project? If someone were about to start on a similar project from scratch, what would you encourage them to do differently?

- This project provided valuable insights into practical challenges that arise when working with machine learning models. One of the challenges was dealing with large datasets while having limited computing resources, which required finding creative solutions to optimize the training process. Another challenge was debugging errors in the code, which helped to develop a deeper understanding of the source of the problem and how to solve it. Finally, optimizing the training process to reduce training time was another important skill learned during the project. We would encourage people starting a similar project to begin by building smaller architectures than what they eventually aspire to build. This will enable the team to ramp up quite fast on the technology stack in question as in our case, Pytorch's Conv2D, RNN and LSTM modules. Furthermore, it is easier to debug smaller architectures. Once a smaller architecture is working, they can then shift effort to building a bigger architecture. Furthermore, in the case of large datasets, you should start by testing and ensuring your models work on a smaller sample size. Finally, we would encourage them to regularly save models at the end of each training epoch/training session. When working with colab and kaggle sometimes the runtime environment would run out of memory and crash which meant that any previous work was lost. Regularly saving the models would ensure that the previous work can be restored easily.

What's left to do?

If you were going to spend another month working full-time on this project, what would you try to accomplish? Why? If I gave your group 1,000,000 USD to use for data collection or compute resources, what would you spend it on? Why?

- If given another month to work full-time on the project, we would aim to build bigger architectures, specifically including a ResNet for multiclass classification. This would allow us to find the best combination of architectures that yield the best accuracy, potentially leading to a significant breakthrough in medical X-ray image diagnosis. With 1,000,000 USD for data collection or compute resources, we would focus on optimizing the training process, investing in better hardware, and exploring more advanced deep learning techniques. By doing so, we hope to significantly reduce the training time, allowing us to train larger models and increase the accuracy of their results. Ultimately, these investments would enable the group to make more meaningful contributions to the medical field and improve the accuracy of X-ray image diagnosis for patients.