Machine Learning Model to Evaluate Endotracheal Tube Placement

Final Report

Tina Gholami
Department of Electrical and Computer Engineering Western University London, Canada

Kyle Lacroix
Department of Electrical and Computer Engineering Western University London, Canada

Spencer Vecile
Department of Electrical and Computer Engineering Western University London, Canada

***Abstract*—More people than ever before are requiring ventilation due to COVID-19. Since proper tube insertion is essential for an optimal outcome, tube placement is verified using x-rays. A radiologist must read the x-ray to determine if the tube was placed properly. Due to mounting caseloads caused by COVID-19, automation of the x-ray reading process would result in higher turnaround times and improved patient care. The National Institutes of Health Clinical Center provided the chest x-ray data of patients showing the placement of the tubes. The x-rays of the chest images were first prepared through preprocessing. The images were reduced in size since the larger images required increased processing time but did not offer any additional information. Two machine algorithms, CNN and Google's EfficientNetB0, were used to detect misplaced tubes. Both models successfully determined if the tube was properly positioned with an accuracy between 70-97%. These models show promise to replace radiologists with machine learning algorithms, which will reduce caseloads and improve patient care especially during these trying times of dealing with COVID-19. Further tuning of the model would likely improve the accuracy of the models.**

Keywords—machine learning, illness

I. Introduction

As COVID-19 continues to spread around the world, hospitals are becoming overwhelmed. Doctors and nurses cannot spend as much time with their patients as they used to which could lead to an increase in human errors. When COVID-19 enters the body, it can slowly lower the oxygen saturation levels. If the blood oxygen level is too low, the blood cannot provide organs with enough oxygen needed for life. To deal with this, hospitals put patients on forced ventilation to provide sufficient oxygen. The patient must be intubated, a procedure in which a doctor or nurse inserts an endotracheal tube into the throat. To ensure the tubes are placed correctly, a physician or radiologist must manually look at an x-ray to verify that they are placed in the optimal position. Not only does this leave room for human error, but delays are also common as radiologists can be busy reading other scans. In fact, it is estimated that as many as one in four patients outside of the emergency room have misplaced endotracheal tubes [1]. Up to 3% of all NG tubes can be malpositioned, with upwards of 40% showing complications [1]. Implementing an automated process that can provide instant feedback on tube placement could help patient care and reduce hospitals' workload. The hospitals would considerably benefit from this process even when the COVID-19 crisis is over. This project will examine the possibility of using machine learning to read x-rays to determine if endotracheal tubes are correctly inserted.

In this paper, we will be discussing the use of two different machine learning algorithms: CNN and EfficientNetB0, to detect malpositioned tubes. Related work that is relevant to this project is presented. The dataset used and the methodologies employed, including preprocessing and implementation of the two models, are explained in detail. The results are analyzed with suggestions for further study.

II. Related work

A recent study by Borkowski *et al*. [2], investigated using AI to diagnose COVID-19 reliably. These researchers used publicly available CXR images for patients with COVID-19 pneumonia, pneumonia from other etiologies, and normal CXRs as a dataset to train Microsoft CustomVision. Pneumonia is an infection that inflames your lungs' air sacs. The researchers created a trained model with 92.9% recall and precision. While their metrics were not as high as some ML projects, it does show promise. Utilizing more data that captures more information might yield an even better result. Another study by Zebin *et al*. [3], examined using deep learning for detection of COVID-19. The researchers used pre-trained convolutional, EfficientNetB0, as a backbone for feature extraction and achieved an overall detection accuracy of 96.8%. The classifier effectively distinguishes inflammation in lungs due to COVID-19 and pneumonia from the ones with no infection (normal). Another study by Jain *et al*. [4] took a closer look at pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning. The researchers created six models. The first two models had two and three convolutional layers respectively. The remaining four models were pre-trained models. By the end of the study, they had six models that had decent validation accuracy ranging from 70.99-92.31%. This study furthers the proof of concept of

using machine learning to read x-rays to determine whether the tubes are properly placed.

III. Dataset

The dataset used for this project consists of x-ray images of catheter lines inside the patient's lungs [5]. This dataset was provided by the National Institutes of Health Clinical Center and was labelled by a team of over 30 doctors. This dataset is broken down into a set containing 3582 test images and 30083 training images; unfortunately, the test images are not labelled as they were for the competition so they will be unusable. The images are not uniform in size and range from 2000-3000 pixels in width and 2000-3000 pixels in height. Also included in the dataset is a train.csv file which has 13 columns including StudyInstanceUID (unique ID for each image), ETT - Abnormal (endotracheal tube placement abnormal), ETT - Borderline (endotracheal tube placement borderline abnormal), ETT - Normal (endotracheal tube placement normal), NGT - Abnormal (nasogastric tube placement abnormal), NGT - Borderline (nasogastric tube placement borderline abnormal), NGT - Incompletely Imaged (nasogastric tube placement inconclusive due to imaging), NGT - Normal (nasogastric tube placement borderline normal), CVC - Abnormal (central venous catheter placement abnormal), CVC - Borderline (central venous catheter placement borderline abnormal), CVC - Normal (central venous catheter placement normal), Swan Ganz Catheter Present and PatientID (unique ID for each patient in the dataset). This CSV file has columns 2-11 in a binary encoding format which is the true labels.

IV. Methodology

A. Exploratory Analysis

The first thing revealed during the dataset exploration was that although there are 30083 images, there are only 3255 unique patients so some patients have multiple x-rays in the dataset. At maximum, a patient has 172 x-rays and at minimum, a patient has one x-ray. Next was the number of observations for each label which is summarised in the Fig. 1 & 2:

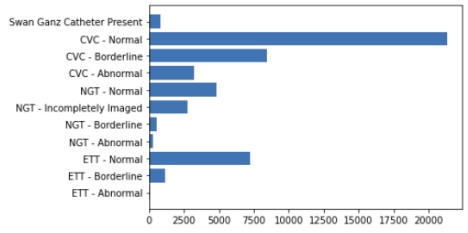

Fig. 1 Bar chart of the data

Malpositions	Number of Observations
ETT - Abnormal	79
ETT - Borderline	1138
ETT - Normal	7240
NGT - Abnormal	279
NGT - Borderline	529
NGT - Incompletely Imaged	2748
NGT - Normal	4797
CVC - Abnormal	3195
CVC - Borderline	8460
CVC - Normal	21324
Swan Ganz Catheter Present	830

Fig. 2. The number of each label in the data

From the chart and table, it can be seen that the classes are not balanced with the majority of them being CVC-normal. It can also be seen from the label totals that each image can have multiple labels which makes this a multi-label classification problem rather than a multi-class classification problem. Further to that point when the csv file containing the labels was explored, there were also many instances where an x-ray was labeled, for example, CVC-Normal and CVC-Borderline. This also happened albeit less frequently within the NGT and ETT categories leading to the conclusion that the CVC tube is difficult to classify even for the doctors that labeled the dataset.

B. Preprocessing

The data leakage issue mentioned in the exploratory analysis was addressed by using a stratified k-fold splitting algorithm that was found online and modified to fit the needs of the problem. The algorithm split the dataset into 5 folds, each of which had similar distributions of each class label while also ensuring that all of a patient's records were only contained within one fold. Each record was assigned a fold number ranging from 0-4 and this information was saved alongside the label information in a csv file called "train_folds.csv". These folds were then used before training to split the data as follows: 3 folds for the training set (folds 0-2), one fold for the validation set (fold 3), and one fold for testing set (fold 4).

Once this was complete, the images were loaded in grayscale format reducing the number of channels from 3 to one. This caused a significant reduction in size which made it much easier to load the dataset and did not cause a significant reduction in information since x-rays are already black and white images. The images were then resized to 224x224 and paired with their corresponding labels and fold number. The step where the labels are added differs slightly depending on which model is being used.

1. Combined Label Model

For the combined label model, first, the training labels are loaded and the columns StudyInstanceUID and PatientID are dropped. A new column is created called combined_label. In this column, first, the binary encodings are concatenated into a single binary label and then encoded using Sklearn's LabelEncoder class. This results in 211 unique labels in the dataset.

2. Multi-Label Distributed Model

The preprocessing for this model differed by the fact that there were no combined labels. Instead, the labels were attached in separated binary format e.g. [1,0,0,1,0,1,0,1,1,0,1].

3. Another alternative approach

Another alternative approach was also taken to deal with the data leakage issue. Since there were multiple samples recorded for some of the patients in the dataset, measures had to be taken to avoid such instances being present in both the train set and test set while they are being split. To address this issue, one approach is to employ cross-validation, which was mentioned previously. The other approach is to separate patients with multiple samples from the dataset, split the dataset into train

and test sets, and then add those separated samples only to the train set.

C. Model

The first model tried was a simple architecture that had very poor results.

1. Combined Label Model

For this model, Google's EfficientNetB0 [6] was chosen for its exceptional performance with image classification tasks. It used randomized initial weights, an input size of 224x224x1, sigmoid activation function on the output layer, and 211 output classes. The model uses sparse categorical cross-entropy loss since this is a multi-class classification problem due to the combined label. It uses Adam's optimizer with a learning rate of 0.01 and sparse categorical accuracy as the metric. Finally, it was trained for 30 epochs with a batch size of 128.

2. Multi-label Distributed Model

This model was an improvement on the combined label model. Instead of using the combined labels in a multi-class classification problem, the binary labels were considered individually in a multi-label classification format. There were also 6 EfficientNetB0 models used instead of one. One would classify EET placement into normal, abnormal, and borderline. One would classify NGT placement into normal, abnormal, incompletely imaged, and borderline. Three models would classify CVC into normal, abnormal, and borderline respectively and finally one was used as a binary classifier to detect if the Swan Ganz catheter was present or not. The idea was to allow the EET, NGT and Swans Ganz models to specialize in one type of tube to improve classification accuracy. Then, three models would be used to detect CVC abnormal, borderline, and normal since CVC was the hardest to detect and was not working well combined into one neural net. Now, since this is a multi-label classification problem, sigmoid was used as the activation at the output layer with binary cross entropy as the loss function. What this does is if for example, the output layer has three output labels, it treats each of them as their own binary classification problem so their probabilities do not sum to one and now abnormal and normal can be predicted simultaneously. Each of the models used an input size of 224x224, Adam's optimizer with a learning rate of 0.01, and binary accuracy as the performance metric. Finally, they were each trained for 10 epochs with a batch size of 128.

3. CNN the First Format, Using Multi-Class Classification

The second model is the Convolutional Neural Network (CNN), which itself is implemented within two different formats. In the first format, the problem was considered to include four separate similar models for each of the ETT, NGT, CVC, and SG labels. Each involved three convolutional layers containing 32, 64, and 64 neurons with kernels of size (3, 3) followed by MaxPooling layers of pool sizes (2, 2). The flattened output was then fed into a feedforward neural network involving two hidden layers of 128 neurons with Relu as the activation function and an output layer of three neurons with softmax as the activation function. Moreover, as a means of using the regularization method in our model, BatchNormalization and Dropout layers were included too. Early stopping was employed to combat overfitting. The model was trained under 1000 epochs and batches of size 32. The results for each of the models are provided in the Result section. As shown in Fig. 3 & 4 and Table 1 for the ETT model, the training loss increases as the training accuracy stays around 28%. Similarly, the validation loss increases with a smaller slope, as its accuracy stands around 10%. Since early stopping is employed, the model stops training at the first few epochs to stop overfitting.

Fig. 3. The Training (blue) and Validation (orange) Loss, The First Format of the CNN Model for the ETT Label

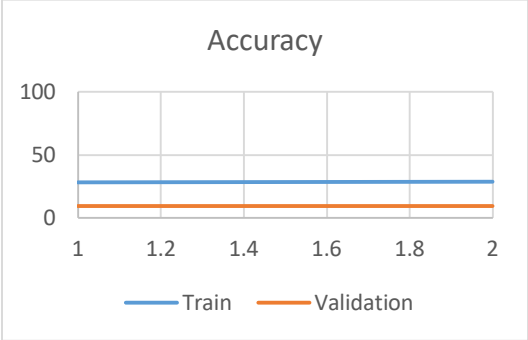

Fig. 4. The Training (blue) and Validation (orange) Loss, The First Format of the CNN Model for the ETT Label

	loss	accuracy	val_loss	val_accuracy
0	3970.299396	0.283845	581.543796	0.094891
1	87025.352235	0.287258	4616.204380	0.094891

Table 1. The model's metrics, The first format of the CNN model for the ETT label

Next, a similar model was trained on the NGT label. The results shown in Fig. 5 & 6 and Table 2 indicate that the training loss increases as the training accuracy stays around 22%. Similarly, the validation loss increases with a smaller slope, as its accuracy stands around 8%.

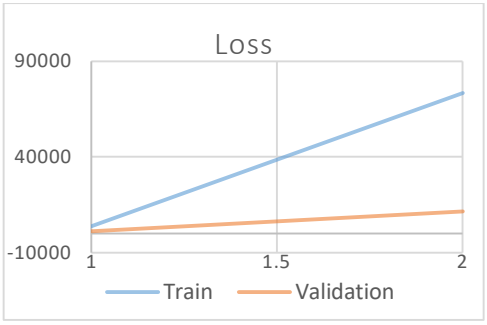

Fig. 5. The training (blue) and validation (orange) loss, The first format of the CNN model for the NGT label

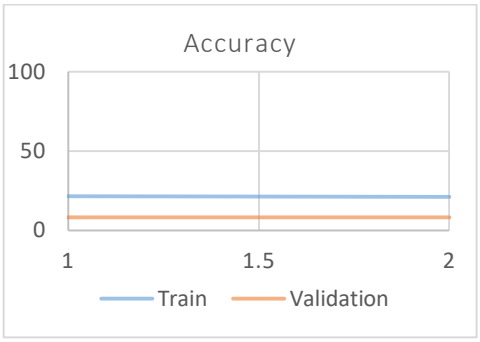

Fig. 6. The training (blue) and validation (orange) accuracy, The first format of the CNN model for the NGT label

	loss	accuracy	val_loss	val_accuracy
0	3903.985608	0.214638	1070.190237	0.080292
1	73463.149249	0.211414	11488.284686	0.080292

Table 2 The model's metrics, The first format of the CNN model for the ETT label

Similarly, for the CVC label, the results are shown through Fig. 7 & 8 and Table 3 indicate that the training loss increases as the training accuracy drops from 52% to 48%. Similarly, the validation loss increases with a larger slope, as its accuracy stands around 57% which does not make sense, as the test accuracy always has to be less than the training accuracy.

Fig. 7. The training (blue) and validation (orange) loss, The first format of the CNN model for the CVC label

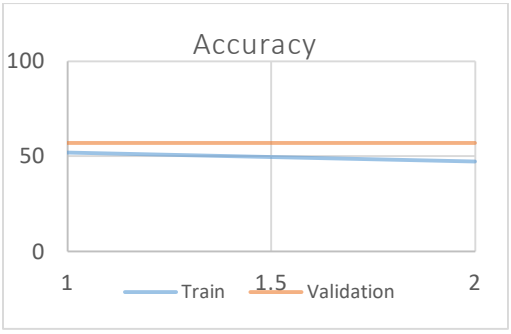

Fig. 8. The training (blue) and validation (orange) accuracy, The first format of the CNN model for the CVC label

	loss	accuracy	val_loss	val_accuracy
0	20.491757	0.518392	117.028420	0.569343
1	1206.700313	0.472886	1745.289688	0.569343

Table 3. The model's metrics, The first format of the CNN model for the CVC label

Finally, the SG model results are shown in Fig. 9 & 10 and Table 4. The results show that the training loss increases as the training accuracy is around 97%. Similarly, the validation loss increases with a larger slope, as its accuracy stands around 98%.

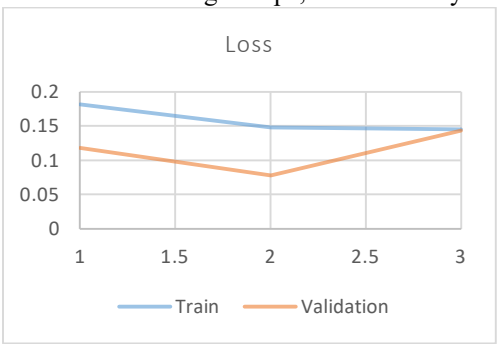

Fig. 9. The training (blue) and validation (orange) loss, The first format of the CNN model for the SG label

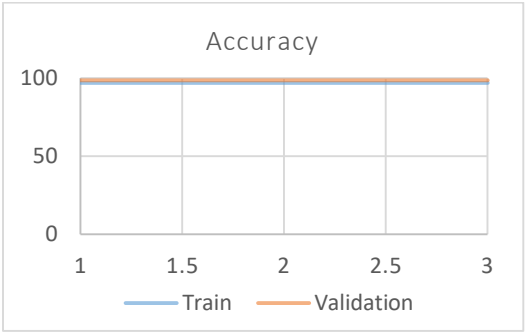

Fig. 10. The training (blue) and validation (orange) accuracy, The first format of the CNN model for the SG label

	loss	accuracy	val_loss	val_accuracy
0	0.181864	0.967956	0.117947	0.985401
1	0.148303	0.969662	0.077859	0.985401
2	0.145081	0.969662	0.143082	0.985401

Table 4. The model's metrics, The first format of the CNN model for the SG label

All of the above implies that such a format of modeling the data is wrong and that we have to consider a second format of the CNN model. The first format tried to do multi-class classification, which was found to be incorrect through the results. Also, when the binary classification was used for the SG target (since it only contained a binary class), the results showed a significant improvement. Furthermore, we found instances in the data that showed for a CVC target, a patient with both CVC-Borderline and CVC-Abnormal at the same time. Thus, multi-class classification does not make sense here. Instead, in the second format of the CNN model, we tried to incorporate a multi-label classification perspective. In this second format, a CVC model is broken down into three separate similar models as CVC-Abnormal, CVC-Borderline, and CVC-Normal, each using binary classification.

4. *CNN the Second Format, Using Multi-Label Classification*

In the second format, multi-label classification is considered. Therefore, the four main labels are broken down into several models based on the number of its categories. For instance, the CVC model has three different categories: abnormal, borderline, and normal. That is why three separate similar models are built for the CVC label, each implementing binary classification. In this paper, only the results of the CVC label under this method are studied. Similarly, the models for the other labels can be built the same way. It is important to mention that each of the label categories is still highly imbalanced. That is why the training overfits rather quickly. To address this issue, transformations should be applied on the data, such as Cox-Box transformation and Log transformation. These transformations help reduce the skewness in the data distribution, and thus, we will end up with a distribution closer to that of a Normal Gaussian. But solving this issue is another literature that is not profoundly studied in this work.

a. CVC-Abnormal Model

As it is shown in Fig. 11, the CVC-Abnormal category is imbalanced. The model for this target label is the same as mentioned in part C.2 for the first format of the CNN, except that there is only one neuron with a sigmoid activation function in the output layer. Accordingly, binary cross-entropy is selected as the loss function. The other hyperparameters are similar to part C.2. The results of the model's performance are shown in Fig. 12 & 13 and Table 5. Both training and validation loss decrease before the model starts to overfit. The accuracies show a significant improvement as opposed to the results of part C.2. The training accuracy is around 90%, with the validation accuracy being around 84%.

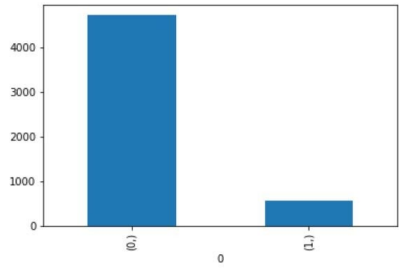

Fig. 11. The class imbalance, The second format of the CNN model for the CVC-Abnormal

Fig. 12. The training (blue) and validation (orange) loss, The second format of the CNN model for the CVC-Abnormal

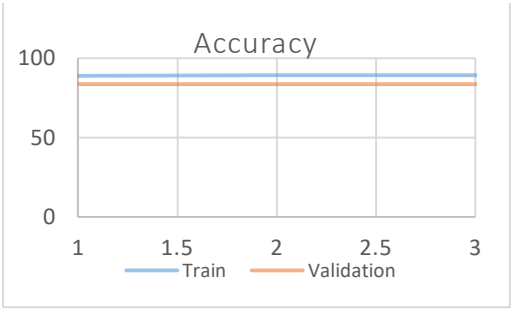

Fig. 13. The training (blue) and validation (orange) accuracy, The second format of the CNN model for the CVC-Abnormal

	loss	accuracy	val_loss	val_accuracy
0	0.377444	0.888699	0.474184	0.839416
1	0.350020	0.894008	0.455701	0.839416
2	0.353163	0.894008	0.442922	0.839416
3	0.347672	0.894198	0.495567	0.839416

Table 5. The model's metrics, The second format of the CNN model for the CVC-Abnormal

b. CVC-Borderline Model

Fig. 14 indicates that this label category is imbalanced as well. A similar model to (a) is used here. Both training and validation loss decrease over epochs. Both accuracies are higher than that of part C.2. The training accuracy increases to 71%, while the validation accuracy is about 75%.

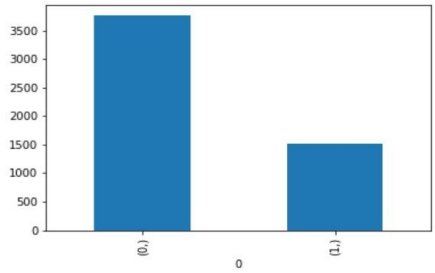

Fig. 14. The class imbalance, The second format of the CNN model for the CVC-Borderline

c. CVC-Normal model

Like the other two, CVC-Normal is also imbalanced, and the model used is the same as well. As the loss decreases, the training accuracy goes to about 71%, and the validation accuracy is about 66%.

In general, the more imbalanced a category is, the less accuracy it yields. As compared to the results of part C.3, the model performance in C.4 has drastically increased.

D. Hyperparameter Tuning

To start, the images were shrunk to 650x650 but this created problems with the dataset being too large. It was then scaled down to 450, 350 and finally 224. There was no difference between classification accuracy that could be seen with the different image sizes; however, with a smaller sized image, training was much quicker and the whole dataset could fit in memory. Batch size was also tuned. First, 64 was used but training was very slow so it was lowered to 32 which cut training time in half while maintaining epochs constant.

Several tunings were applied to the CNN model. First, three optimizers: Adam, Stochastic gradient descent, and Root mean square propagation, as well as three different learning rates: 0.1, 0.001, and 0.0001 were checked. The best results (training accuracy 70.65% and validation accuracy 66.42% on CVC-Normal) were achieved via Adam as the optimizer with a learning rate of 0.1. This result was then further analyzed using K-fold cross-validation, which yielded an average accuracy of 70.65% with a standard deviation of 2.06%.

Next, Grid Search was applied on three different batch sizes: 10, 32, and 100. And since early stopping was employed, three different epochs: 2, 5, 10 were used as well. However, the same results as before were achieved. Hence, neither the number of epochs nor batches make a stark difference in the results. Moreover, a smaller dataset of only 500 images was also fed to the model, and according to the grid search results, the batch sizes and the epochs influenced the model's performance. Therefore, as the dataset gets larger and more diverse, the model becomes robust to the number of epochs and batches.

V. Results for EfficientNetB0

The results for both models C.1 and C.2 will be shown and discussed.

A. C.1

The first model employed is C.1. The results of the model are shown in Fig. 15. Since the training accuracy was quite high and the validation accuracy was very low, the model did not generalize well. The best test accuracy achieved was 35%.

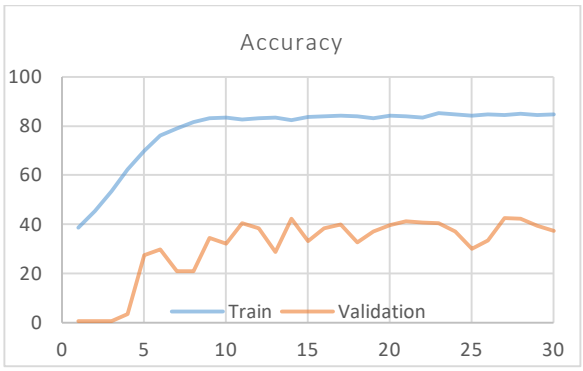

Fig. 15. Model 1 Accuracy

B. C.2

The second model (C.2) used six EfficientNetB0 models combined to create a more distributed model to allow C.2 to specialize. Fig. 16 & 17 shows the results of the first model of C.2 which is the EET model. The EET model resulted in a test accuracy of 91% which is excellent. The optimal number of epochs for training was determined to be 10. After 10 epochs, the accuracy on the training and validation started to decrease and the validation loss started to increase, suggesting the model was overfitting.

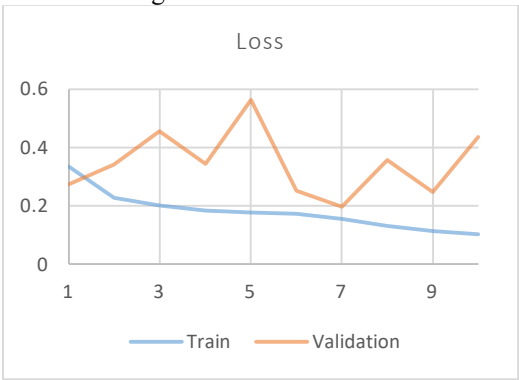

Fig. 16. Model 2 EET Loss

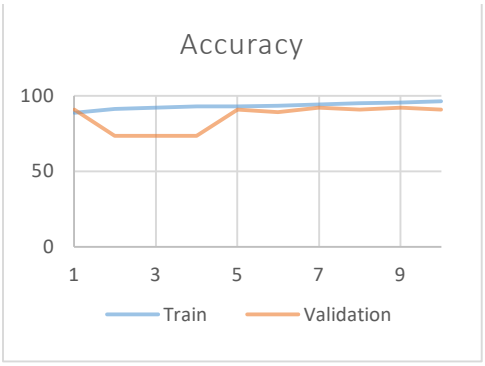

Fig. 17. Model 2 EET Accuracy

The results of the NGT model were similar to the EET model and can be seen in Fig. 18 & 19. The test accuracy of the NGT model was 1% better than EET at 92%. The optimal number of

epochs was 9 for training. After 9 epochs, the model started to overfit.

Fig. 18. Model 2 NGT Loss

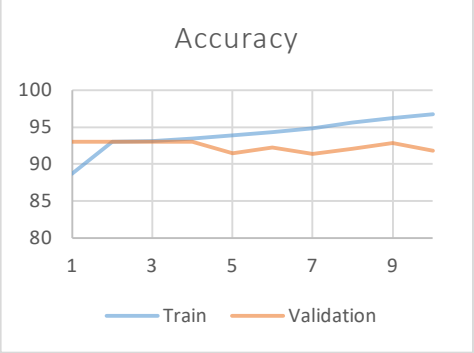

Fig. 19. Model 2 NGT Accuracy

The results of the SG model can be seen in Fig. 20 & 21. The SG model received an accuracy of 97% which is excellent. The optimal number of epochs was 5 for training before it started to overfit.

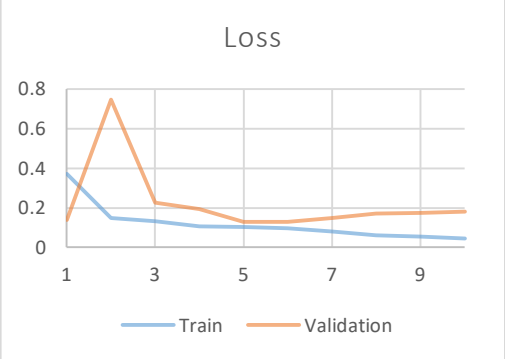

Fig. 20. Model 2 SG Loss

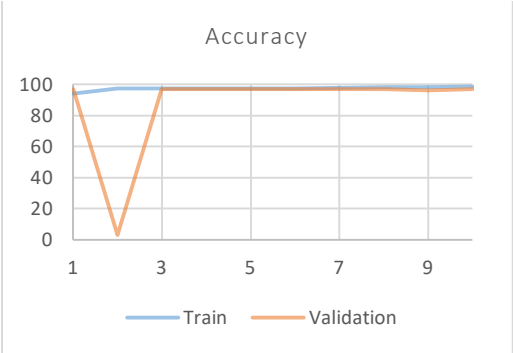

Fig. 21. Model 2 SG Accuracy

The results for the CVC normal model can be seen in Fig. 22 & 23. The CVC normal best accuracy was 71% on the test set which is disappointing; however, compared to the first model, C.1, which only achieved an accuracy of 35%, 71% is a significant improvement.

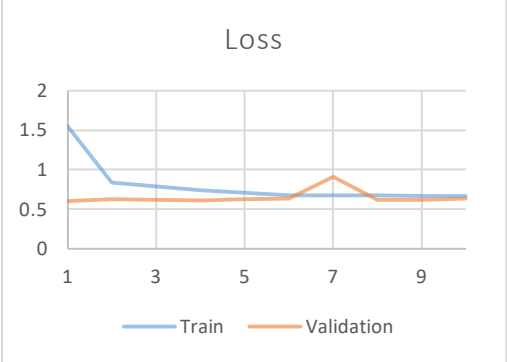

Fig. 22. Model 2 CVC Normal Loss

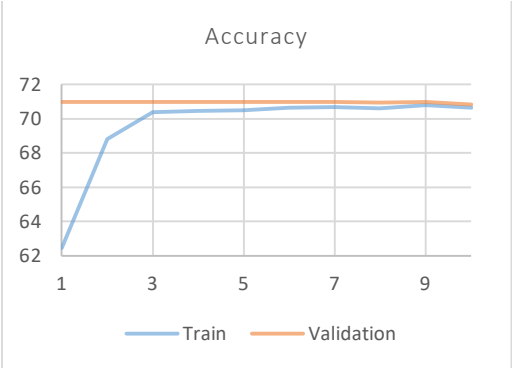

Fig. 23. Model 2 CVC Normal Accuracy

The results of the CVC abnormal model are seen in Fig. 24 & 25. The CVC abnormal best test accuracy was 89% which is very promising. CVC abnormal did better than CVC normal. It is more beneficial to have a higher accuracy for a model that predicts abnormal CVC tube placement than a model that can predicts normal placement.

Fig. 24. Model 2 CVC Abnormal Loss

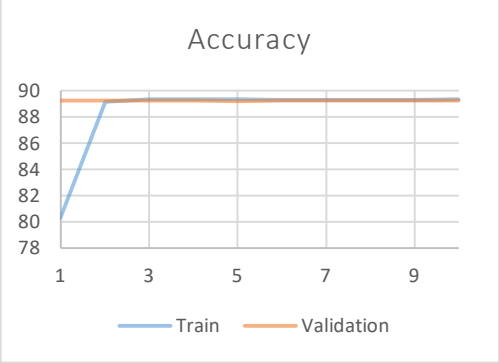

Fig. 25. Model 2 CVC Abnormal Accuracy

The results of CVC borderline model are seen in Fig. 26 & 27. The CVC borderline model achieved a test accuracy of 72% which is only slightly better than the CVC normal model. This is not surprising because they are often predicted the same by radiologists; hence, similar performance was expected. Radiologists in hospitals often have trouble telling the two apart and sometimes label them the same. Therefore, it is not surprising the model is also having a hard time distinguishing them.

Fig. 26. Model 2 CVC Borderline Loss

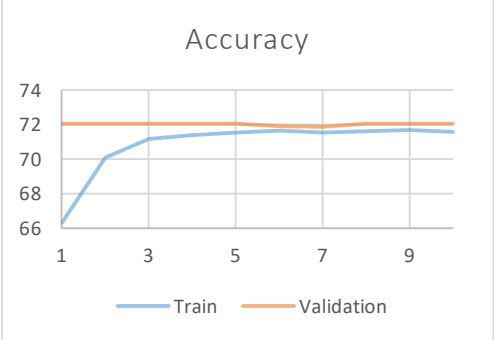

Fig. 27. Model 2 CVC Borderline Accuracy

VI. Conclusion

Two different machine learning models, CNN and EfficientNetB0, were studied to access the practicability of using a machine learning model to read a chest x-ray instead of a radiologist. After performing background research into the current medical standards for tube placement, the data was preprocessed including resizing the images and splitting it into 5 folds. The two different models were trained and tested, and then evaluated by looking at the loss and accuracy. Using these metrics, hyperparameters were tuned such as optimizers, learning rates, and batch size. Both models achieved accuracies between 70-97% in detecting tube placement. Overall, the success obtained for the two different models shows promise in using a machine learning model for accessing tube placement from a chest x-ray. The results suggests that the implementation of machine learning in the medical field could reduce complications in hospitals and lead to faster identification of misplaced tubes which could improve patient care especially since COVID-19 is straining hospital resources.

References

[1] C. Hale, “GE Healthcare's new chest X-ray AI double-checks ventilator tube placement in COVID-19 patients,” FierceBiotech, 24-Nov-2020. [Online]. Available: https://www.fiercebiotech.com/medtech/ge-healthcare-s-new-chest-x-ray-ai-double-checks-ventilator-tube-placement-covid-19. [Accessed: 22-Mar-2021].

[2] A. A. Borkowski, N. A. Viswanadhan, L. B. Thomas, R. D. Guzman, L. A. Deland, and S. M. Mastorides, “Using Artificial Intelligence for COVID-19 Chest X-ray Diagnosis,” Federal practitioner : for the health care professionals of the VA, DoD, and PHS, Sep-2020.

[3] T. Zebin and S. Rezvy, “COVID-19 detection and disease progression visualization: Deep learning on chest X-rays for classification and coarse localization,” Applied Intelligence, vol. 51, no. 2, pp, 2020.

[4] R. Jain, P. Nagrath, G. Kataria, V. S. Kaushik, and D. J. Hemanth, “Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning,” Measurement, 01-Jul-2020.

[5] “RANZCR CLiP - Catheter and Line Position Challenge,” Kaggle. [Online]. Available: https://www.kaggle.com/c/ranzcr-clip-catheter-line-classification.

[6] Tan, Mingxing, and Quoc Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in Int. Conf. on Machine Learning, 2019, pp. 5.

Appendix

Contributed to the Project:

Tina Gholami
- Designed the CNN model.
- Implemented the CNN model.
- Preprocessed the data for the CNN model.
- Participated in the project presentation.
- Worked on Progress Report.
- Worked on Final Report.

Kyle Lacroix
- Created the presentation.
- Background research for the project.
- Formatted the Progress Report & Final Report.
- Participated in the project presentation.
- Worked on Progress Report.
- Worked on Final Report.

Spencer Vecile
- Designed the EfficientNetB0 model.
- Implemented the EfficientNetB0 model
- Preprocessed the data for the EfficientNetB0 model.
- Participated in the project presentation.
- Worked on Progress Report.
- Worked on Final Report.

Bitbucket repository link:

https://bitbucket.org/svecile/kagglecomp/src/main/