MASTER OF TECHNOLOGY

(INTELLIGENT SYSTEMS)

Pattern Recognition Systems

PROJECT REPORT

Prediction System of Pulmonary Fibrosis Progression

Group 04

Team Members

Mohamed Mikhail Kennerley

Loy Pei Xin Veronica

Oh Chun How

Darrel Yee Zhi Quan

Table of Contents

Executive Summary

Artificial Intelligence (AI) and Machine Learning has been used in the medical field since the early 70s, when AI was only 15 years old (Kaul MD, FASGE et al., 2020). Since then, AI has been used to assist medical professionals to make informed decisions, particularly in areas where diagnosis or prediction is less straightforward. One such area would be in predicting the progression of fibrotic lung disease, which is what this project aims to do. Taking patients' physiological and behavioral characteristics as well as Computerized Tomography (CT) scans of their lungs, and processing them through an ensemble of traditional machine learning techniques as well as deep learning techniques, our intelligent system then returns to clinicians a prediction of the decline in patient's lung function in the form of forced vital capacity (FVC). This paper discusses the approach in building this system and delves into the mechanics behind it. Our evaluations show that the system achieves competitive performance. The system is made available for download on GitHub.

1. Background

Pulmonary Fibrosis (PF) is a disease caused by the scarring of lung tissue. This disease currently has no known cure, and prognosis from the time of examination can range from rapid lung failure within several months to long-term stability (Silvia, PhD, 2020). Diagnosis of PF is often challenging due to the varied presentations of the disease, which results in high rates of misclassification. Due to this, clinicians are not frequently able to determine the severity, and consequently the prognosis of a patient. Just in the US alone, about 30,000-40,000 new cases of pulmonary fibrosis are diagnosed each year (Healthline, 2016).

With AI, this project aims to assist doctors to view the decline of the patient's lung function based on the patient's medical information, to help them make more informed decisions and form a baseline of the patient's prognosis.

1.1 Existing Solutions

Currently, the only way to confirm PF with any level of accuracy is through lung biopsy, which extracts a lung sample for analysis. This is not a fool-proof method however, and experts often disagree on the conclusions to the results. In addition, performing a biopsy is slow, which is not ideal given the possibility of rapid patient deterioration. Non-invasive alternatives include taking Computerized Tomography (CT) scans of the patient, as well as conducting breathing and lung function tests and blood tests (British Lung Foundation, 2019). The patient's medical history such as BMI, smoking history, age, and other coexisting diseases often assist in the process (Silvia, PhD, 2020), but due to the wide range of factors, none of these methods are able to extract a concrete long-term prognosis from the patient.

1.2 Project Objectives and Scope

As discussed, existing solutions fail to provide a quick and reasonably accurate method of determining the long term outcome of a patient with suspected PF. Given the abundance of data in this domain (patient medical history, CT scans, etc.), a solution employing AI may be able to solve this problem.

The objective of this project is to predict the severity of a patient’s lung decline over a period of time by **predicting an FVC value** for each week, given initial lung CT scans, FVC readings, and other patient metrics. In addition to predicting the FVC values, the **measure of**

confidence for each predicted FVC value will also be predicted, and this will be provided as a stand deviation from the prediction.

1.3 Data

Data for this project is sourced from a kaggle competition (See **Appendix A**). The data is provided in two forms: 1) Tabular patient history, and 2) patient CT scans.

Tabular Data

	Patient	Weeks	FVC	Percent	Age	Sex	SmokingStatus
0	ID00007637202177411956430	-4	2315	58.253649	79	Male	Ex-smoker
1	ID00007637202177411956430	5	2214	55.712129	79	Male	Ex-smoker
2	ID00007637202177411956430	7	2061	51.862104	79	Male	Ex-smoker
3	ID00007637202177411956430	9	2144	53.950679	79	Male	Ex-smoker
4	ID00007637202177411956430	11	2069	52.063412	79	Male	Ex-smoker
5	ID00007637202177411956430	17	2101	52.868646	79	Male	Ex-smoker

Figure 1: Tabular data for one patient

Figure 1 above shows a sample of the tabular data for a single patient. The data contains several fields described below:

Patient: This is a unique ID for the patient.

Weeks: The number of weeks passed from which the measurements in the entry was taken. Week 0 corresponds to the week that the CT scan was taken. Accordingly, negative weeks represent entries gathered before the CT scan.

FVC: Forced Vital Capacity represents the total volume of air that can be exhaled by the patient. This measurement is taken using a device called a spirometer.

Percent: This is a computerized field which approximates the patient's FVC as a percentage of the typical FVC for a person with similar characteristics.

Age, Sex: The patient's age and sex.

SmokingStatus: This classifies the patient as either a current smoker, ex-smoker, or never smoked.

A total of 1549 entries were provided in total with 176 patients represented, each patient having 7-9 individual weeks of data. An interesting and potentially challenging feature of the data is the significant and varied gaps in weeks between FVC measurements. This reflects real world medical data where schedules for patient clinical visits (where measurements are taken) are often irregular.

Image Data

The CT scan images are provided as DICOM (Digital Imaging and Communications in Medicine) images which, along with the scans themselves, also embed metadata about the scans such as image scaling, position of the scans, and gap between slices. The scans are made of 2D image slices of the patient's lung (see **Figure 2a**), which come together to form a 3D representation of the full lung. The number of slices required for each patient is decided by the measuring clinician, and can vary greatly. There are 176 DICOM sets provided, one for each patient.

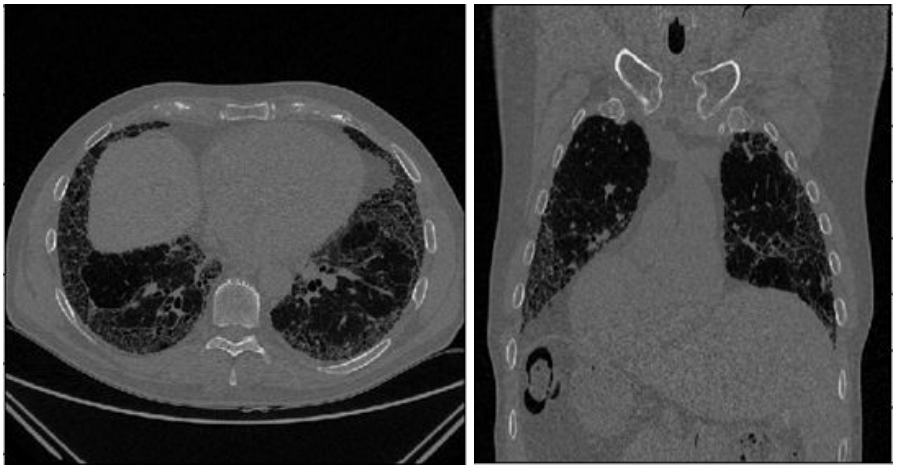


Figure 2a (left) & 2b (right): The slices are provided cross sectional to the body (Fig. 2a). The full 3D scan can also be manipulated to show other perspectives, such as the front (Fig. 2b).

Due to the different sources the CT scans come from, there is also a wide variation in other properties such as slice gap, greyscale composition, and resolution between patient scans (see **Figure 3**).

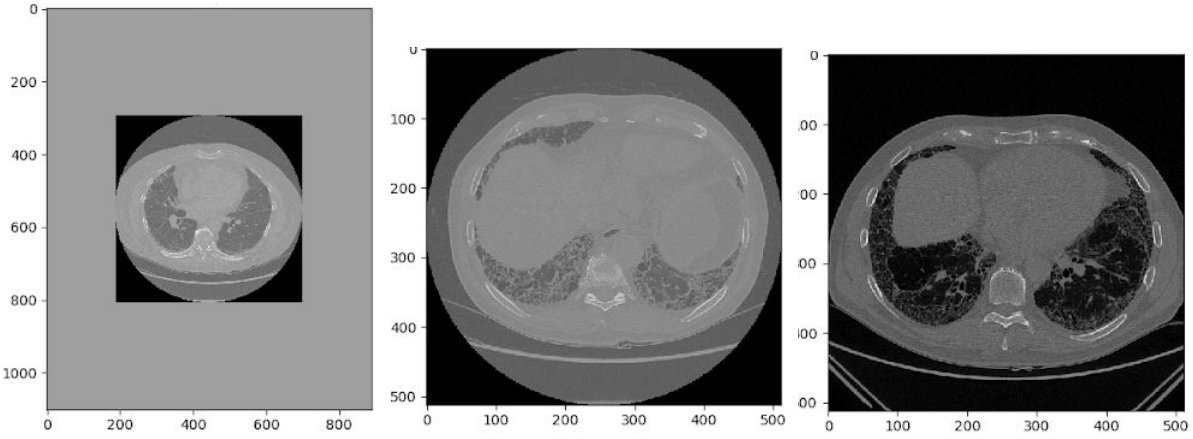


Figure 3a (left), 3b (middle), and 3c (right): Example scan samples. Wide variations in colour composition and the presence of border pixels (Figure 3a & 3b) can be observed.

2. System Architecture

The architecture is illustrated in **Figure 4**. The system consists of 4 machine learning sub-models and an ensemble model which aggregates the results and determines the final output. The four models are as follows:

- MLP model
- Huber linear model
- EfficientNet model
- 3D CNN model

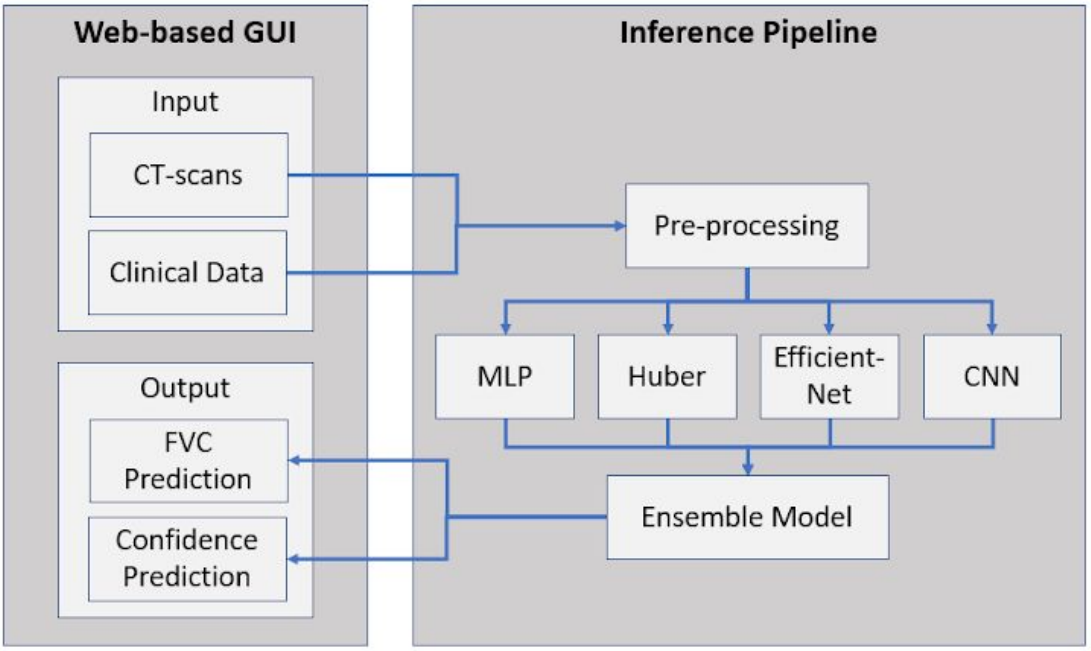


Figure 4: System Architecture

Inputs by the user are accepted through a web-based GUI, which also displays the prediction results.The inputs first go through the necessary pre-processing steps before being fed into the individual models. The model outputs are then passed through the ensemble model before being returned to the user as a 133-week forecast of the patient's FVC and the confidence in each week's forecast.

3. System Design and Implementation

3.1 Data Preprocessing

This section describes the tabular data pre-processing steps common to all models. The pre-processing steps for the DICOM image data will be described in the sections for the models that require it. From the tabular data presented in Section 1.3, the following steps were taken during pre-processing:

One-hot Encoding: The ‘Sex’ and ‘SmokingStatus’ features are categorical, and were thus encoded in One-hot format.
Engineered Features: A composite feature estimating the height of the patient could be calculated from the ’FVC’ and ‘Age’ features through the following formula:

$$height_{female} = \frac{FVC}{(21.78 - 0.101 \times Age)}$$

$$height_{male} = \frac{FVC}{(27.63 - 0.112 \times Age)}$$

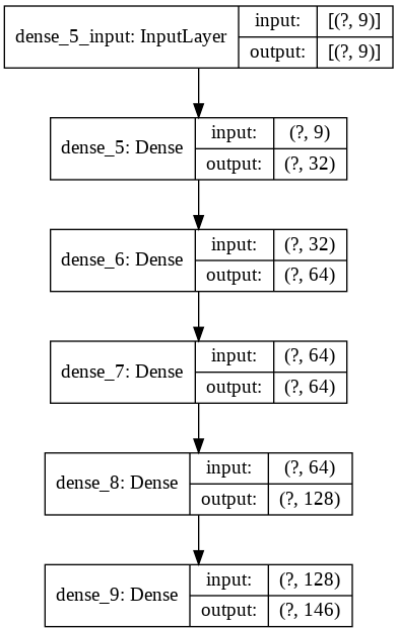
Reformatting: For the MLP and CNN models, an additional step was taken to format the data. The dataset was grouped by patient, and the ‘Week’, ‘FVC’, and ‘Percent’ features were merged into one entry each per patient, as a list (Refer to ‘Week’, ‘FVC’ and ‘Percent’ columns in **Figure 5**).

	Patient	Weeks	FVC	Percent	Age	Male	Female	Currently smokes	Ex-smoker	Never smoked
0	ID00007637202177411956430	[-4, 5, 7, 9, 11, 17, 29, 41, 57]	[2315, 2214, 2061, 2144, 2069, 2101, 2000, 206...	[58.2536487166583, 55.7121288374434, 51.862103...	79	1.0	0.0	0.0	1.0	0.0
1	ID00009637202177434476278	[8, 9, 11, 13, 15, 22, 33, 45, 60]	[3660, 3610, 3895, 3759, 3639, 3578, 3625, 339...	[85.28287818063191, 84.11781153881999, 90.7586...	69	1.0	0.0	0.0	1.0	0.0
2	ID00010637202177584971671	[0, 1, 3, 5, 7, 13, 25, 37, 54]	[3523, 3373, 3327, 2993, 3030, 3103, 2993, 247...	[94.72467197246719, 90.6915465691547, 89.45472...	60	1.0	0.0	0.0	1.0	0.0
3	ID00011637202177653955184	[6, 7, 9, 11, 13, 19, 32, 43, 58]	[3326, 3419, 3541, 3502, 3410, 3477, 3269, 334...	[85.9875904860393, 88.3919338159255, 91.546018...	72	1.0	0.0	0.0	1.0	0.0
4	ID00012637202177665765362	[33, 35, 37, 39, 41, 47, 58, 71, 87]	[3418, 3759, 3276, 3443, 3268, 3449, 3324, 323...	[93.7260063617418, 103.07666995722299, 89.8321...	65	1.0	0.0	0.0	0.0	1.0

Figure 5: Input data after preprocessing.

3.2 MLP Model

The MLP model works with only the tabular data. A summary of the model is shown in **Figure 6**.



Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 32)	320
dense_6 (Dense)	(None, 64)	2112
dense_7 (Dense)	(None, 64)	4160
dense_8 (Dense)	(None, 128)	8320
dense_9 (Dense)	(None, 146)	18834

Total params: 33,746
Trainable params: 33,746
Non-trainable params: 0

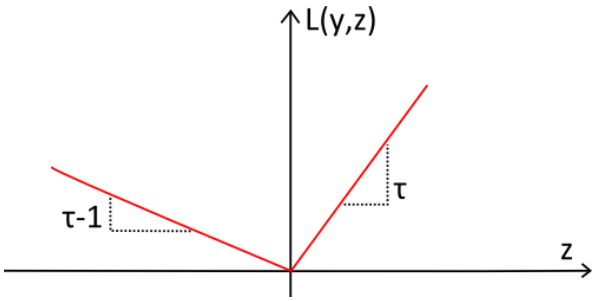
Figure 6a (left) and 6b (right): Summary of the MLP design.

The model takes the first element of the *'Weeks'*, '*Percent*', and '*FVC*' features as input, along with the *'Age'*, and one-hot encoded *'Male'*, *'Female'*, '*Currently smokes*', '*Ex-smoker*', and '*Never smoked*' features. The model is then trained on the remaining elements from *'Weeks'* and '*FVC*' as labels. The output to the model is an array with 146 elements per sample; this can be set to an arbitrary number of weeks that require an FVC forecast.

The MLP model does not predict confidence. Instead, it assumes that the confidence for each week is the same. The tuning of this constant value is described in the following section.

3.2.2 Loss Function and Evaluation Method

As the loss function needs to take into account both FVC and confidence, a simple MSE or MAE loss is insufficient. The loss function has to reward accurate and confident forecasts, but penalize confident forecasts that turn out inaccurate, and vice versa. The metric selected was the *pinball loss* as described below, where L_T is the loss value, (y-z) is the FVC error, and τ is the predicted confidence in standard deviation.



$$L_\tau(y,z) = (y-z)\tau \quad \text{if } y \geq z$$
$$= (z-y)(1-\tau) \quad \text{if } z > y$$

Figure 7: The relationship between the pinball loss function, and the error and confidence.

Using this loss function, the static confidence value can be iteratively optimized until the best pinball loss result is generated.

3.3 ResNet - 3D CNN

3.3.1 Data Preparation

This model trains on both tabular and CT scan data. Due to the variations in scan properties between patients, a pre-processing step is required before the scans can be trained on. The *pydicom* library was used to read and process the DICOM CT scans.

File ordering and image extraction

The first part of preparing the image data is ensuring the images are loaded in the correct order; this is done by reading the DICOM images in descending order based on the names[1]. Once the files have been sorted, the dcm images are loaded to memory as 2D pixel arrays. To manage memory footprint, only a sample of 50 images max is read per patient.

Cropping, stacking and resizing

Once the images have been extracted, the aspect ratio of the image may be different from each other, another common problem is borders surrounding some of the images. The first part of this process is sending the image to a cropping function that detects these borders and only extracts the relevant scan data. The cropped images are then stacked to create a 3D pixel array. Finally, the array is resized to (48,48,48) *uint8* pixels for the neural net to take in as input.

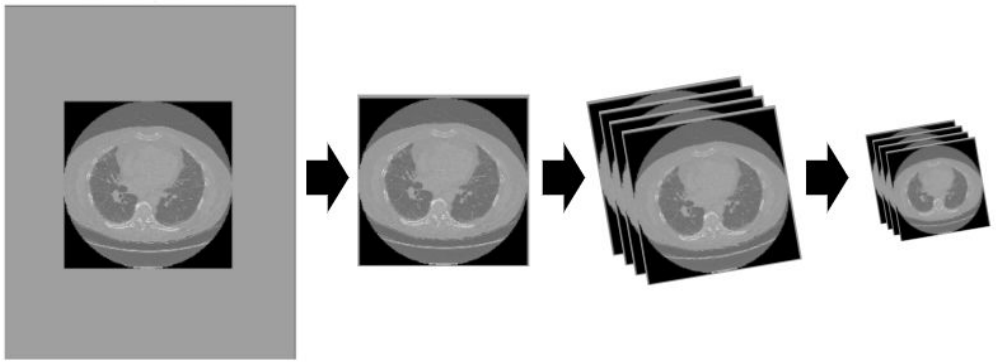


Figure 8: Illustration of the DICOM preprocessing steps..

[1] The DICOM files are named by their position in the 3D stack, for example, '*1.dcm*' and '*2.dcm*'.

3.3.2 Approach

Exploring Methods

Due to the nature of 3D images, two main options presented itself to train the models with. The first option was using a CNN-LSTM to take the images and run them through the model as a sequence to generate a prediction of the FVC, this involved running each 2d image through a series of standard CNN layers and feeding the flattened output into the LSTM layers.

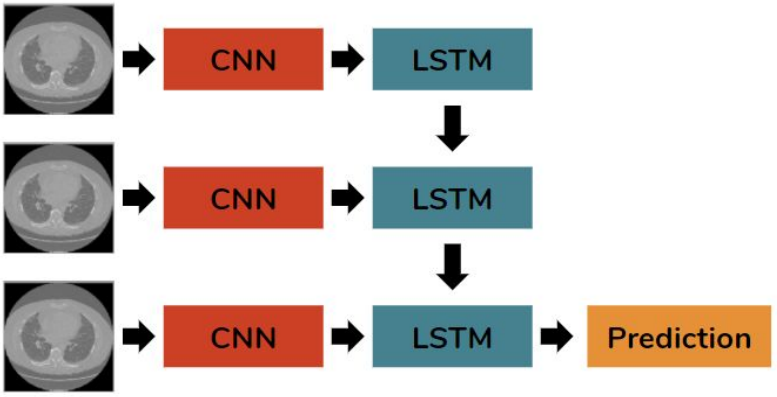


Figure 9: Summary of the CNN-LSTM design..

The second method was to use 3D convolutions that directly work with the 3D images. This was the method that was ultimately selected due to its performance relative to the other methods.

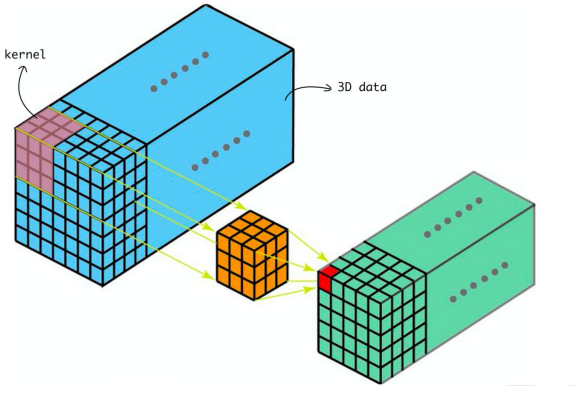


Figure 10: Direct 3D convolution.

To provide more context and information to the model, a Y shaped neural network was used with the main portion being the image model and the other input being the tabular data before combining the extracted information to provide a prediction.

Residual Neural Network

One of the problems with working with a smaller dataset is that a deep neural network may overfit the data or lead to worse performance than a shorter data set. To solve this, we can continuously test the model performance with different depth neural networks or create residual neural networks.

The ResNet contains 4 main building layers that are used to create a simple ResNet block or a down ResNet block that down-samples the data.

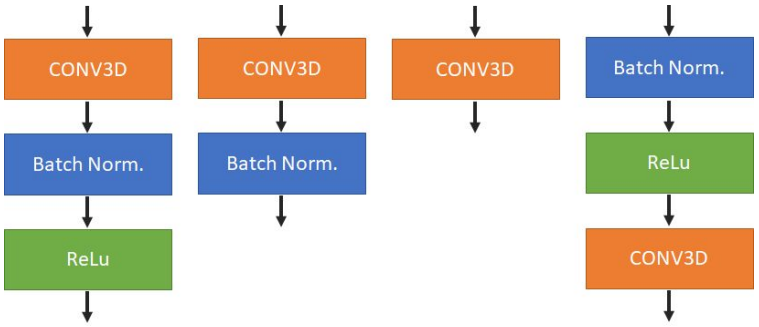


Figure 11: 4 Main building layers for ResNet creation. From left to right: ResLyr1, ResLyr2, ResLyr3 and ResLyr4

The Simple ResNet Block is made up of ResLyr1+ResLyr2 merged with the input before going into a ReLu activation layer. The Down ResNet Block is made up of ResLyr1, with an increased stride of 2,+ResLyr2 merged with the input going through ResLyr3, with an increased stride of 2, before going into a ReLu activation layer.

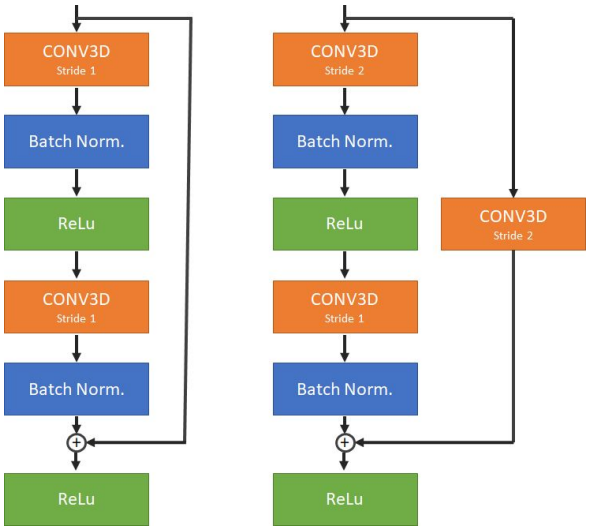


Figure 12: Simple ResNet block (left) and Down ResNet block (right).

The Simple and Down ResNet blocks are then used to build the model that the 3d images will go through. The model will go through ResLyr1 and then through 3 Down ResNet Blocks

before going through a MaxPooling layer and Flattening layer, after flattening it will go through 2 dense layers with 1024 and 256 nodes before going into the next portion of the model.

Y-Shaped Neural Network

The 2nd input of the Y-shaped neural network takes in the tabular data in the same format as the MLP model; however, in this case it is standardized before feeding into the Y-shaped NN. The tabular input goes through 2 dense layers with 64 and 256 nodes before combining with the output of the ResNet model. The full model is summarized in **Figure 13**.

The 2 outputs are concatenated and put through a series of dense layers with 512, 1024, 2048 and 512 nodes with ReLu activation before going through the output layer with 146 nodes and a linear activation. The loss function is similar to that used by the MLP.

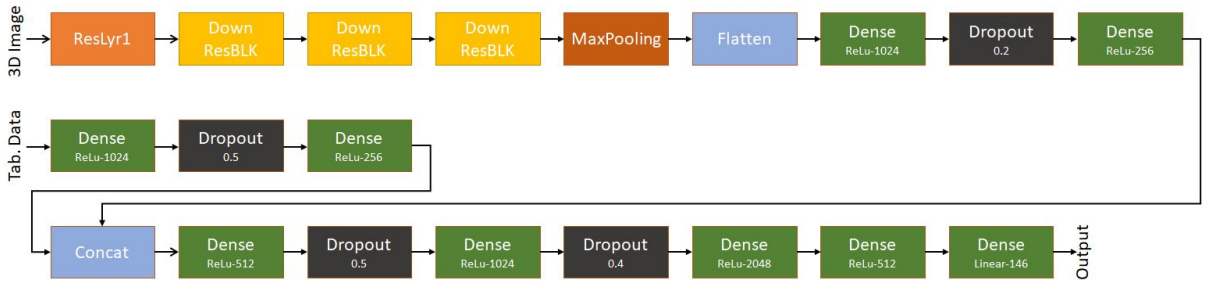


Figure 13: Full Y-shaped Residual Neural Network Model using both tabular and 3D Image Data

Smoothing the Output

A problem with the data is that its distribution is most concentrated for earlier weeks, and tapers off as the number of weeks get larger; this property can be observed in **Figure 14**. Most of the weeks beyond week 105 are unrepresented in the data. Due to this, the model cannot train adequately to be accurate for those week ranges, which results in erratic predictions (see **Figure 15a**).

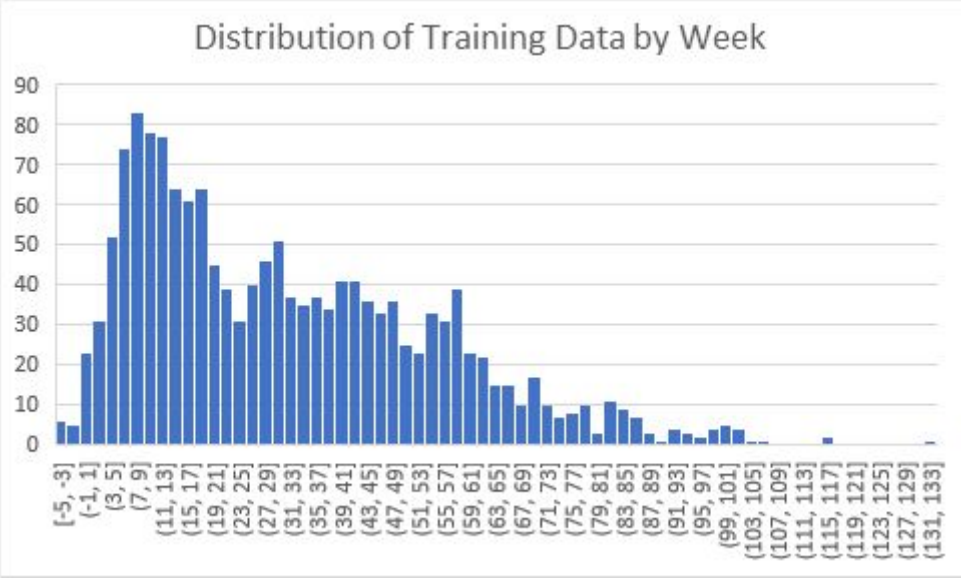


Figure 14: Distribution of training data, binned by the 'Week' feature. Note the decreasing occurrence of higher week data.

To solve this, a stable range of the output is used to create a linear regression to use as a base to smooth the output. The range to use is selected by getting the index of the output prediction when it starts going above 1001 FVC for the start and when it next goes below 1001 FVC for the end. The value of 1001 was chosen as the FVC of the patient should not reach lower than 1000.

The line and actual predicted output is then merged based on the constraint: *if the predicted FVC value is more than +/- of the line value, replace the predicted FVC value with line value*. This constraint allows us to produce accurate predictions for well-represented weeks while smoothing the areas where the model does not perform as well.

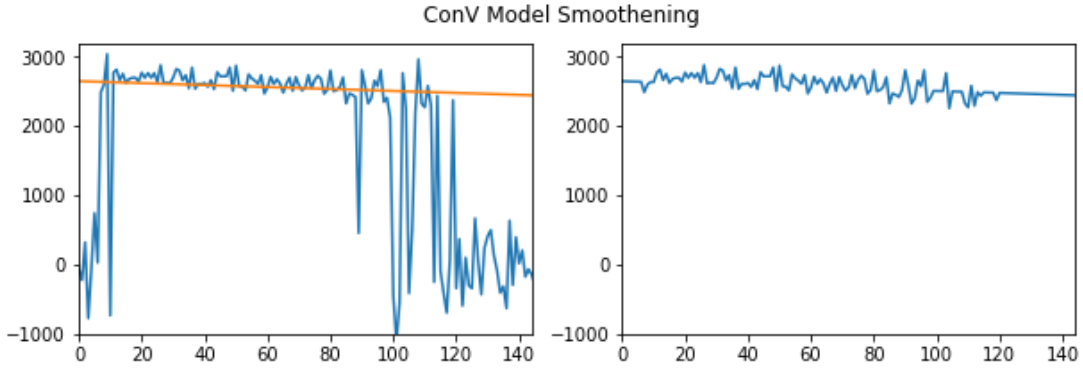


Figure 15a(left) & 15b(right): Graphs of uncorrected (15a) and corrected (15b) results.

3.4 Gradient Boosted Huber Regression

Motivation

In exploring the data, we looked into how the FVC value declines for a number of patients in the dataset, and found that the decline in FVC values largely follows a linear downward trend despite spikes in some weeks (see **Figure 16)**. This suggests that, on top of NN-based models, a linear model could also be useful in predicting FVC values across the weeks for the patients. An additional benefit of a linear model over their NN-based counterparts is better extrapolation over weeks not covered in the training data.

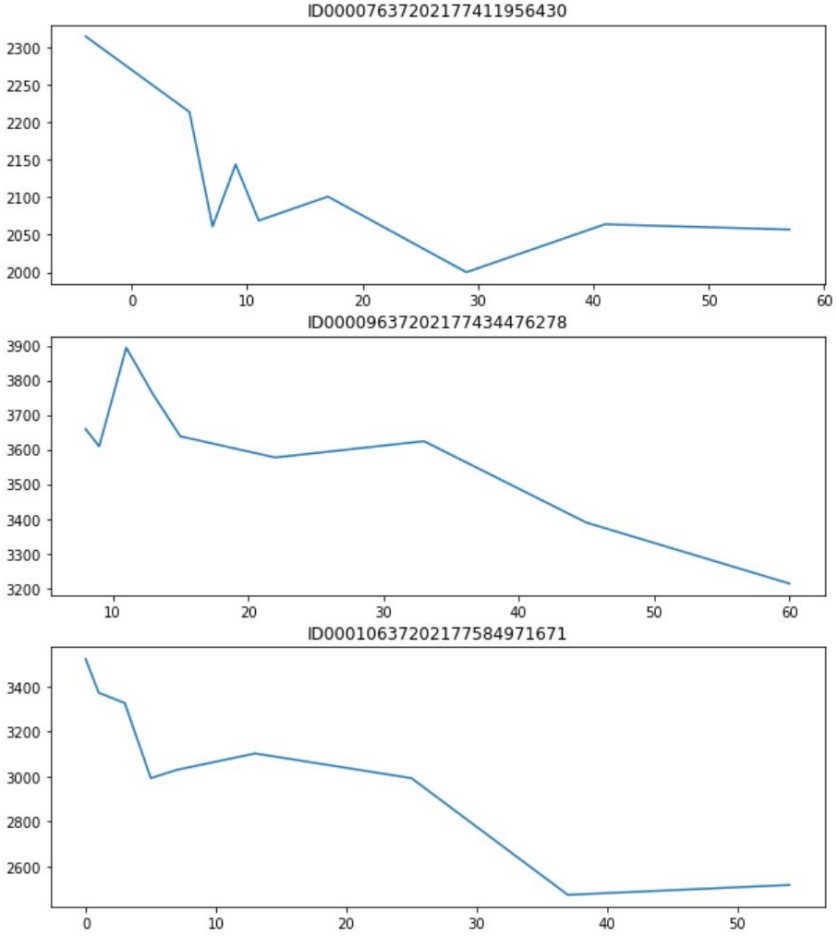


Figure 16: Three examples of patient FVC data.

Huber Regression

Due to the noise in the dataset, a linear regression that was robust to outliers was picked. The Huber Regressor is a more robust form of linear regressor because it additionally optimizes for the absolute loss on top of optimizing for squared loss. Because it does not purely square all errors, Huber regressor reduces the penalty imposed on outliers and was deemed suitable for the tasks at hand.

The Huber loss function is as follows. The first equation shows the square difference between actual and predicted FVC value, while the second equation shows the absolute difference between the actual and predicted FVC value. $\alpha \in \mathbb{R}^+$ is a hyperparameter.

$$\mathcal{L}(y, \hat{y}) = \begin{cases} (y - \hat{y})^2 & \dots \quad |y - \hat{y}| \leq \alpha \\ |y - \hat{y}| & \dots \quad |y - \hat{y}| > \alpha \end{cases} \quad (1)$$

Gradient Boosting

The Huber regressor, used by itself and without properly tuning the alpha parameter, is more prone to underfitting than overfitting. To make the Huber regressor more robust, we used gradient boosting to train the models in a gradual, additive and sequential manner. That is, at each stage an additional Huber regressor is fit on the negative gradient of the given loss function, effectively overfitting on the errors. The end result is an ensemble of Huber regressors that is robust to outliers, and yet does not over generalize.

Quantile Regression

To supplement the prediction from the gradient-boosted Huber regressor, quantile regression was also used to provide the confidence interval of the prediction. The lower confidence was set at 25th percentile and the upper confidence at 75th percentile, effectively giving a 50 percent probability interval. While the coverage is not overly-extensive, especially in the realm of medical domain, this was deemed sufficient because the Huber Regressor ensemble was to be further ensembled with three other equally well performing models.

Quantile regressions minimize the quantile loss in predicting a certain quantile and is defined as:

$$L_\gamma(y, y^p) = \sum_{i=y_i<y_i^p} (\gamma - 1).\,|y_i - y_i^p| + \sum_{i=y_i \geq y_i^p} (\gamma).\,|y_i - y_i^p|$$

where, γ is the required quantile and has value in range (0, 1).

3.5 EfficientNet (Based on Linear Decay)

3.5.1 Introduction to EfficientNet

In 2019, Google AI proposed a new model scaling method that uses a simple yet highly effective compound coefficient to uniformly scale all dimensions of depth/width/resolution of convolutional neural network architecture in a more structured manner. Unlike conventional approaches that arbitrarily scale network dimensions, such as width, depth and resolution, this method uniformly scales each dimension with a fixed set of scaling coefficients. Consequently, Google AI has developed a family of models, called EfficientNets using this scaling method and recent progress on AutoML, which superpass state-of-the-art accuracy with up to 10 times better efficiency.

Compound Model Scaling: A Better Way to Scale Up CNNs

While scaling individual dimensions improves model performance, it is observed that balancing all dimensions of the network—width, depth, and image resolution—against the available resources would best improve overall performance.

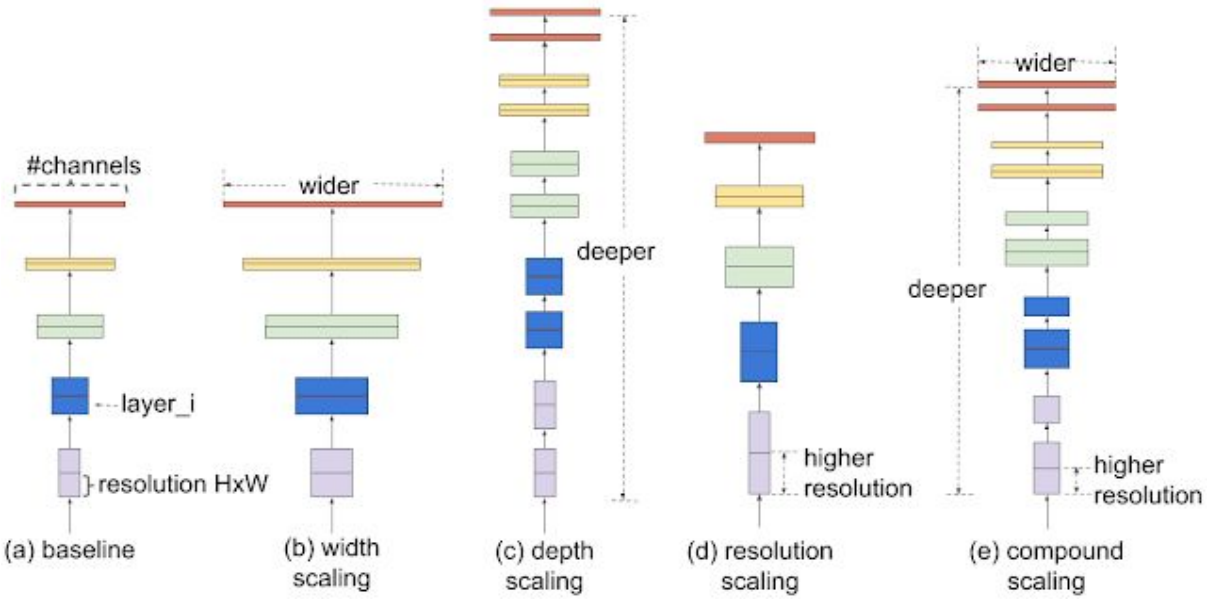


Figure 17: Comparison of different scaling methods. (b)-(d):Conventional scaling methods that arbitrarily scale a single dimension of the network; (e) Compound scaling method uniformly scales up all dimensions in a principled way..

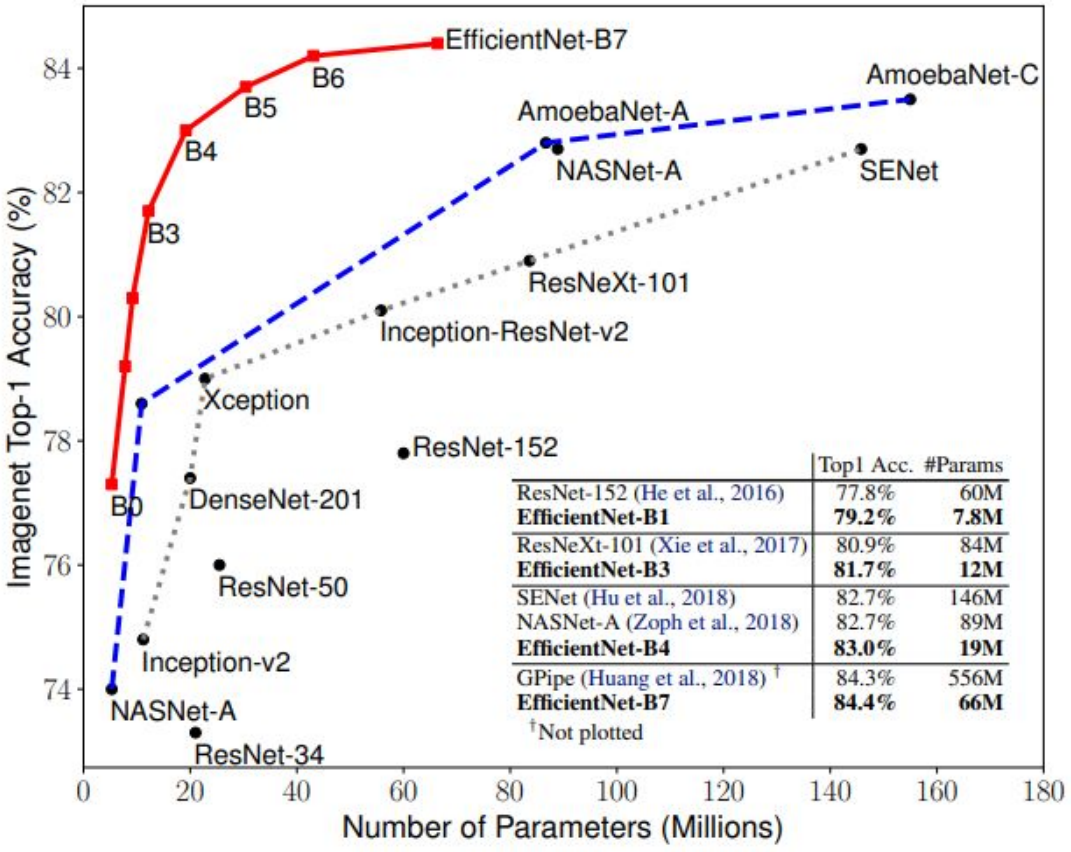


Figure 18: Comparison of model size with ImageNet accuracy with other models.

EfficientNets consists of 8 models, B0 to B7. After testing across the models, the B4 model was chosen to be used in this solution, based on the input size, required computing resources, accuracy and efficiency gain across the models.

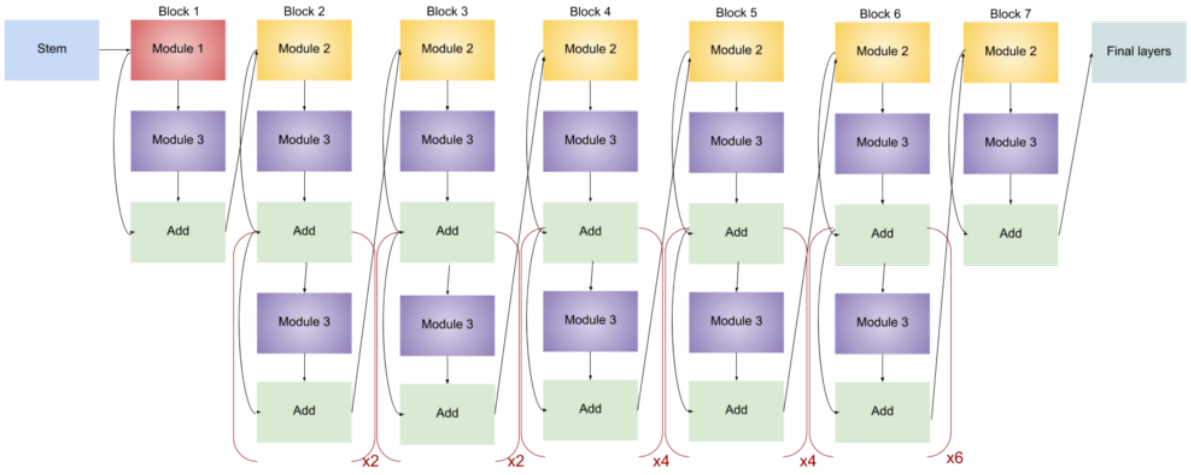


Figure 19: Architecture of B4 EfficientNet model

3.5.2 Approach

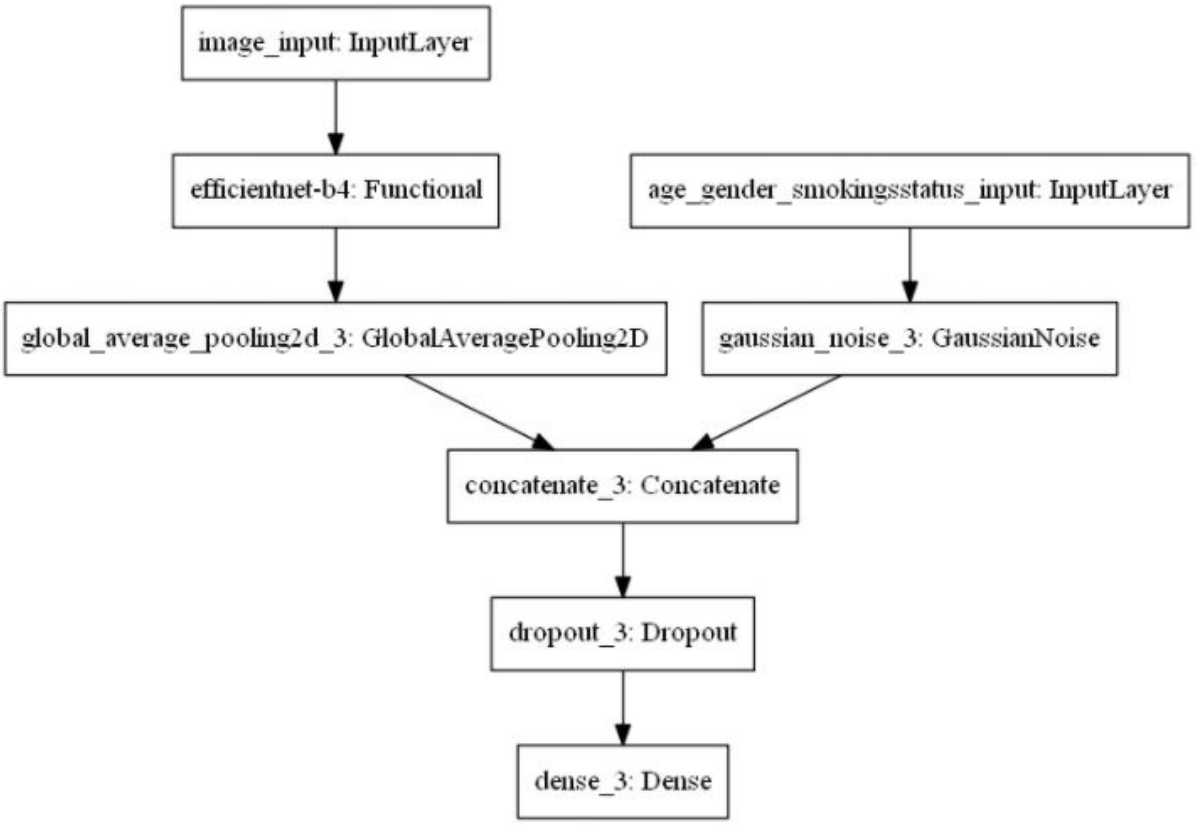
The approach for this model is to predict the progress of loss of lung functions over time using the linear decay function. It is assumed that the progress of the loss of lung function is similar to a linear decay. Hence, the gradient will be predicted using the neuralnet model and prediction of week-by-week FVC and confidence are calculated using the output gradient in the formula below:

$$\mathbf{FVC_{week(x)} = FVC_{input} + Gradient} \times abs\,\mathbf{(X)}$$

$$\mathbf{Confidence_{week(x)} = Percent - Gradient} \times abs\,\mathbf{(X)}$$

where X = No of weeks from Input week; FVC_{input} = FVC on Input week

DCM images are fed into the EfficientNet and then combined with tabular data input similar to a Y-shaped neural net resulting in a single output, which is the gradient. The summary of the model is shown below in **Figure 20**.



Layer (type)	Output Shape	Param #	Connected to
image_input (InputLayer)	[(None, 128, 128, 1)	0	
efficientnet-b4 (Functional)	(None, 4, 4, 1792)	17672952	image_input[0][0]
age_gender_smokingsstatus_input	[(None, 3)]	0	
global_average_pooling2d_3 (Glo	(None, 1792)	0	efficientnet-b4[0][0]
gaussian_noise_3 (GaussianNoise	(None, 3)	0	age_gender_smokingsstatus_input[0
concatenate_3 (Concatenate)	(None, 1795)	0	global_average_pooling2d_3[0][0] gaussian_noise_3[0][0]
dropout_3 (Dropout)	(None, 1795)	0	concatenate_3[0][0]
dense_3 (Dense)	(None, 1)	1796	dropout_3[0][0]

Total params: 17,674,748
Trainable params: 17,549,548
Non-trainable params: 125,200

Figure 20: Summary of the EfficientNet model design.

3.5.2.1 Image Data Input

For the image data input, only the images between 0.15 and 0.8 quantile in the set of DCM images for a particular patient are taken because the top and bottom slices of the CT scans do not show much of the lung in the cross sectional image. The selected images are resized to 128x128 resolution before feeding to the EfficientNet as input. Global Average Pooling2D is applied to the output of the EfficientNet.

3.5.2.2 Tabular Data Input

An array consists of 'Age', Sex', 'Smoking Status' features is used as the tabular data input. 'Age' feature is normalized while 'Sex' and 'Smoking Status' features are labelled categorically as follows:

Sex	Male	0
	Female	1
Smoking Status	Never smoked	0
	Ex-smoker	1
	Currently smokes	2

Gaussian noise is used to add noise to this data input to prevent overfitting and improve model generalization because the dataset is small.

3.5.2.3 Gradient Output

Both image data input and tabular data input are then concatenated before dropout is applied for the same reason to improve the model robustness. Finally, it is connected to a dense layer to produce the final node output which is the gradient constant. The output gradient is used in the linear decay formula to generate week by week prediction of FVC and confidence.

3.6 Ensemble Model

The ensemble model applies a weighted average to each model's output to find a final weighted sequence of FVC and Confidence values (See **Figure 21** for illustration).

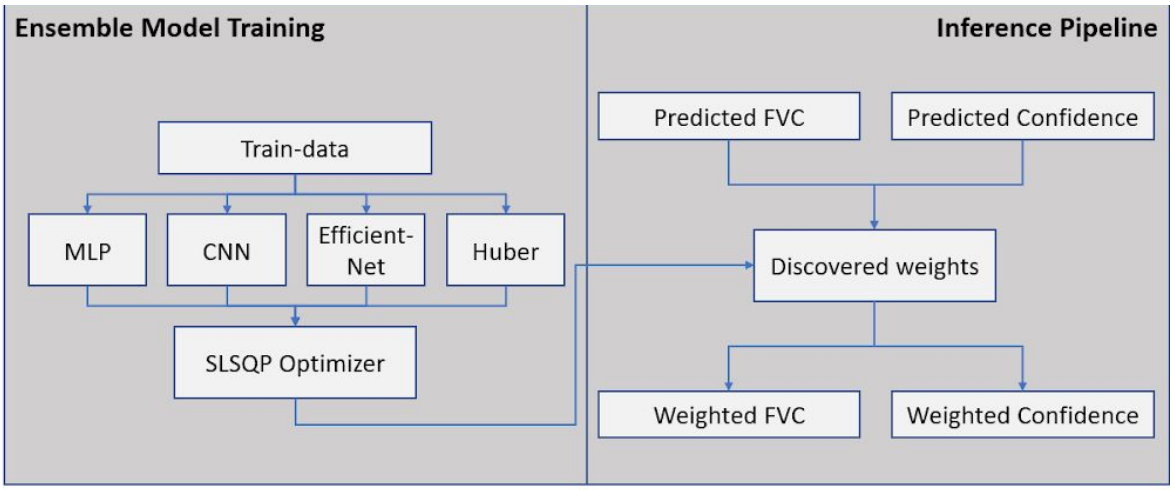


Figure 21: An illustration of the ensemble model and the weighted averaging of the inferred results.

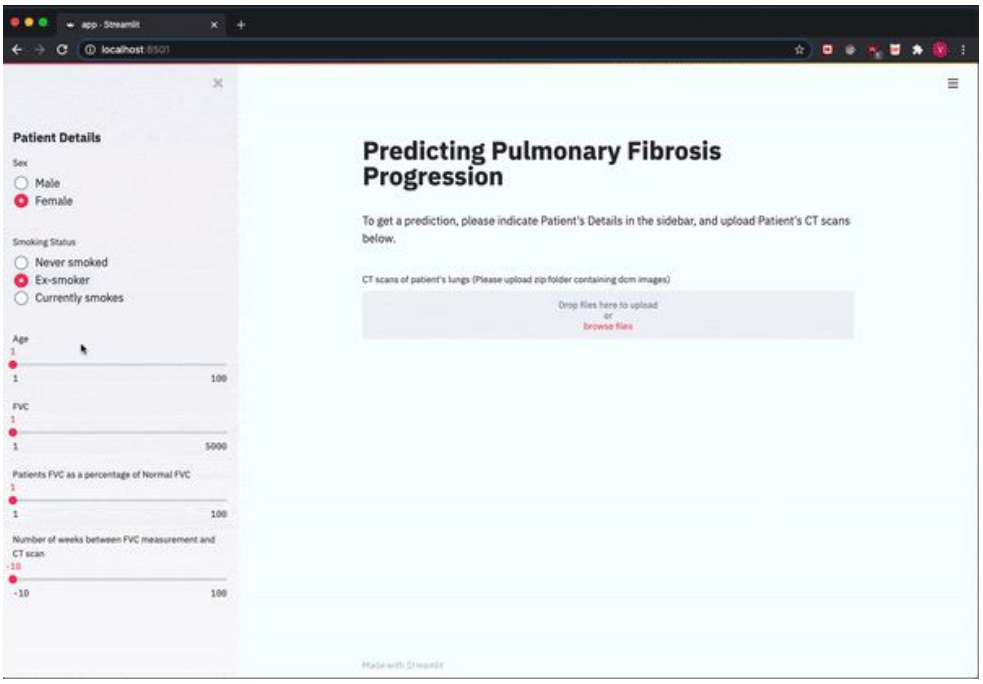
The weights were discovered using a Sequential Least Squares Programming (SLSQP) minimizer from scipy; SLSQP was used due to its computational efficiency and because it can accept constraints to its parameters[2]. The objective function to the minimizer is set as the delta between prediction and label. The minimizer takes in the prediction of each model on the training set, and finds the weights that can give the best score on the objective.

3.7 Graphical User Interface

The user interface is intended to be simple, interactive and provides a quick prediction to clinicians who are either seeking a second opinion to their FVC prediction, or looking for a rough benchmark for their prediction. The user interface supports prediction for one patient at a time. Upon entering the patient details and uploading a zip file containing CT scans of patient lungs in .dcm format, a graph of the next 133 weeks of fvc predictions, along with a table of the weekly predictions will be shown. These patient details and dcm images required in the user interface follows the features present in our training set. Depending on how set in stone the patient details are, e.g. concerning FVC, percent, currently smokes or ex-smoker, clinicians can further tweak these parameters to observe the change in FVC predictions given a change in any of these inputs. Finally, clinicians can download the graph or print the table of predictions for further reference.

[2] Constraints are needed to cap the sum of weights to 1.

The application was deployed on a virtual machine, using Amazon Web Services, and can be accessed via the url: http://54.186.100.151:8501/. The virtual machine has a Ubuntu operating system, with specifications 1vCPU and 2GB memory. Due to memory limitations, the model deployed on this virtual machine is a simple version of the ensemble: only the MLP model, EfficientNet, and Huber regression were used. This specification allows a single user to use the application at any one time, and is deemed sufficient for a proof-of-concept. The application will be hosted on AWS for a period of 2 months, till 31st December 2020.



.

4. Results

In order to validate our models' performance, we submitted it to the kaggle competition which provided the data. Submitted models are evaluated against a private dataset never seen before. The kaggle competition uses a modified Laplace Log Likelihood (LLL) metric to score submissions, as described below:

$$\sigma_{clipped} = max(\sigma, 70),$$

$$\Delta = min(|FVC_{true} - FVC_{predicted}|, 1000),$$

$$metric = -\frac{\sqrt{2}\Delta}{\sigma_{clipped}} - \ln(\sqrt{2}\sigma_{clipped}).$$

σ represents the confidence, which is clipped at 70 ml to reflect uncertainty when the FVC values were measured. Δ represents the error between predicted FVC and actual FVC, and is thresholded at 1000 ml to avoid large errors adversely penalizing results. A larger LLL metric indicates a more accurate FVC and confidence prediction. The LLL acts similar to pinball loss by evaluating both predicted confidence and FVC together.

The ensemble model scored **-6.87** on the private dataset, which represents an approximate standing of 250 out of 2100 participants. The first place in the competition scored -6.83, while the median score was -6.89. The close scores of the top submissions, along with our position, gives confidence that our model is performing close to the best possible for the data provided.

5. Conclusion

With diseases like Pulmonary Fibrosis, tools, such as what this project has presented, will play an important part when doctors and clinicians work with patients to determine their prognosis.

This project has presented an aid for medical professionals in a simple and intuitive package, via a graphic-user interface, that provides them with a predicted prognosis to assist in caregiving, and to afford patients a peace of mind towards their condition.

Based on the performance of the model in the private data set, it shows that while the model created may not have been the best created, it made full use of the data available and performed within the top 15%. With further medical detail of the patient as well as a bigger dataset, more robust and accurate models could be created.

Also, with more and better quality data, the model can possibly be made to perform even better. The current dataset only involves the participation of 176 patients, and several deficiencies are noted in the data such as an underrepresentation of current smokers among the sample patients and lack of week/FVC data beyond week 100. More patient metrics, such as weight, BMI, blood pressure, heart rate, and ethnicity can be included in the dataset as well. It is expected that having more data will allow a more accurate long-term forecast, as well as consider edge cases among the subpopulation as well.

Through the completion of this application, we have also achieved the objectives of the Practice Module which is to demonstrate our proficiency across the skills that we learned in the course in solving real world problems.

Appendices

Appendix I: ResNet CNN Model

Model: "CNN"

Layer (type)	Output Shape	Param #	Connected to
original_img (InputLayer)	[(None, 48, 48, 48,	0	
Inpt_conv3D (Conv3D)	(None, 48, 48, 48, 3	2624	original_img[0][0]
Inpt_bn (BatchNormalization)	(None, 48, 48, 48, 3	128	Inpt_conv3D[0][0]
Inpt_relu (Activation)	(None, 48, 48, 48, 3	0	Inpt_bn[0][0]
Stg1_Blk1_Res1_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Inpt_relu[0][0]
Stg1_Blk1_Res1_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk1_Res1_conv3D[0][0]
Stg1_Blk1_Res1_relu (Activation	(None, 24, 24, 24, 3	0	Stg1_Blk1_Res1_bn[0][0]
Stg1_Blk1_Res2_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Stg1_Blk1_Res1_relu[0][0]
Stg1_Blk1_lin_conv3D (Conv3D)	(None, 24, 24, 24, 3	1056	Inpt_relu[0][0]
Stg1_Blk1_Res2_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk1_Res2_conv3D[0][0]
tf_op_layer_Stg1_Blk1_add (Tens	[(None, 24, 24, 24,	0	Stg1_Blk1_lin_conv3D[0][0] Stg1_Blk1_Res2_bn[0][0]
Stg1_Blk1_relu (Activation)	(None, 24, 24, 24, 3	0	tf_op_layer_Stg1_Blk1_add[0][0]
Stg1_Blk2_Res1_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Stg1_Blk1_relu[0][0]
Stg1_Blk2_Res1_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk2_Res1_conv3D[0][0]
Stg1_Blk2_Res1_relu (Activation	(None, 24, 24, 24, 3	0	Stg1_Blk2_Res1_bn[0][0]
Stg1_Blk2_Res2_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Stg1_Blk2_Res1_relu[0][0]
Stg1_Blk2_Res2_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk2_Res2_conv3D[0][0]
tf_op_layer_Stg1_Blk2_add (Tens	[(None, 24, 24, 24,	0	Stg1_Blk1_relu[0][0] Stg1_Blk2_Res2_bn[0][0]
Stg1_Blk2_relu (Activation)	(None, 24, 24, 24, 3	0	tf_op_layer_Stg1_Blk2_add[0][0]
Stg1_Blk3_Res1_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Stg1_Blk2_relu[0][0]
Stg1_Blk3_Res1_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk3_Res1_conv3D[0][0]
Stg1_Blk3_Res1_relu (Activation	(None, 24, 24, 24, 3	0	Stg1_Blk3_Res1_bn[0][0]
Stg1_Blk3_Res2_conv3D (Conv3D)	(None, 24, 24, 24, 3	27680	Stg1_Blk3_Res1_relu[0][0]
Stg1_Blk3_Res2_bn (BatchNormali	(None, 24, 24, 24, 3	128	Stg1_Blk3_Res2_conv3D[0][0]
tf_op_layer_Stg1_Blk3_add (Tens	[(None, 24, 24, 24,	0	Stg1_Blk2_relu[0][0] Stg1_Blk3_Res2_bn[0][0]
Stg1_Blk3_relu (Activation)	(None, 24, 24, 24, 3	0	tf_op_layer_Stg1_Blk3_add[0][0]
Stg2_Blk1_Res1_conv3D (Conv3D)	(None, 12, 12, 12, 6	55360	Stg1_Blk3_relu[0][0]
Stg2_Blk1_Res1_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk1_Res1_conv3D[0][0]
Stg2_Blk1_Res1_relu (Activation	(None, 12, 12, 12, 6	0	Stg2_Blk1_Res1_bn[0][0]
Stg2_Blk1_Res2_conv3D (Conv3D)	(None, 12, 12, 12, 6	110656	Stg2_Blk1_Res1_relu[0][0]
Stg2_Blk1_lin_conv3D (Conv3D)	(None, 12, 12, 12, 6	2112	Stg1_Blk3_relu[0][0]
Stg2_Blk1_Res2_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk1_Res2_conv3D[0][0]
tf_op_layer_Stg2_Blk1_add (Tens	[(None, 12, 12, 12,	0	Stg2_Blk1_lin_conv3D[0][0] Stg2_Blk1_Res2_bn[0][0]
Stg2_Blk1_relu (Activation)	(None, 12, 12, 12, 6	0	tf_op_layer_Stg2_Blk1_add[0][0]
Stg2_Blk2_Res1_conv3D (Conv3D)	(None, 12, 12, 12, 6	110656	Stg2_Blk1_relu[0][0]
Stg2_Blk2_Res1_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk2_Res1_conv3D[0][0]
Stg2_Blk2_Res1_relu (Activation	(None, 12, 12, 12, 6	0	Stg2_Blk2_Res1_bn[0][0]
Stg2_Blk2_Res2_conv3D (Conv3D)	(None, 12, 12, 12, 6	110656	Stg2_Blk2_Res1_relu[0][0]
Stg2_Blk2_Res2_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk2_Res2_conv3D[0][0]
tf_op_layer_Stg2_Blk2_add (Tens	[(None, 12, 12, 12,	0	Stg2_Blk1_relu[0][0] Stg2_Blk2_Res2_bn[0][0]
Stg2_Blk2_relu (Activation)	(None, 12, 12, 12, 6	0	tf_op_layer_Stg2_Blk2_add[0][0]

Stg2_Blk3_Res1_conv3D (Conv3D)	(None, 12, 12, 12, 6	110656	Stg2_Blk2_relu[0][0]
Stg2_Blk3_Res1_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk3_Res1_conv3D[0][0]
Stg2_Blk3_Res1_relu (Activation	(None, 12, 12, 12, 6	0	Stg2_Blk3_Res1_bn[0][0]
Stg2_Blk3_Res2_conv3D (Conv3D)	(None, 12, 12, 12, 6	110656	Stg2_Blk3_Res1_relu[0][0]
Stg2_Blk3_Res2_bn (BatchNormali	(None, 12, 12, 12, 6	256	Stg2_Blk3_Res2_conv3D[0][0]
tf_op_layer_Stg2_Blk3_add (Tens	[(None, 12, 12, 12,	0	Stg2_Blk2_relu[0][0] Stg2_Blk3_Res2_bn[0][0]
Stg2_Blk3_relu (Activation)	(None, 12, 12, 12, 6	0	tf_op_layer_Stg2_Blk3_add[0][0]
Stg4_Blk1_Res1_conv3D (Conv3D)	(None, 6, 6, 6, 128)	221312	Stg2_Blk3_relu[0][0]
Stg4_Blk1_Res1_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk1_Res1_conv3D[0][0]
Stg4_Blk1_Res1_relu (Activation	(None, 6, 6, 6, 128)	0	Stg4_Blk1_Res1_bn[0][0]
Stg4_Blk1_Res2_conv3D (Conv3D)	(None, 6, 6, 6, 128)	442496	Stg4_Blk1_Res1_relu[0][0]
Stg4_Blk1_lin_conv3D (Conv3D)	(None, 6, 6, 6, 128)	8320	Stg2_Blk3_relu[0][0]
Stg4_Blk1_Res2_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk1_Res2_conv3D[0][0]
tf_op_layer_Stg4_Blk1_add (Tens	[(None, 6, 6, 6, 128	0	Stg4_Blk1_lin_conv3D[0][0] Stg4_Blk1_Res2_bn[0][0]
Stg4_Blk1_relu (Activation)	(None, 6, 6, 6, 128)	0	tf_op_layer_Stg4_Blk1_add[0][0]
Stg4_Blk2_Res1_conv3D (Conv3D)	(None, 6, 6, 6, 128)	442496	Stg4_Blk1_relu[0][0]
Stg4_Blk2_Res1_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk2_Res1_conv3D[0][0]
Stg4_Blk2_Res1_relu (Activation	(None, 6, 6, 6, 128)	0	Stg4_Blk2_Res1_bn[0][0]
Stg4_Blk2_Res2_conv3D (Conv3D)	(None, 6, 6, 6, 128)	442496	Stg4_Blk2_Res1_relu[0][0]
Stg4_Blk2_Res2_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk2_Res2_conv3D[0][0]
tf_op_layer_Stg4_Blk2_add (Tens	[(None, 6, 6, 6, 128	0	Stg4_Blk1_relu[0][0] Stg4_Blk2_Res2_bn[0][0]
Stg4_Blk2_relu (Activation)	(None, 6, 6, 6, 128)	0	tf_op_layer_Stg4_Blk2_add[0][0]
Stg4_Blk3_Res1_conv3D (Conv3D)	(None, 6, 6, 6, 128)	442496	Stg4_Blk2_relu[0][0]
Stg4_Blk3_Res1_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk3_Res1_conv3D[0][0]
Stg4_Blk3_Res1_relu (Activation	(None, 6, 6, 6, 128)	0	Stg4_Blk3_Res1_bn[0][0]
Stg4_Blk3_Res2_conv3D (Conv3D)	(None, 6, 6, 6, 128)	442496	Stg4_Blk3_Res1_relu[0][0]
Stg4_Blk3_Res2_bn (BatchNormali	(None, 6, 6, 6, 128)	512	Stg4_Blk3_Res2_conv3D[0][0]
tf_op_layer_Stg4_Blk3_add (Tens	[(None, 6, 6, 6, 128	0	Stg4_Blk2_relu[0][0] Stg4_Blk3_Res2_bn[0][0]
Stg4_Blk3_relu (Activation)	(None, 6, 6, 6, 128)	0	tf_op_layer_Stg4_Blk3_add[0][0]
max_pooling3d (MaxPooling3D)	(None, 3, 3, 3, 128)	0	Stg4_Blk3_relu[0][0]
flatten (Flatten)	(None, 3456)	0	max_pooling3d[0][0]
tab_input (InputLayer)	[(None, 9)]	0	
dense (Dense)	(None, 1024)	3539968	flatten[0][0]
dense_2 (Dense)	(None, 64)	640	tab_input[0][0]
dropout (Dropout)	(None, 1024)	0	dense[0][0]
dense_3 (Dense)	(None, 256)	16640	dense_2[0][0]
dense_1 (Dense)	(None, 256)	262400	dropout[0][0]
dropout_1 (Dropout)	(None, 256)	0	dense_3[0][0]
concatenate (Concatenate)	(None, 512)	0	dense_1[0][0] dropout_1[0][0]
dense_4 (Dense)	(None, 512)	262656	concatenate[0][0]
dropout_2 (Dropout)	(None, 512)	0	dense_4[0][0]
dense_5 (Dense)	(None, 1024)	525312	dropout_2[0][0]
dropout_3 (Dropout)	(None, 1024)	0	dense_5[0][0]

dense_6 (Dense)	(None, 2048)	2099200	dropout_3[0][0]
dense_7 (Dense)	(None, 512)	1049088	dense_6[0][0]
FVC (Dense)	(None, 146)	74898	dense_7[0][0]

Total params: 11,058,930
Trainable params: 11,056,178
Non-trainable params: 2,752

Appendix II: Raw Data

The full dataset can be downloaded at the following link:

https://www.kaggle.com/c/osic-pulmonary-fibrosis-progression/data

https://github.com/VeronicaLoy/pulmonary-fibrosis-progression

Appendix III: Download our Intelligent System

https://github.com/VeronicaLoy/pulmonary-fibrosis-progression

Citations

British Lung Foundation. (2019, 08 -). *How is pulmonary fibrosis diagnosed?* British Lung Foundation. Retrieved 10 16, 2020, from https://www.blf.org.uk/support-for-you/pulmonary-fibrosis/diagnosis

Healthline. (2016, 12 13). *IPF: Statistics, Facts, and You*. Healthline. Retrieved 10 16, 2020, from https://www.healthline.com/health/managing-idiopathic-pulmonary-fibrosis/ipf-facts#prevalence

Kaul MD, FASGE, V., Enslin PA-C, S., & Gross MD, FASGE, S. A. (2020, October). History of artificial intelligence in medicine. *Gastrointestinal Endoscopy*, *92*(4), 807-812. https://www.sciencedirect.com/science/article/pii/S0016510720344667

NIH: National Heart, Lung, and Blood Institute. (-, - -). *Pulmonary Fibrosis*. MedlinePlus. Retrieved 10 16, 2020, from https://medlineplus.gov/pulmonaryfibrosis.html

Silvia, PhD, P. (2020, 10 06). *Pulmonary Fibrosis Prognosis*. Pulmonary Fibrosis News. Retrieved 10 16, 2020, from https://pulmonaryfibrosisnews.com/pulmonary-fibrosis-prognosis/#:~:text=All%20forms%20of%20pulmonary%20fibrosis,treatment%20can%20significantly%20improve%20survival.