Configuration Manual

MSc Research Project
Programme Name

Jose Alberto Cruz Sanchez
Student ID: X23191236

School of Computing
National College of Ireland

Supervisor: Dr Devanshu Anand

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Jose Alberto Cruz Sanchez
Student ID:	X20191236
Programme:	MSc in AI **Year:** 2024
Module:	Practicum II
Lecturer:	Dr Devanshu Anand
Submission Due Date:	08/08/2025
Project Title:	Hybrid CNN-RNN Models and Explainability in early Lung Cancer Detection from Chest Radiographs: A Study on Enhancing Radiologist Confidence and Diagnostic Accuracy.
Word Count:	9916 **Page Count:** 24

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Jose Alberto Cruz Sanchez
Date:	8/8/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Forename Surname: Jose Alberto Cruz Sanchez
Student ID: x20191236

1 Project Environment Setup:

I did everything on my own laptop. It's running Windows 11, nothing fancy, just what I already had. I didn't use Colab or cloud services at all. To be honest, I prefer doing things locally. It's easier to track what's happening and I don't lose progress if something crashes or disconnects.

I used Anaconda to manage everything. Most of the time I was in the Anaconda Prompt, installing libraries and creating the environment. I created a new one from scratch because I didn't want to mess with anything else I had installed before. It took a while to get the right versions of stuff to play nice with each other. I had some weird errors with Tensor Flow at first, probably version conflicts, but I reinstalled things until it worked. Kind of trial and error.

My Python version was 3.10. I installed tensor flow, keras, opencv-python, pandas, numpy, matplotlib, seaborn, and also scikit-learn. Later I added lime, shap and something called tf-keras-vis to get Grad-CAM working. That one took a bit more time to set up. I had to follow a few tutorials to understand how to apply it to my model.

Most of the code I wrote in Jupyter Notebooks. I like how you can test stuff step by step. But sometimes I used VS Code when I needed to edit a bunch of things at once, especially model files or longer scripts. Honestly, switching between both worked well for me.

The dataset (ChestX-ray14) was stored on my hard drive. I had to split it into folders for training, validation and testing. I didn't automate that part, just used Python scripts and some manual checks to make sure things weren't duplicated across sets. I also resized the images and saved them locally so I wouldn't have to do that every time I ran the code. That helped a lot during training.

Nothing was super polished, but it got the job done. I had problems at times, like broken environments or things crashing, but that's part of the process. Once I had everything running, I didn't change too much.

2 Dataset Handling and Preprocessing:

For this project I used the ChestX-ray14 dataset. It's a pretty large set, thousands of chest X-ray images labeled with different lung conditions. Some files were clean, others not so much. A few images were blurry or oddly cropped, which kind of made things feel real, like what you'd get from a busy hospital system.

First thing I did was filter out images that didn't belong to adult patients. Then I checked for duplicates. I actually found more than I expected. I used a script to compare image names and patient IDs and cleaned those out. It took me a couple tries to get it right, the labels and image paths didn't always line up perfectly.

I resized all images to 224 by 224 pixels. That size matched the model I wanted to use later, so I stuck with it. I also ran histogram equalization using OpenCV, to improve contrast.

Visually, it made a difference. Some of the patterns that were hard to see looked clearer after that.
There was also the problem of class imbalance. Some conditions, like infiltration, appeared in a lot of images, while others, like Hernia, barely showed up. To help with that, I added data augmentation. I rotated, flipped and zoomed a few copies randomly. Nothing too aggressive, just enough to add some variation.
Since images could have more than one label, I converted all the conditions into binary vectors. Basically, a one-hot style label but allowing multiple 1s. That way, the model could learn to detect several things at once, like how a real X-ray might have more than one issue.
I split everything into three sets: training (70%), validation (15%), and test (15%). I made sure each patient was only in one of the sets, no mixing. I double-checked patient IDs to be safe. That took time, but it was worth it.
Once I finished, I had everything in local folders, resized images, organized sets and CSVs for the labels. Preprocessing was slow at first, but after it was set, it didn't need to be repeated.

3 Model Architecture and Training Configuration:

So for the model, I started with a convolutional base, EfficientNetB3 to be specific. I picked that one because it gave me good results in earlier experiments, and it's not too heavy compared to some other options. I didn't use the top classification layer that comes with it. Instead, I took the output and passed it into something else.
I decided to add a Bidirectional LSTM after the CNN part. I know that sounds a bit unusual, but I wanted to see if the model could learn something from the spatial patterns like a sequence, kind of like how one part of the lung might relate to another. It worked better than I expected, honestly. I connected that to a dense layer with 14 outputs, one for each condition in the dataset. Each output had a sigmoid activation since it was multi-label, meaning multiple diseases could appear in the same image.
I used binary cross entropy for the loss function. At first, I thought about using something more complex, but this one just worked. Also, I added class weights because the dataset was pretty imbalanced. Some conditions had thousands of examples, others barely a few hundred. Without the weights, the model just ignored the rare ones.
I trained the model using Adam optimizer. Learning rate was set to 1e-4, which I picked after testing a few options. If it was too high, the loss fluctuated like crazy. Too low, and it didn't really learn much. This one seemed stable.
Batch size was 32. I went higher once, to 64, but my GPU started acting up, memory issues, so I stayed with 32. I trained for up to 15 epochs, but I also used early stopping to avoid overfitting. In most runs, the training stopped before reaching 15. It usually plateaued around epoch 10–12.
For metrics, I tracked accuracy, AUC, precision, recall, and also specificity. Those were helpful, especially for the less frequent classes. I didn't focus too much on accuracy since it doesn't mean much in multi-label setups, but I included it anyway.
After training, I saved the model and ran predictions on the test set. The scores were okay. AUC was around 0.79, accuracy somewhere near 72%, and recall just above 67%. Not perfect, but for this kind of medical data, it was acceptable.

4 Explainability Integration:

After I got the model to train properly, I wanted to understand why it was making certain predictions, not just whether they were right or wrong. So I started looking into explainability

stuff. At first, I wasn't really sure where to begin. There are a bunch of tools out there and it's easy to get overwhelmed.
The one that worked best for me was Grad-CAM. It's a method that generates heat maps over the original image, showing which parts of the image the model was focusing on when it made a prediction. I used a library called tf-keras-vis for that. It wasn't super straightforward to set up. I had to change a few layers in the model and load the right weights. Took me a couple of tries, but in the end it gave pretty good results.
I tested Grad-CAM mostly on the test set. In some cases, the heat map clearly showed that the model was paying attention to the right area, like a mass in the upper left lung, or a thickened region near the pleura. But not always. Sometimes it highlighted parts of the image that didn't seem related at all. I guess that's just how deep learning works sometimes, it finds patterns we don't really see. Or maybe it's just guessing.
Besides Grad-CAM, I also tried LIME and SHAP. LIME was okay for local explanations, but it didn't give me much clarity with image data. SHAP was a bit more technical and hard to interpret visually. Grad-CAM was the most intuitive of the three, especially since I could overlay the heat map right on top of the X-ray.
I didn't go super deep into every explanation tool, but at least I got a sense of how the model was behaving. It helped me feel more confident that the predictions weren't completely random, at least in some cases, the model was actually "looking" at relevant areas.

5 Version Control and Reproducibility:

I didn't really use GitHub or anything online for this. I just worked from folders on my laptop. Most of the time I'd copy a folder before trying something new, like if I was going to change the model or mess with the dataset. That way if something broke, I could just go back to the old one.
I did have Git installed and I used it a bit, but just locally. I never pushed to GitHub. Wasn't sure if I should upload the dataset stuff and honestly, I didn't want to deal with it. So I just kept commits on my own machine. That helped me a couple of times when I forgot what I changed.
I saved the environment using the conda export thing. It creates a .yml file with all the packages and versions. That's useful if I ever need to run everything again. A few times I updated something and it broke stuff, so having the export was a lifesaver.
For the models, I saved the weights and also the training results. Not just the metrics, but also predictions, some of the heat maps and even some screenshots when I couldn't be bothered to automate it. Yeah, it's not the cleanest setup, but it worked for me.
There were moments where I forgot to save things, like intermediate models or some of the curves. I'd have to retrain or just skip it. But overall, I kept track of enough stuff that I could repeat most of the steps if I really had to.

6 Challenges and Lessons Learned:

There were honestly a lot of small issues along the way. Some I expected, others just came out of nowhere. At the start, I thought the hardest part would be building the model, but it turns out the setup and data cleaning took more time than I planned for.
One big problem was getting the libraries to work together. I lost count of how many times I had to uninstall and reinstall TensorFlow or some dependency because something didn't match. At one point I had an environment that just stopped working, and I still don't know why. I had to rebuild it from scratch. That was frustrating.

Also, working with medical data, it's not like working with clean datasets from Kaggle. A lot of the images had weird formats or were low quality and the labels weren't always easy to handle. The multi-label setup made it more complicated, especially when I was trying to evaluate the model. Precision and recall weren't giving me the full picture, so I had to learn more about things like specificity.
Another thing that caught me off guard was the class imbalance. Some diseases showed up a lot and others barely appeared. That really messed with the training at first. I tried oversampling, but it didn't help much. What worked better was adjusting the class weights and adding data augmentation, nothing fancy, just flipping and rotating images here and there.
Explainability was also new for me. Grad-CAM was helpful once I figured it out, but getting it to work took some digging. I had to read several blog posts and GitHub threads before I got it right. But once it worked, it was pretty cool to see where the model was looking. Some of the results made sense. Others… not so much.
Biggest lesson? Things break. And usually not where you expect. Also, save everything, checkpoints, scripts, outputs, even the stuff you think you won't need again. There were moments I had to go back and rerun something I hadn't saved, and it was annoying.
But overall, I learned a lot. Not just about deep learning, but also about patience, debugging, and how important it is to keep your work organized. If I had to do it all again, I'd probably plan better, but at least now I know what to expect.

7 Conclusions:

I am not going to lie, this project was kind of a rollercoaster.
Some days everything worked fine, and others… it felt like I was just fixing stuff nonstop. Errors, broken environments and weird bugs I couldn't explain. But yeah, I kept going.
One thing I'm honestly happy about is that I didn't use any fancy platforms. No cloud, no auto tools. Just me, my laptop and a bunch of trial and error. I learned a lot that way, sometimes the hard way. But it sticks more, I think, when you figure things out by yourself.
I had no idea explainability would take that much effort. Grad-CAM sounded simple until I actually had to implement it. Same with SHAP. But once I got some results, it felt worth it. Seeing what the model "looked at" was kind of cool, even if it didn't always make sense.
The main takeaway? Stuff breaks. Plans change. You go in thinking one thing and end up building something totally different. But I guess that's how learning works.
Would I do it all over again? Maybe not exactly the same way… but yeah, I'd do it again.
And weirdly enough.
I'm kind of proud of it.

References

[1] World Health Organization, Cancer Fact Sheet, 2024. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cancer

[2] American Cancer Society, Key Statistics for Lung Cancer, 2024. [Online]. Available: https://www.cancer.org/cancer/lung-cancer/about/key-statistics.html

[3] C. Li et al., "Advances in lung cancer screening and early detection." Cancer Biology & Medicine, vol. 19, no. 5, pp. 591–608, May 2022, doi: https://doi.org/10.20892/j.issn.2095-3941.2021.0690.

[4] J. Esteva, K. A. Robicquet, B. Ramsundar, et al., "A guide to deep learning in healthcare." Nat. Med., vol. 25, pp. 24–29, 2019. [Online]. Available: https://doi.org/10.1038/s41591-018-0316-z

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier." in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., 2016, pp. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[6] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction." Comput. Struct. Biotechnol. J., vol. 13, pp. 8–17, 2015. [Online]. Available: https://doi.org/10.1016/j.csbj.2014.11.005

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition" Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition

[8] G. Litjens, T. Kooi, B. E. Bejnordi, et al., "A survey on deep learning in medical image analysis." Med. Image Anal., vol. 42, pp. 60–88, Dec. 2017. [Online]. Available: https://doi.org/10.1016/j.media.2017.07.005

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks." in Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[10] W. Shen, M. Zhou, F. Yang, et al., "Multi-scale convolutional neural networks for lung nodule classification." in Inf. Process. Med. Imaging, vol. 9123, pp. 588–599, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-19992-4_46

[11] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification." in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2012, pp. 3642–3649. [Online]. Available: https://www.researchgate.net/publication/221663833_Multi-column_Deep_Neural_Networks_for_Image_Classification

[12] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks." in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), 2013, pp.

6645–6649. [Online]. Available: https://www.researchgate.net/publication/258818168_Speech_Recognition_with_Deep_Recurrent_Neural_Networks

[13] F. Chollet, Deep Learning with Python, 2nd ed., Manning Publications, 2021.

[14] Z. C. Lipton, “The mythos of model interpretability,” Queue, vol. 16, no. 3, pp. 31–57, 2018. [Online]. Available: https://doi.org/10.1145/3236386.3241340

[15] B. Goodman and S. Flaxman, “European Union regulations on algorithmic decision-making and a right to explanation.” AI Mag., vol. 38, no. 3, pp. 50–57, 2017. [Online]. Available: https://doi.org/10.1609/aimag.v38i3.2741

[16] R. R. Selvaraju, M. Cogswell, A. Das, et al., “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization.” in Proc. IEEE ICCV, 2017, pp. 618–626. [Online]. Available: https://ieeexplore.ieee.org/document/8237336

[17] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions.” in Adv. Neural Inf. Process. Syst., 2017, pp. 4765–4774. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[18] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv preprint arXiv:1409.1556, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[19] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases.” in Proc. IEEE CVPR, 2017, pp. 2097–2106. [Online]. Available: https://ieeexplore.ieee.org/document/8099852

[20] H. Rajpurkar, J. Irvin, K. Zhu, et al., “CheXNet: Radiologist-level pneumonia detection on chest X-rays with deep learning.” arXiv preprint arXiv:1711.05225, 2017. [Online]. Available: https://arxiv.org/abs/1711.05225

[21] Y. Zhang, J. Zhu, and B. Yao, “Weakly supervised medical diagnosis and localization from multiple resolutions.” arXiv preprint arXiv:1803.07703, 2020. [Online]. Available: https://arxiv.org/abs/1803.07703

[22] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[23] J. Hochreiter and J. Schmidhuber, “Long short-term memory.” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[24] M. Tjoa and C. Guan, “A survey on explainable artificial intelligence (XAI): Towards medical XAI.” IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 11, pp. 4793–4813, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9233366