COVID-19 risk stratification on Chest X-Rays: performance on a small cohort of patients in Switzerland

Shuangqi Li, Chenyang Cao, Tiannan Sha
{shuangqi.li, chenyang.cao, tiannan.sha}@epfl.ch
School of Computer and Communication Sciences, EPFL, Swizterland

Abstract

Clinical outcomes of COVID-19 range from asymptomatic carriage to death. Early prognostication i.e. predicting the likely course of COVID-19 is of great importance for allocating limited medical resources and preventing avoidable death. In this project, with very limited data from 72 COVID-19 patients at Lausanne University Hospital, we aim to predict how severe the symptoms of a COVID-19 patient will become from X-ray images and clinical findings (demographics, symptoms, signs and paraclinical exams). To overcome the challenge of having very limited amount of data, we leveraged three approaches: (i) transfer learning with images as input (ii) traditional statistical learning methods with tabular data as input and (iii) including tabular clinical finding data during the training of CNN.

1 Introduction

The ongoing COVID-19 pandemic remains a global health threat even after a year of circulation. COVID-19 patients can experience significantly different outcomes, ranging from asymptomatic carriage to death and recently it has also been linked to chronic disease (AKA "long COVID")(1). As a result, prognosis, i.e. predicting the likely course of COVID-19 is of great importance for better allocating limited medical resources and preventing avoidable death.

In this project, we essentially addressed a binary classification problem with small and dataset. With 72 chest X-ray (CXR) images and associated clinical findings from 72 COVID-19 patients at Lausanne University Hospital, we aim to perform binary classification to predict whether a patient's clinical information at triage can discriminate their later outcome between 1) mild disease (self-resolving disease not requiring hospitalisation) or 2) severe disease (require hospitalization, ventilation or resulting in death). The main challenge of this project is the very limited amount of data available. To overcome this challenge, we took three general approaches: (i) training CNNs with pretrained weights on our dataset (ii) using traditional statistical learning methods and (iii) combining the image features and tabular features to train the CNN.

2 Dataset

2.1 Cohort

All of our data are from 72 COVID-19 patients at Lausanne University Hospital. There are three components of our data. First CXR images in DICOM format. Each patient has from one to three CXR. Each patient has at least one AP/PA X-ray image (i.e. taken from the front/back of the patient as opposed to from the side). Some patients also have additional AP/PA X-ray images and/or lateral X-ray images (taken from the side).

The second component is clinical data in CSV format. There are 158 variables which are divided into 2 parts: (i) Ground Truth labels: the severity categorisation of the patient after a 30 day follow-up period (i.e. mild disease treated as an outpatient, or severe disease requiring hospitalisation with oxygen therapy or futher interventions) . (ii) CXR variables: expert analyses of the CXR images. Expert labels are divided into 4 zones: left lower(ll), left upper(lu), right lower(rl), right upper(ru) where different pathological labels are attributed such as *predominace of the abnormalities*, *presence of consolidation* and *presence of ground glass opacity*, etc. What's more, there is a feature called *severity score* in each zone, which means the severity of the disease in the zone. And the global severity is just the sum of the *severity score* in all zones. Besides, there are some other global clinic findings, such as *pleural effusion* and *mediastin*.

2.2 Preprocessing

2.2.1 Tabular data

Feature engineering is essential to augment the predictive capacity of this small dataset. Because most of the features are given in levels, we can't directly use the level labels in our model as numeric values. So we choose to use 0/1 variables to represent the levels, which is called dummy variables. After that, we find that there are several levels contain no sample, so we choose to combine some levels in order to reduce the number of useless features.

After the data preprocess, we can use the tabular data in model (ii) and (iii).

2.2.2 X-ray images

In this project, we utilized the model in (2) as a pretrained model, which was trained on four of the largest public datasets using over 200k unique chest X-rays after filtering for one AP or PA view per patient. To ensure our training dataset has similar distribution as the dataset used for pretraining, we also only used one AP or PA view per patient and ignored lateral views. Moreover, only few patients have lateral views. If we include these lateral views for these few patients, we risk overfitting for these few patients.

We also scaled our images pixel values to the same scale as the pretraining dataset to ensure we have similar distribution.

3 Methods

3.1 Transfer Learning

The intuition behind utilizing pretrained convolutional neural networks (CNN) is that by pretraining the CNN on a much larger dataset than ours, the CNN can better learn features that are of vital importance for our image classification task. We can then feed our dataset to the pretrained CNN to adjust it to fit our particular task.

3.1.1 DenseNet Architecture

We choose DenseNet as our architecture because DenseNets (3) have been shown to be a very effective architecture for CXR predictive models (Rajpurkar et al., 2017). It has been shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and output. Densenet architecture embraces this observation and connects each layer to every other layer in a feed-forward fashion as shown in Figure 1.

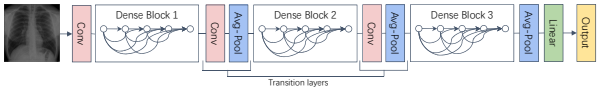

Figure 1: An example DenseNet with three blocks

3.1.2 Pretrained models

We chose DenseNet-121 as our architecture. We tried three variations in terms of pretraining: no pretraining, pretrained on CXR images and pretrained on ImageNet dataset.

The DenseNet-121 model pretrained on the ImageNet dataset is available in the **torchvision** Python package. The DenseNet-121 model pretrained on CXRimages is available at torchxrayvision repo. It was trained on four of the largest public chest X-ray datasets utilizing over 200k unique chest X-rays after filtering for one AP or PA view per patient.(2).

We modified the last layer of the pretrained models by changing the number of output classes to 2. For all pretrained models in our experiments, we freeze the weights (and bias) of all layers except the last layer.

3.1.3 Training

Training DenseNet with no pretraining SGD is used with *learning rate* $= 0.1$, *momentum* $= 0.9$, Nesterov momentum enabled, learning rate decay parameter $\gamma = 0.1$. Number of epochs is 1100.

Training pretrained DenseNets Adam is used with *learning rate* $= 0.1$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Number of epochs is 700.

Data Augmentation To improve generalization, we also implemented random rotation, random crop and random horizontal flip.

3.2 Statistical Methods

3.2.1 Model design

When using all the features to train a baseline model, we can get a model with okay performance. But if we do so (i) there will be so many features in the model that it will be difficult to use and interpret. (ii) some expert knowledge of the features can't be added into the model. That means even though the baseline model have okay performance, it's poor at usability, scalability and interpretability.

In order to select the features and train the models more wisely, based on the dataset, we proposed a hierarchical model, which uses the local information and global information in different ways. Our statistical model architecture is shown in Figure 2

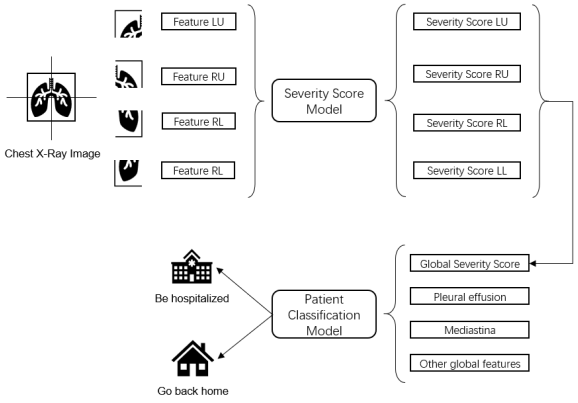

Figure 2: Tabular model

In our dataset, there are both local features and global features. In different zones, the kind of (local) features are the same. A basic idea is to use a model to get a summary of the local features. In our case, the local *severity score* can be considered as the summary of a zone. And the sum of all the local *severity score* is the feature *global severity score*. Together with the other global features, we can finally do the classification.

By using this hierarchical model, we can get: (i) Interpretability. Because the local features are summarized by the Severity Score Model, there will be fewer features in the Patient Classification Model, and the doctor can easily know how the model makes a decision. Also, by looking into the Severity Score Model, the influence of the local features can also be interpreted very well. (ii) Scalability. We select the features by our dataset, but the user can add or drop some local features, which will not impact the Patient Classification Model and also not changing the meaning of the *global severity score*. What's more, one can also change the type of global features without changing the Severity Score Model.

3.2.2 Model optimization

Binary Severity Score In the dataset, the feature *severity score* has 5 levels. That means the Severity Score Model must solve a multi-class classification problem, which is much more difficult than the binary classification problem. The distribution of the feature are imbalanced in each level. So we tried to combine some levels and transform it into a binary classification problem. Because the feature *global severity score* is the sum of the local *severity score*, we need to train a new Patient Classification Model based on the binary local *severity score*.

Combination of the zones Instead of train four different models for the zones, we use a common model to train more than one zone. We made two attempts. The first one regards position as two binary features: *left/right* and *lower/upper*. And adding these position features into the model. The second one trains two models, one for the lower zones, the other one for the upper zones.

Training with weight There is a global feature *cxr quality*, which means the quality of the patient's CXR image. Our idea is training the model with weighted data, and the image with high quality has higher weight. There are two features related to the image quality, the *cxr type* and *cxr quality*. We find that no outpatient has an AP CXR image, so it may lead to some bias because of the imbalance. However, the feature *cxr quality* is balanced, so we can define the weight as:

$$\mathrm{w}(\alpha) = \begin{cases} 1-\alpha & cxr\_quality = sufficient \\ 1 & cxr\_quality = exemplary \end{cases}$$

And then we can find the best model by tuning the parameter α.

3.3 Mixed-features model

To combine the best of both worlds, we integrate the tabular features in our DenseNet model. The last layer of the DenseNet architecture is a fully-connected layer which has 1024 as input size and the number of predicting classes as output size. We can concatenate the input vectors with the standardized, scaled tabular features and feed into the fully-connected layer which then has 1024 + number of dimensions of tabular features as input size. This idea is demonstrated in Figure 3. As shown in Table 1, integrating the tabular features improves the performance.

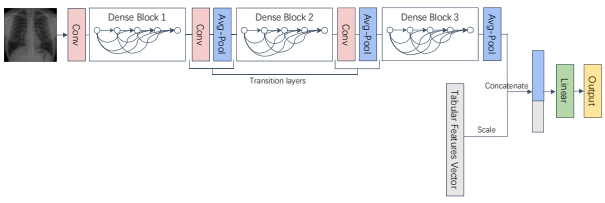

Figure 3: Mixed model

4 Results

Our results of the models are shown in Table 1.

Results are obtained through stratified k-fold cross validation on the entire dataset. Deep learning works best on large datasets which can be split in to train, test and validation sets. A large cross validation set for tuning hyper-parameters will be more comparable to the population from which is was drawn and hence can help us estimate models' performance without the effect of sampling bias.

However, this convention does not apply to our project. Our dataset is very small, if we also have a testset, one consequence is that we will have even fewer data to train our model; the other is that our testset will be so tiny that the distribution of this tiny testset is likely to be different from the population's distribution and the evaluation done on testset will be highly inaccurate. For example, suppose we have one obviously unhealthy lung's CXR image as testset, and we correctly predicted its value, then (a) the 100% accuracy is unreliable and (b) the population distribution of all CXR images will be very different from this testset.

For our models, given the computational resource available, we chose stratified 6-fold cross validation and averaged the performance of each fold as our final results. For different models, we kept the same random seed to generate the same stratified k-fold splits, in order to mitigate the unpredictability of results. We chose stratified cross validation to ensure the class distribution in each fold is similar to our entire dataset.

4.1 Discussion

For statistical models, in most cases, Logistic Regression models have the best performance. Training two models for the upper and lower zones are better than one model and four models. The binary optimization can improve the Acc and F1 of the SS model, but lead to information loss of the PC model. The weighted model can slightly improve the AUC. However because we choose the weights based on the optimal AUC, the Acc and F1 score may become lower.

For pretrained CNNs, we observe that pretraining is essential for obtaining good performance. The models pretrained on CXR images do not prevail over the models pretrained on Imagenet.

Including tabular data vectors during the training of CNN seems to provide the overall best performance considering all of AUC, accuracy and F1

Model	Tabular	AUC	Acc	F1
DN	N	0.613	0.672	0.785
DN ImgNet	N	0.804	0.778	0.844
DN CXR	N	0.792	0.726	0.780
DN ImgNet	Y	0.811	0.764	0.824
DN CXR	Y	**0.812**	**0.794**	**0.845**
Model	Type	AUC	Acc	F1
Baseline	LR	**0.802**	0.750	0.836
	RF	0.631	**0.792**	**0.879**
	SVM	0.743	0.694	0.789
SS	LR	0.817	0.436	0.408
	RF	0.786	0.443	0.408
	SVM	0.803	0.400	0.360
SS+b	LR	**0.887**	0.772	**0.751**
	RF	0.842	0.723	0.673
	SVM	0.853	0.737	0.676
SS+b@U	LR	0.842	**0.794**	0.713
SS+b@LU	LR	0.760	0.729	0.596
PC	LR	0.816	0.694	0.779
	RF	0.752	0.681	0.785
	SVM	0.802	0.694	0.779
PC+w	LR	0.825	0.708	0.787
	RF	0.766	0.681	0.785
	SVM	**0.827**	0.681	0.765
PC+b	LR	0.779	**0.722**	**0.800**
	RF	0.758	0.681	0.770
	SVM	0.783	0.653	0.738
PC+b+w	LR	0.787	0.708	0.787
	RF	0.791	0.681	0.770
	SVM	0.783	0.653	0.738

Table 1: **Results of different models**. DN: Densenet-121. ImgNet/CXR: pretrained on imagenet/CXR images. Tabular: include tabular data when training CNN. SS: severity score. PC:patient classification . LR: logistic regression. RF: random forest. SVM: support vector machine. Suffixes: the optimization methods. "+b": binary local *severity score*; "@": train a model in the specific zones; "+w": train the model with weighted data.

scores.

5 Conclusion

We explored the three approaches to solve the problem of binary classification on a small and imbalanced dataset. While both pretrained CNN and statistical models provide satisfactory results, the best results are obtained through including the clinical finding data during the training of pretrained CNN.

Acknowledgements

The authors thank Annie, Mariko and CS433 TAs for their constructive advice.

References

[1] T. Struyf, J. J. Deeks, J. Dinnes, Y. Takwoingi, C. Davenport, M. M. Leeflang, R. Spijker, L. Hooft, D. Emperador, S. Dittrich *et al.*, "Signs and symptoms to determine if a patient presenting in primary care or hospital outpatient settings has covid-19 disease," *Cochrane Database of Systematic Reviews*, no. 7, 2020.

[2] R. B. Joseph Paul Cohen, Mohammad Hashir and H. Bertrand, "On the limits of cross-domain generalization in automated x-ray prediction," *Proceedings of Machine Learning Research*, vol. arXiv:2002.02497, 2020. [Online]. Available: https://arxiv.org/abs/2002.02497

[3] Z. L. Gao Huang and L. van der Maaten, "Densely connected convolutional networks," *Computer Vision and Pattern Recognition*, vol. arXiv:1608.06993, 2018, version 5. [Online]. Available: https://arxiv.org/abs/1608.06993