


A grayscale photograph of the James Cook University Singapore entrance gate. The gate features a traditional Chinese architectural style with tiled roofs and white pillars. A sign on the gate reads "JAMES COOK UNIVERSITY SINGAPORE" and "www.jcu.edu.sg". In the background, a building with "JAMES COOK UNIVERSITY" is visible. A purple rectangular overlay is positioned in the foreground, containing the title text.

RESPIRATORY DISEASE DETECTION WITH CNN



Done By: Josiah Teh
Student ID: 14103845

INTRODUCTION

In 2019, the coronavirus disease (COVID-19) showed that detecting new respiratory diseases was not only expensive, but also takes considerable time and manpower to detect positive cases efficiently. During the early stages of the pandemic, the reverse transcription polymerase chain reaction test (RT-PCR) was the main testing kit used, being both expensive, while also requiring experienced medical personal to administer the testing. Due to these circumstances, governments were unable to effectively carry out testing, thus bringing about a global shutdown that lasted for nearly an entire year. Because of this, experts have been developing more easily accessible tools as an alternate diagnosis source, with one of the recent developments being x-ray imaging. Utilising artificial intelligence, specifically the field of deep learning, detection of undiscovered respiratory diseases with the usage of chest x-rays may be expedited while being able to maintain high accuracy. While deep learning has been successful in detecting positive COVID cases, I will be attempting to find out if convolutional neural networks are able to differentiate between COVID, viral pneumonia and normal cases.

LITERATURE REVIEW

Convolutional neural networks have been used in the medical field, being proven to be highly beneficial in detecting anomalies in five major organ systems: brain, pelvis, abdomen, musculoskeletal system, and chest/breast (Soffer et al., 2019). Benefits include expedited localisation of tumour locations (Khan et al., 2022), high accuracies and precisions without the need of image augmentation (Chowdhury et al., 2020), and increased opportunities for further development of state-of-the-art facilities.

In studies related to this research field, medical experts have a common consensus that compared to MRI or other techniques, CNN networks can produce results at a much faster and cheaper rate, while still maintaining a prediction rate of over 95% (Almezhghwi et al., 2021). While the models to vary to an extent, out of the 7 papers reviewed, classifiers from the VGG family, VGG16 and VGG19, had the highest accuracy rates, with the lowest being 94.1% (Naveen & Diwan, 2021) to a perfect accuracy of 100% (Khan et al., 2022).

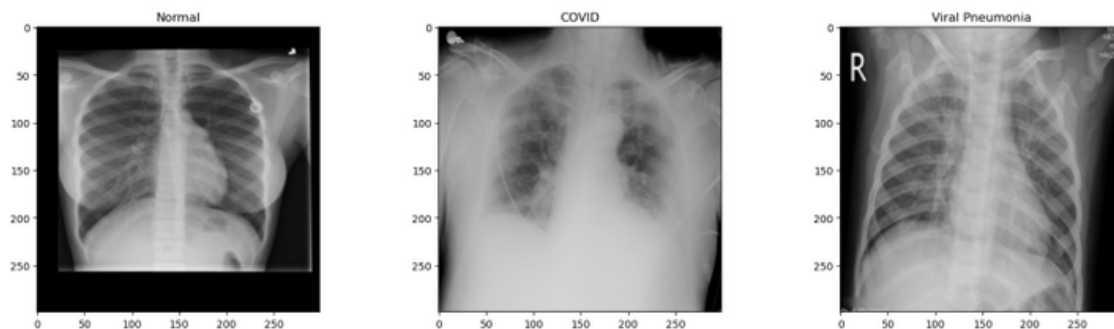
While these results seem promising, there are difficulties that greatly limit the development of deep-learning detection systems. One of the most significant difficulties is that, as with all deep learning models, a large number of labelled images must be collected by qualified personnel. This issue is widespread in the development of other automated models in the field of medicine. This issue can be summarised by saying that unlike more generalised image datasets, datasets consisting of quality x-ray images are sparse and small.

Another major drawback is that while these models have been tested greatly on datasets, none of the papers reviewed had their performance validated in an actual experiment. This means that while concepts such as zero-shot, one-shot or reinforced deep learning have had great success, their performance cannot be thoroughly endorsed unless tested against real clinical trials.

Applying concepts developed from the studies discussed prior, it can be said with high confidence that using techniques and CNN structures recommended will result in high precision metrics that can be applied to the benefit of both the community involved in research and development of this field, and for me to try and test my course knowledge to develop models that tackle real world issues.

For the capstone, a radiography database provided by Dr. Muhammad Chowdhury, Mr. Amith Khandakar and Mr. Tawsifur Rahman was selected to train our model. In this database, three classes, normal, COVID-19, and viral pneumonia we present, each having 10316, 3616 and 1315 respectively.

FIRST IMAGES FROM DATASET

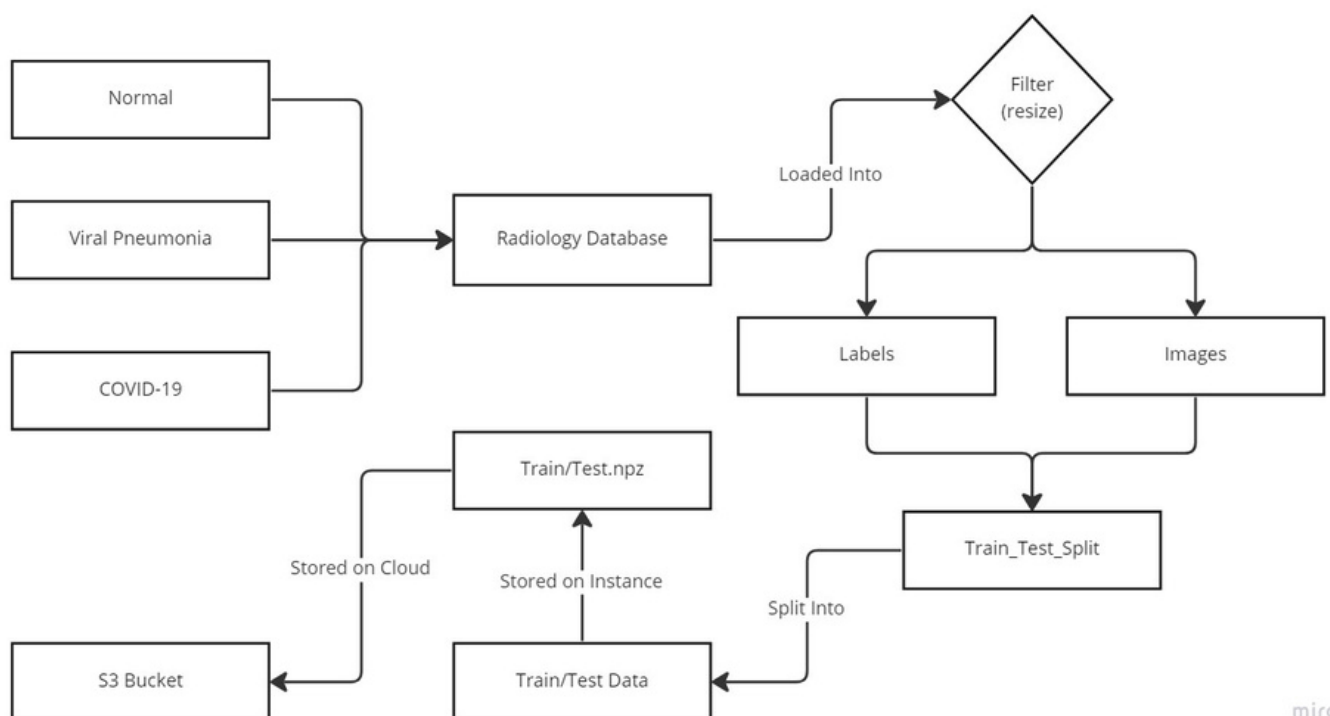


Graph of the first image of each of the three classes plotted with opencv

Each image had a resolution of 299 by 299, in the colour format red, green blue. VGG19 and VGG16 seemed to have the best results in terms of accuracy, I decided to resize each image to a resolution of 224 by 224. While normalisation was originally planned, I had to decide against it. This is because when applied in the pre-processing stage, the ram usage would exceed, even with a ml.m5.xlarge instance, and when executed via the python file when fitting the model, the kernel would simply just die. As each class consisted of at least 10% of the dataset, transformations such as random cropping or horizontal flipping were not applied.

After converting all classes to a numpy array format, the labels and images were stored on the memory into a y variable and x variable respectively. After this, the data was then split into a training and testing set, with a ratio of 80% to 20%. Storing them into a npz format, the two sets were then stored on the notebook instance, before being uploaded to the Amazon web services elastic storage system, otherwise known as the S3 Buckets.

DATA WRANGLING FLOWCHART



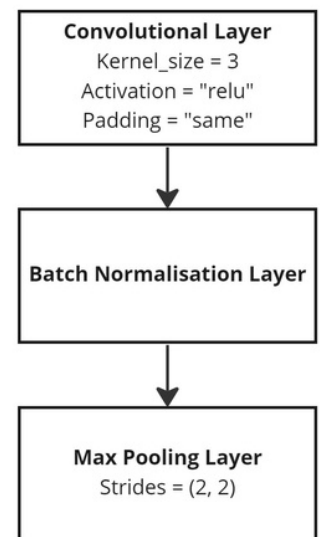
PROPOSED MODEL

04

For my proposed model, I heavily based my structure off the VGG19 model, as it was observed to maintain a high level of accuracy on non-augmented images among the papers reviewed. A total of 8 convolutional blocks were used for my model, each consisting of a convolutional layer, a batch normalisation layer and a max pooling layer. The values produced were then passed into a flattening layer, followed by 3 dense layers, the last one being a softmax activation to produce the output. As normalisation wasn't applied to our data, batch normalisation layers were used to compensate for this, while also speeding up the training process, while also allowing our model to improve its training process fast. Altogether, this structure produced 2,009,715 parameters, of which 2,007,555 were trainable.

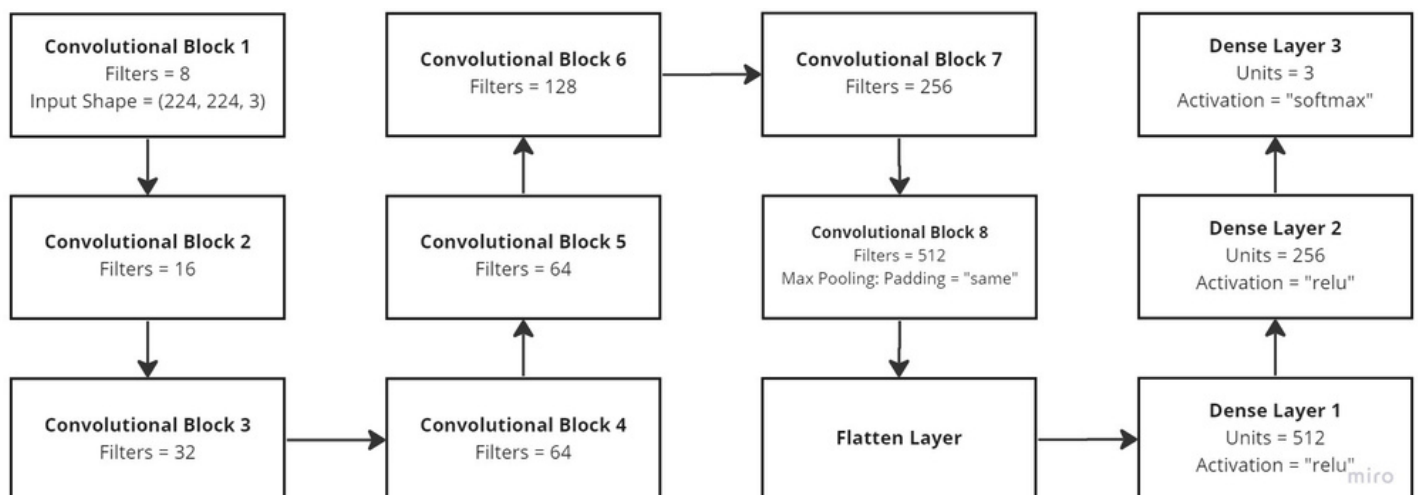
Regarding compilation, categorical cross entropy was chosen as our loss function due to our problem being a multi-class classification problem. Adaptive optimiser Adam was also chosen to be my optimiser, since it has been known to work well with complex problems involving large amounts of parameters. Another key reason for using the Adam optimiser is that it trained the model more efficiently, while requiring less memory commitment. This was one of the more important advantages, as kernel deaths and crashes due to over usage of ram issues were extremely problematic during my model development.

CONVOLUTIONAL BLOCK STRUCTURE



Standard structure of the eight convolutional blocks used in our proposed model

PROPOSED CNN STRUCTURE



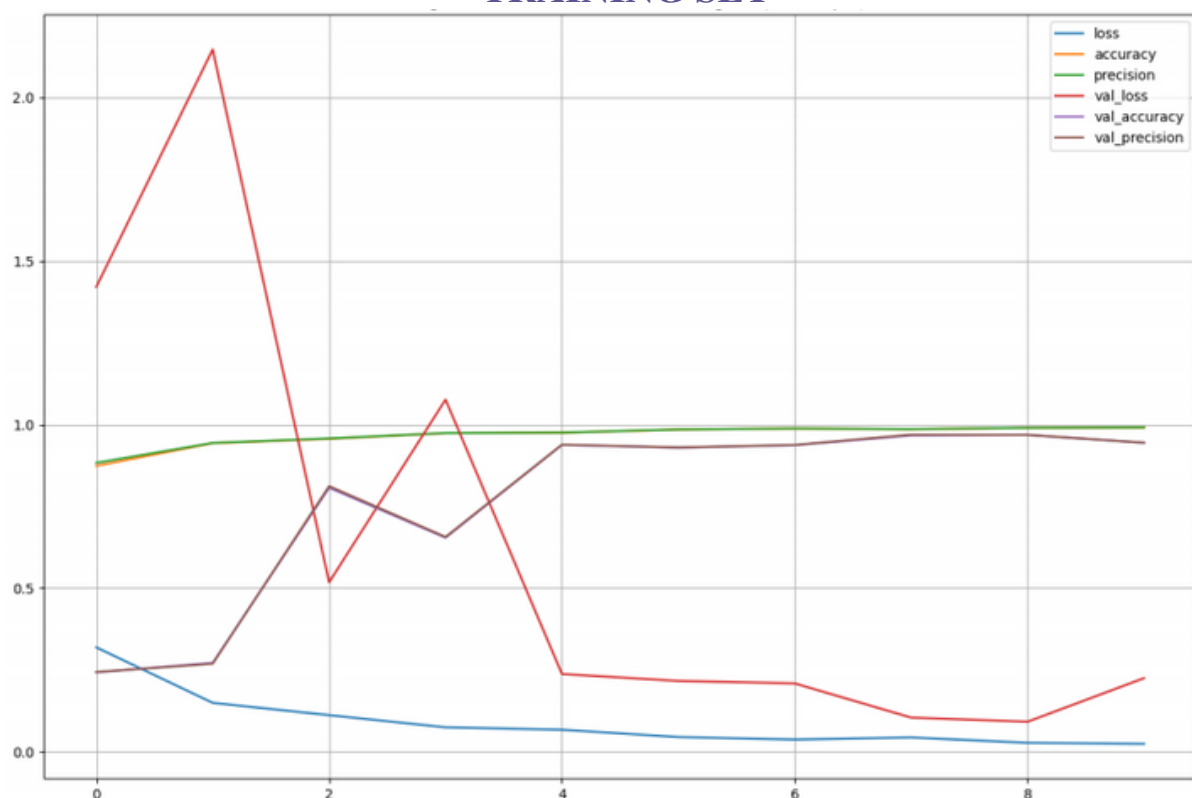
Complete diagram of proposed structure of CNN for chest x-ray classification for respiratory diseases

The model was trained in batches of 128, running for 10 epochs with a validation split of 90-10. Each iteration took an average of 115 seconds.

TRAINING SET

Storing the fitting process of the 10 epochs on a separate variable, I was able to visualise the data by converting the dictionary produced into a dataframe, plotting the graph with pyplot.

TRAINING & VALIDATION METRICS ON TRAINING SET



History of fitting produced on a 90-10 split from local python notebook

While loss for the training set during the fitting reduced at a constant rate, we can observe that the validation loss seems to spike at times. Visually, we can observe that validation loss seems to hit its minima at 0.09 at the ninth epoch, spiking up to 0.22 on the tenth. While training loss seems to still constantly decrease, we can observe that the difference between the ninth and tenth epoch is a 0.003 decrease. Because of this lack of significant decrease, I decided not increase the number of epochs for training my model.

Training Set Loss: 0.0248 Accuracy: 0.9913 Precision: 0.9915

Validation Set Loss: 0.2254 Accuracy: 0.9448 Precision: 0.9448

With our training accuracy and precision being less than 1% a perfect score, additional structural layers may be needed to further increase the effectiveness of the model on unseen data. Numerically, the validation set accuracy and precision lags behind that on the training set by around 5%. With the converging values, we can conclude that the data, applied with the proposed parameters make for a good fit.

MODEL EVALUATION

06

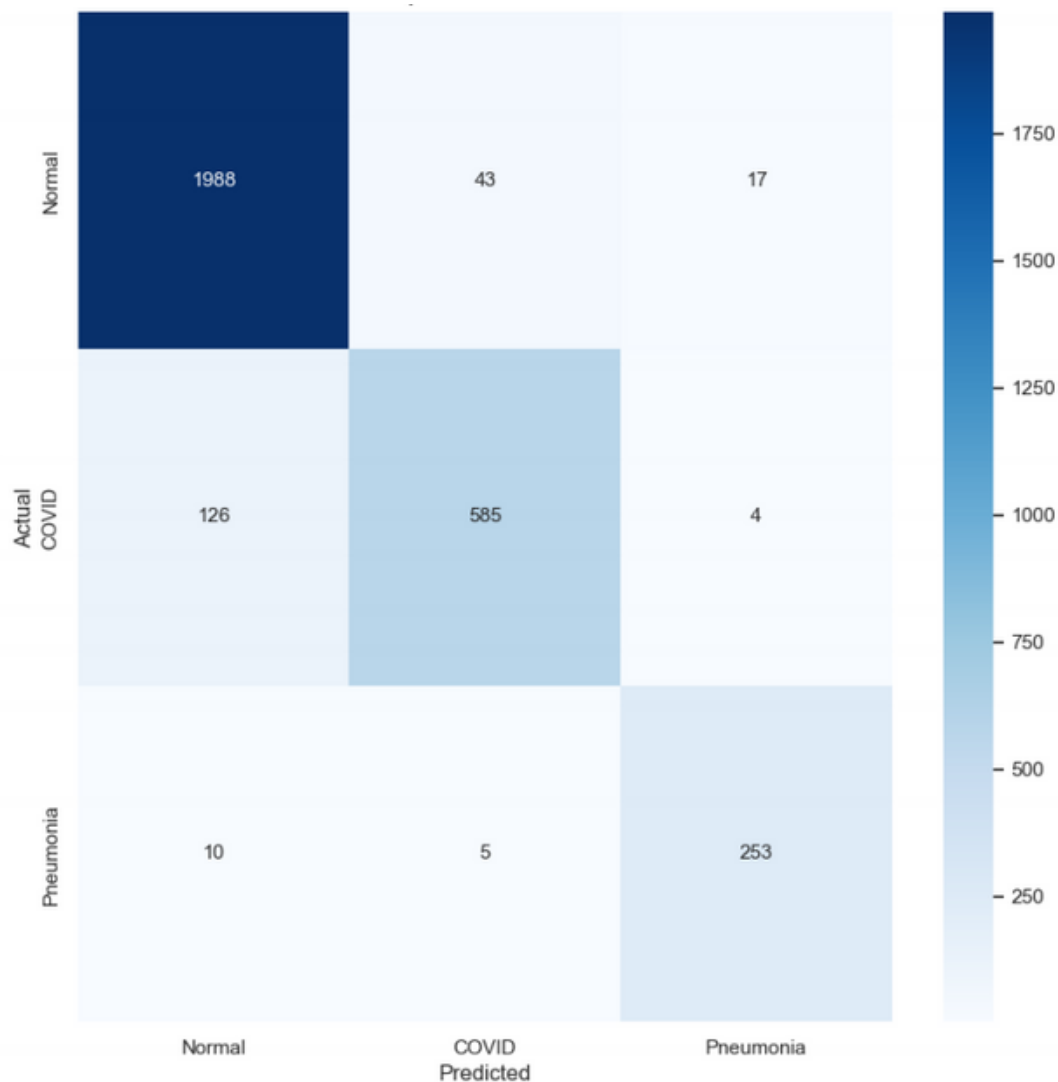
TESTING SET

Applying the test set to the evaluate function of the model, the loss, precision and accuracy were able to be obtained.

Test Set Loss: 0.2754 Accuracy: 0.9324 Precision: 0.9329

We can observe that performance on the test set is similar to that of the validation set, showing that the values produced during the fitting process were true, and that the model shows signs of a good fit.

PREDICTED VS ACTUAL CLASSES ON THE TEST SET



From the confusion matrix, we can see that the greatest number of misclassified cases are COVID-19 cases that are predicted to be normal. Fortunately, the model showed great precision, being able to distinguish between COVID-19 cases and pneumonia cases, classifying only 5 out of 585 and 4 out of 253 wrongly. For this case of detecting respiratory diseases, we can say that our model is much better at classifying the type of disease, rather than being able to detect them. With an accuracy of around 93% on the test set, I would say that the model performs at a satisfactory level, and may be useful as a point of reference when applied to real clinical uses.

ELASTIC CLOUD COMPUTING (EC2)

Elastic Cloud Compute Cloud, known more commonly as EC2, is a scalable cloud computing offered by Amazon Web Services. The EC2 allows the users to rent virtual servers, configure networks and security, and manage cloud storage. The main advantage of EC2 to other cloud computing services is that its scalable, allowing the user to adapt the load needed depending on how much traffic is needed.

For our project, two main EC2 services were used, a virtual computing environment, known as instances, for use in SageMaker, and the S3 buckets, which are persistent storage blocks. In the case for our virtual computing environment, 3 types of instances were used. the ml.m5.xlarge instance, ml.m4.xlarge.instance and ml.t3.medium instance.

ml.m5.xlarge VCPU: 4 Memory: 16gb \$ per Hr: 0.288

ml.m4.xlarge VCPU: 4 Memory: 16gb \$ per Hr: 0.244

ml.t3.medium VCPU: 2 Memory: 4gb \$ per Hr: 0.05

ENDPOINT DEPLOYMENT

Amazon SageMaker allows the user to deploy previously trained neural network models as endpoints, allowing the user or other parties to access them from another notebook instance. While this concept was known to me, and a endpoint was deployed on my end, I was unable to successfully access the endpoint to perform my predictions.

```
# Deploy model as an endpoint
proposed_predictor = cnn_model.deploy(initial_instance_count=1, # The initial number of instances to run in the Endpoint created from this
                                     instance_type='ml.t2.medium', # The EC2 instance type to deploy this Model to.
                                     endpoint_name='proposed-CNN-model') # The name of the endpoint to create
```

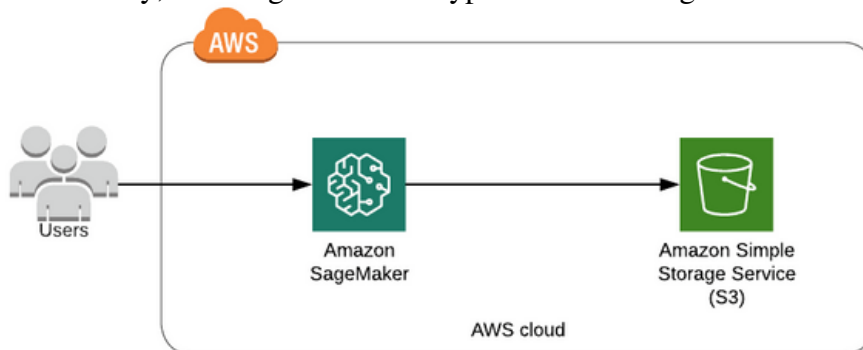
WARNING:sagemaker.deprecations:update_endpoint is a no-op in sagemaker>=2.
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> (<https://sagemaker.readthedocs.io/en/stable/v2.html>) for details.
INFO:sagemaker.tensorflow.model:image_uri is not presented, retrieving image_uri based on instance_type, framework etc.
INFO:sagemaker:Creating model with name: tensorflow-training-2023-01-31-16-13-05-881
INFO:sagemaker:Creating endpoint-config with name proposed-CNN-model
INFO:sagemaker:Creating endpoint with name proposed-CNN-model

Endpoint deployment of CNN model in MA3832_A4(Part1)_AWS

The main difference is that when the model is hosted locally, it can be used to predict simply by inputting the correct function, while the endpoint needs to be accessed via the boto3 library. Even though you are supposed to be able to just predict with the endpoint when initialised in the same notebook, I was unable too, and got met with a permission denied error. Another issue was that while I was able to set up my credentials via boto3, I couldn't predict, as the invoke_endpoint function requires the data to be in a file format. Unfortunately, I was unable to coerce my data into this format properly, so I had to perform the predictions on my local machine.

ELASTIC STORAGE (S3)

Another feature in AWS that was roughly touched on is its elastic storage service, also known as Amazon Simple Storage Service, or S3. This system allows users to store data within a cloud, and easily access it through other web services. In the case of SageMaker, data can be pre-processed and wrangled on a separate instance, uploaded to the S3 bucket, and then accessed directly via the Tensorflow entry point to train the model directly, allowing the user to bypass downloading the data onto their instance.



Architecture overview of dataflow from S3 to SageMaker from <https://awstip.com/deep-dive-to-train-deploy-and-evaluate-a-model-in-amazon-sagemaker-3c30514d9d17>

While the original plan was to use data pre-processed and stored in the S3 bucket to fine-tune my pre-trained model, I ran out of put, copy, post or list requests in my free tier. This, combined with my allocated quota reaching its limit, made me re-run the data pre-processing performed in the transfer learning model

USAGE

When I first started on the project, a `ml.t3.medium` instance was used to develop my models, but due to the low ram capacity, the instance crashed whenever I loaded the dataset onto the memory. Because of this, I split the notebook into two different instances, both using a `ml.m5.xlarge` instance. Compared to the `t3 medium` instance, which had only 4gb of ram, the `m5 xlarge` instance had 16gb of memory allocated, making it a much better choice than the previous instance. This instance was used for both data wrangling, before uploading the test and training set to the S3 bucket in the form of a npz file, and for structuring my own proposed model.

INSTANCE TRAINING ON SAGEMAKER

```
2023-01-31 16:11:48 Uploading - Uploading generated training model
2023-01-31 16:11:59 Completed - Training job completed
Training seconds: 1156
Billable seconds: 320
Managed Spot Training savings: 72.3%
```

Conclusion of training our proposed model using an entry point, deployed onto a `ml.m4.xlarge` instance

Using the SageMaker version of Tensorflow, an entry point to our designated python file was created, with the model creating, compiling and fitting being hosted on a separate `ml.m4.xlarge` instance. While the instance took a while to initialise, training on a separate instance was faster compared to when I trained on my local machine. Additionally, with how the python file receives the train and test data from the buckets, training could have been carried out on a separate instance to that of the data wrangling. Altogether, SageMaker took 1156 seconds to train the model, with only 320 of those seconds being billable.

VGG19

The VGG19 convolutional neural network is a variant of the Visual Geometry Group, with the base structure consisting of 16 convolutional layers, 3 fully connected layers, 5 max pooling layers and 1 softmax classification layer. Designed to compete in the ImageNet Large Scale Visual Recognition Challenge in 2014, it has been adapted in many types of object classification networks. The main advantage of using the VGG19 to other networks is that it uses a much smaller receptive window, using filters of 3 by 3, in strides of 1.

STANDARD STRUCTURE OF VGG19

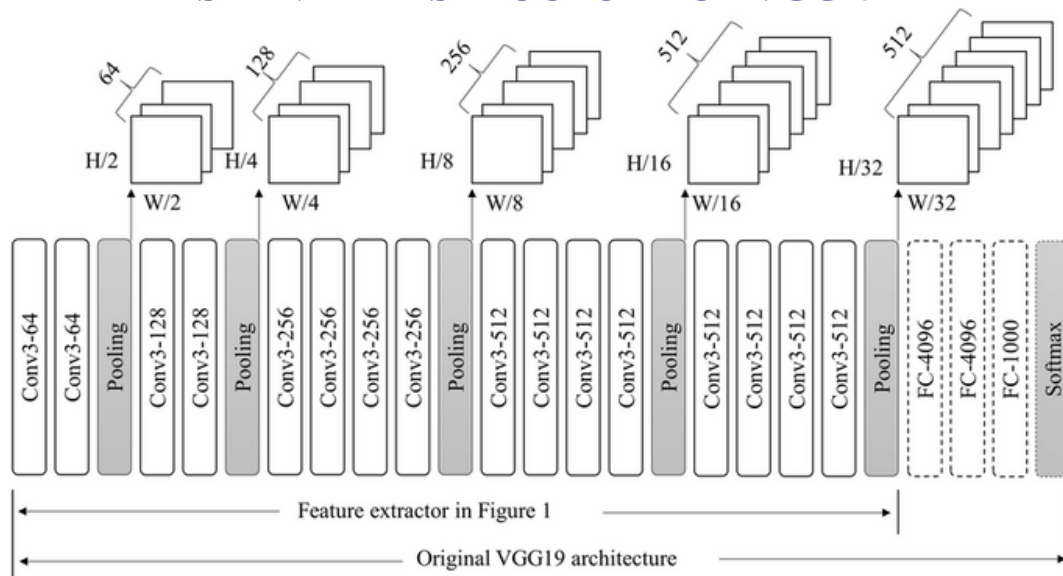
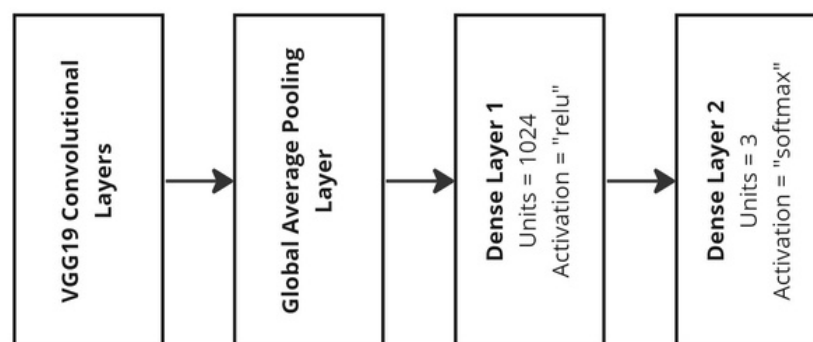


Image taken from: https://www.researchgate.net/figure/Feature-extractor-architecture-using-VGG19-network_fig1_344440237

Loading in the model, the 3 fully connected layers and softmax layers were removed, and were replaced by a global average pooling layer, a fully connected layer and my own softmax layer. The model was then compiled with similar parameters to my own model, with the optimiser being set to Adam, and loss function being categorical cross-entropy. Altogether, the model had 20,552,771 parameters, with 528,387 of them being trainable.

TRANSFER LEARNING MODEL STRUCTURE



After compiling, layers from the original VGG19 structure were frozen, and our two dense layers were then trained for 7 epochs. While training time per epoch averaged around 2 minutes for our own model, the transfer learning model took much longer, averaging around 30 minutes per epoch.

MODEL COMPARISON

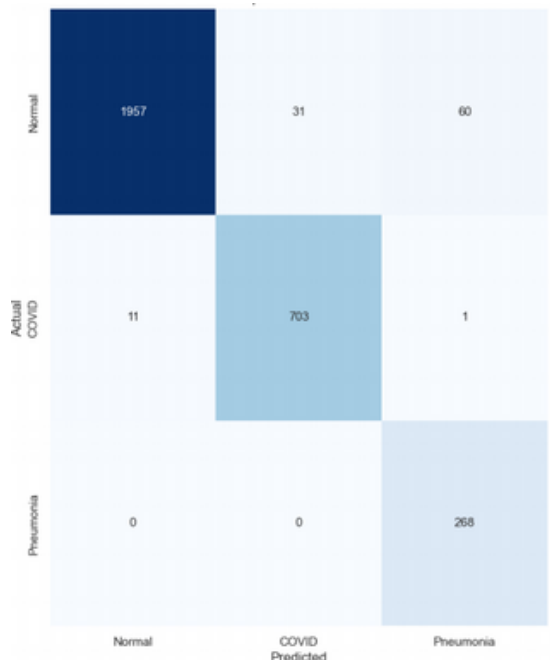
10

Proposed Model: Loss: 0.2754 Accuracy: 0.9324 Precision: 0.9329

VGG19 Model: Loss: 0.1030 Accuracy: 0.9693 Precision: 0.9693

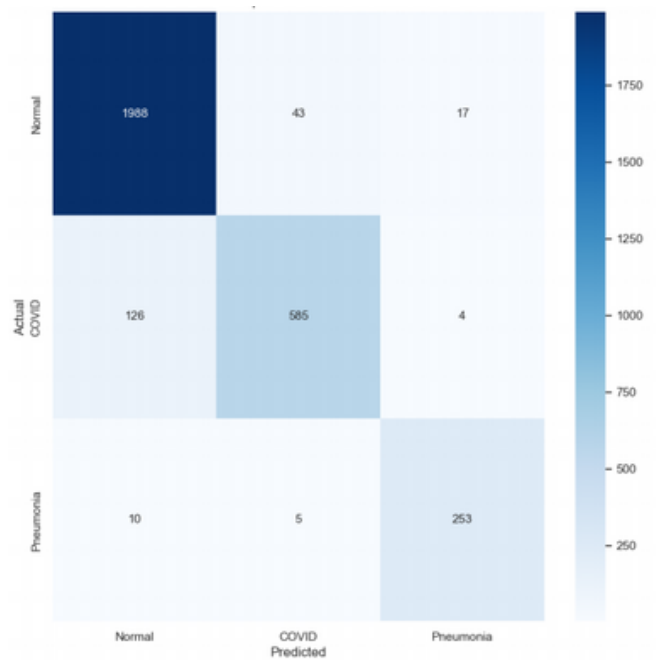
Evaluating on the test set, the transfer learning model had a much lower loss value, while also being superior in both prediction and accuracy, performing nearly 4% better than our own model. Similar to the training time, the VGG19 model took much longer to predict on the test set, compared to the proposed model.

VGG19



	precision	recall	f1-score	support
Normal	0.99	0.96	0.97	2048
COVID	0.96	0.98	0.97	715
Pneumonia	0.81	1.00	0.90	268
accuracy			0.97	3031
macro avg	0.92	0.98	0.95	3031
weighted avg	0.97	0.97	0.97	3031

PROPOSED MODEL



	precision	recall	f1-score	support
Normal	0.94	0.97	0.95	2048
COVID	0.92	0.82	0.87	715
Pneumonia	0.92	0.94	0.93	268
accuracy			0.93	3031
macro avg	0.93	0.91	0.92	3031
weighted avg	0.93	0.93	0.93	3031

Consulting the confusion matrices, we can see that the VGG19 model was able to predict both normal and COVID-19 cases with a much greater accuracy than my own. However, it can be noted that it misclassified a greater majority of viral pneumonia cases, with 60 being classified as normal, compared to the 17 in our own model.

Looking at the respective classification reports, it can be seen that the VGG19 model classifies normal and COVID-19 cases 5% and 4% better respectively, but accuracy in pneumonia cases is much worse, being 11% worse than our own model.

While our model was not as accurate as the VGG19, it was far more efficient at fitting and predicting, being able to train 15 times faster than the pre-trained model. As the validation accuracy on the test set is at 99%, it can be inferred that further improvements to our model have to be made by tweaking certain parameters. Even so, I am pretty satisfied with the performance with my model, and can say that it is able effectively classify x-ray images at a moderately efficient rate.

- Almezhghwi, K., Serte, S., & Al-Turjman, F. (2021). Convolutional neural networks for the classification of chest X-rays in the IoT era. *Multimedia Tools and Applications*, 80(19), 29051–29065. <https://doi.org/10.1007/s11042-021-10907-y>
- Chowdhury, M. E. H., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M. A., Mahbub, Z. B., Islam, K. R., Khan, M. S., Iqbal, A., Emadi, N. A., Reaz, M. B. I., & Islam, M. T. (2020). Can AI Help in Screening Viral and COVID-19 Pneumonia? *IEEE Access*, 8, 132665–132676. <https://doi.org/10.1109/access.2020.3010287>
- Khan, Md. S. I., Rahman, A., Debnath, T., Karim, Md. R., Nasir, M. K., Band, S. S., Mosavi, A., & Dehzangi, I. (2022). Accurate brain tumor detection using deep convolutional neural network. *Computational and Structural Biotechnology Journal*, 20, 4733–4745. <https://doi.org/10.1016/j.csbj.2022.08.039>
- Naveen, P., & Diwan, B. (2021, March 1). Pre-trained VGG-16 with CNN Architecture to classify X-Rays images into Normal or Pneumonia. *IEEE Xplore*. <https://doi.org/10.1109/ESCI50559.2021.9396997>
- Rahman, T., Khandakar, A., Qiblawey, Y., Tahir, A., Kiranyaz, S., Abul Kashem, S. B., Islam, M. T., Al Maadeed, S., Zughaier, S. M., Khan, M. S., & Chowdhury, M. E. H. (2021). Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *Computers in Biology and Medicine*, 132, 104319. <https://doi.org/10.1016/j.combiomed.2021.104319>
- Setiawan, W., & Damayanti, F. (2020). Layers Modification of Convolutional Neural Network for Pneumonia Detection. *Journal of Physics: Conference Series*, 1477, 052055. <https://doi.org/10.1088/1742-6596/1477/5/052055>
- Soffer, S., Ben-Cohen, A., Shimon, O., Amitai, M. M., Greenspan, H., & Klang, E. (2019). Convolutional Neural Networks for Radiologic Images: A Radiologist's Guide. *Radiology*, 290(3), 590–606. <https://doi.org/10.1148/radiol.2018180547>