Performance of Shallow Neural Network: A Challenge on Covid Image Classification

Francesco Tomaselli[†]

***Abstract*—Image classification is a crucial task in medical and epidemiological research, as it can prevent pandemic diffusion by detecting patients affected by particular illnesses. The COVID-19 pandemic crisis is a recent and prominent example. This report aims to quantify the performance of several neural network architectures in predicting patients' classes from chest X-rays (CXRs). The proposed models include: 1) a shallow CNN, an improvement of LeNet; 2) a CNN combined with an RNN via an LSTM cell; 3) a Visual Geometry Group network with 6 convulutional blocks and its regularized version; 4) the COVID-Net and a regularized version of it. We compare and discuss these models in terms of performance, memory occupation, and time requirements. Instead of identifying the best overall model, we consider the pandemic as a life-cycle process (growth, saturation, and decline) and suggest using different tools depending on the pandemic phase. To simulate the initial phase of a pandemic, where data is limited, we use a dataset of 4576 CXRs. Our results show that a shallow and simple neural network achieves the best performance on our data.**

***Index Terms*—COVID Dataset, Chest X-Rays, Convolutional Neural Network, Recurrent Neural Network.**

I. Introduction

The COVID-19 pandemic has underscored the inadequacy of current international medical research tools to effectively respond to serious pandemic events. Therefore, it remains imperative to continue developing new ancillary tools for medical research in this context. The aim of this paper is to contribute to the advancement of novel statistical models designed to classify patients' diseases. This contribution will be achieved through a comprehensive comparison with established models, focusing on their predictive capabilities and resource utilization.

The paper aims to compare various convolutional neural networks (CNNs) based on their predictive capacity, time consumption, and parameter requirements. Previous studies have mainly focused on achieving the optimal model fit, driven by the urgency to address the global pandemic caused by SARS-CoV-2. However, it is now imperative to further analyze the behaviors of deep learning models to better prepare for future health crises.

In this study, we tackle a supervised learning problem using a dataset comprising 4576 chest X-ray images from patients categorized into three classes: normal, pneumonia, and COVID (1525 images each). The primary objective of this paper is to compare various neural network architectures based on their predictive performance, computational time requirements, and parameter storage capacity. A noteworthy aspect of our findings is the notable performance of shallow models in comparison to deeper architectures that are commonly employed in contemporary literature.

This paper is structured as follows. In Section II we describe the state of the art, the system and data models are respectively presented in Section III and IV. The proposed CXRs processing technique is detailed in Section IV and its performance evaluation is carried out in Section V. Discussion and concluding remarks are provided in Sections VI and VII, respectively.

II. Related Work

Since 2020, numerous scientific papers have contributed to the development of tools serving as "second eyes" for image recognition of SARS-CoV-2 from chest X-rays (CXRs) and computed tomography (CT) scans:

1) Hemdan et al. [1] demonstrated how transfer learning [2] from deep learning models can be used to perform COVID-19 detection using images from three most commonly used medical imaging modes X-Ray, Ultrasound, and CT scan. They compared different deep networks and selected the VGG19 [3] for the CXRs classification, because it reached the best performance, i.e. F1 score of 0.87. Moreover, notice that they exploited only 140 COVID-19, 322 Pneumonia and 60000 Normal CXRs, while the dimension chosen was $(1024 \times 1024 \times 3)$. Concluding, compared with our research, this study focus on transfer learning, starting from huge dataset of Normal patients, it has few observations and a much more precise image storing.
2) Khan et al. [4] introduced a specific neural network architecture, the Deep COVID DeteCT (DCD), a convolutional neural network (CNN) that utilizes the entire chest CT volume to automatically predict COVID-19 (COVID+), pneumonia, and normal controls. The analysis are performed on 12 different research institutes and performances for each of them are reported. Moreover, this study has 1000 CXRs across all the institutes and each image is standardized in a $(254 \times 254 \times 24)$ shape. Anyway, the accuracy for each site varies from China-1, for which classification accuracy is 0.707, and GUMS, for which the classification accuracy is 0.944.
3) Whang et al. [5] presented COVID-Net, a deep neural network specialized for COVID detection from CXRs depicting COVID, pneumonia, and normal cases. For this study, 13975 CXRs with resolution are exploited,

[†]Department of Mathematics, University of Padova, email: francesco.tomaselli@studenti.unipd.it

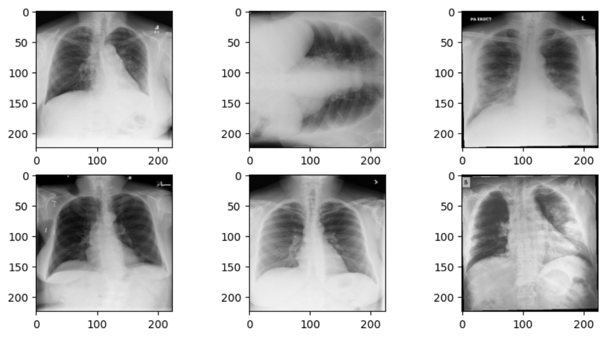

Fig. 1: First 6 CXRs of standardized COVID dataset.

each image with a resolution of $(480 \times 480 \times 3)$. The accuracy of COVID-Net model is the best one, compared with VGG19 and ResNet-50, equal to 0.933.

4) Buonsenso et al. [6] conducted a study focused on the use of deep learning for the classification and localization of signs of COVID-19 in lung ultrasound, which may also be relevant to X-ray imaging. The best classification model is a particular form of CNN, developed by the authors, for which the prediction accuracy is 0.89.

While many other studies on COVID datasets have been produced, we focus on a reduced dataset in this paper, considering that the COVID dataset comprises thousands of images. Our emphasis lies in modifications of shallow architectures, which we will subsequently compare with deeper ones. Moreover, this study keeps central the importance to obtain performance of the same order, trying to focus on memory and time consuming.

III. Processing Pipeline

The goal of this report is to propose neural network architectures capable of predicting patients' conditions from chest X-rays (CXRs) when the true labels are unknown, i.e., to evaluate model performance on undiagnosed patients. Previous studies have demonstrated that image standardization is essential for accelerating the training phase and enhancing model performance [2], [7]–[9]. For our dataset, we standardized the images, which originally varied in size, to a square shape of $(224 \times 224 \times 3)$ pixels and normalized the pixel values to a range between 0 and 1. Notice that other studies, for example in L. Wang et al. [5], chose more precise image approximation, i.e. $(448 \times 448 \times 3)$. This choice has an impact on results, since the more rich is the dataset, the more the net can learn from it. However, the available resources to write this paper didn't allowed to store more than 4 thousand images with that high resolution. In Figure 1 is shows an example of the standardized dataset we are working with. Notice from the dataset example that the rotation of CXRs can help the algorithm to be invariant to translation. In other words, we can focus better in detecting the presence of an object instead to know where the object is.

Additionally, Table 1 summarize the dataset splitting we used for the analyses.

	Training	Validation	Test	Tot
Percentage	75%	12.5%	12.5%	100%
Absolute val.	3432	572	572	4576

TABLE 1: Dataset splitting.

The dataset splitting phase is in general not trivial, since it has to satisfy several principles: 1) training set has to be large enough to enable the network to learn from data; 2) validation set has to be large enough to ensure a good hyperparameter analyses from validation loss and accuracy plots; 3) the test set has to be large enough in order to provide a non-biased and consistent estimate of the generalization error. We need to keep in mind the random sample for each set is fundamental, in order to avoid selection bias in our analyses and correlation in data. Moreover, the patients' class are perfectly balanced among *Normal*, *Pneumonia* and *COVID*, so no stratified sampling or generation is required. In the following paragraphs, we provide a brief introduction to the models we adopted. Further details regarding the Convolutional Neural Network (CNN), Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM), VGG9 and COVID-Net models are presented in Section IV. The general tools used for all models include the Adam and Early Stopping algorithms [2]. The first one, is called *Adaptive Moments*, and it has two particular aspects: 1) Momentum is incorporated directly as an estimate of the first order moment; 2) Adam includes bias correction to the estimates of both 1-st and 2-nd order. Adam is generally regarded as being fairly robust to the choice of hyperparameters, while more attention is needed on learning rate. In Algorithm 1 is shown the Adam Algorithm. The choice of the optimization algorithm is not crucial, since there is not empirical evidence on what algorithm is better than others among all applications. In practice, we tried other optimization algorithm with adaptive learning rate, such as RMSProp algorithm.

Algorithm 1 Adam Algorithm

```
**Require:** Step size $\epsilon$, exponential decay rates $\rho_1$ and $\rho_2$,
  constant $\delta$ for numerical stability.
  Initialize parameters $\theta$
  Initialize 1-st and 2-nd moments $s = 0$ and $r = 0$
  $t = 0$
  **while** Not Convergence **do**
      Sample a minibatch from $(X, y)$
      Compute gradient $g = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L\left[f(x_i, \theta), y_i\right]$
      $t = t + 1$
      $s \leftarrow \rho_1 s + (1 - \rho_1) g$
      $r \leftarrow \rho_2 r + (1 - \rho_2) g^T g$
      $\hat{s} \leftarrow \frac{s}{1 - \rho_1^t}$
      $\hat{r} \leftarrow \frac{r}{1 - \rho_2^t}$
      $\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$
      $\theta \leftarrow \theta + \Delta\theta$
  **end while**
  **Return** $\theta$
```

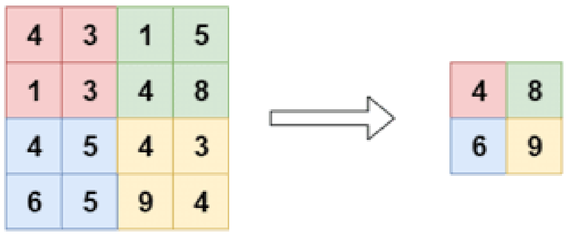

Fig. 2: 2D Max Pooling (2×2) and downsampling (2×2).

The latter, Early Stopping, is an algorithm that terminates the training phase and retains the best model parameters when the validation accuracy does not improve after 15 epochs. This technique also acts as a form of regularization for validation error with a U-shaped curve. We now describe the general convolution block used in each architecture, for which we refer to [2], [10]–[14]. Firstly, we define a 4-D tensor K with element $K_{i,j,k,l}$ representing the connection strength between a unit in channel i of the output and a unit in channel j of the input, with an offset of k rows and l columns between the output and input units. Assume our input consists of data V with element $V_{i,j,k}$ representing the value of the input unit within channel i at row j and column k, and output Z with the same offset as V. The output Z is obtained by convolving K across V without flipping K:

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n} \tag{1}$$

The **convolution** operator allows us to introduce equivariance to translation and sparse interactions. If we want to sample only every s pixels in each direction in the output, in order to reduce the computational cost, then we can define a downsampled convolution function such that

$$Z_{i,j,k} = \sum_{l,m,n} \left[V_{l,(j-1)\times s+m,m,(k-1)\times s+n} K_{i,l,m,n} \right] \tag{2}$$

where s is the **stride** of the downsampled convolution. Another important feature we exploited for the construction of the neural networks is the special case of **zero-padding**, the **same** convolution, for which we introduce the number of zero-padding that are necessary to keep the size of the output equal to the size of the input after the convolution stage. Another component is the **pooling** stage, which introduces invariance to small translations and is important when we are more concerned with the presence of a feature than its exact position, as in our case. In particular, we exploited the 2D Max Pooling operator, for which, every p pixels, we take as the respective output the maximum value among these p^2 values. In Figure 2 a convolution operator with Max Pooling and downsampling is shown.

Keep in mind that Max Pooling and downsampling are powerful tools to approximate underlying functions, but they can lead to underfitting if the assumption of the function is not accurate. Batch Normalization is introduced as a method of adaptive reparameterization that can be applied to any input or hidden layer, significantly reducing the problem of coordinating updates across many layers. Let H be a minibatch of activation of the layer to normalize, then the normalized layer is

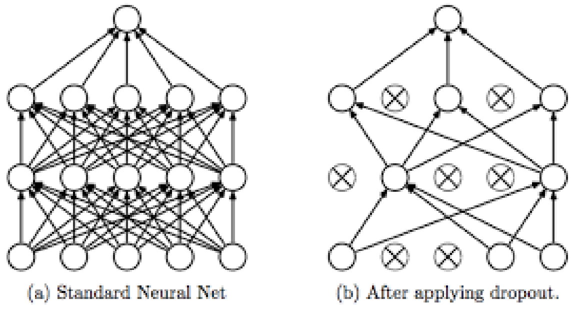

Fig. 3: Dropout example.

$$H' = \frac{H - \mu}{\sigma}, \tag{3}$$

where μ is the minibatch mean and σ is the standard deviation. Finally, we contemplated some regularization tools, that can handle the bias-variance trade-off

$$E[(y - \hat{y})^2] = \text{bias}^2 + \text{variance}. \tag{4}$$

In particular, we exploited $L2$-norm and Dropout. The first one introduces a penalty norm to the objective function $J(\theta, X, y)$:

$$\tilde{J}(\theta, X, y) = J(\theta, X, y) + \lambda ||\theta||_2^2, \tag{5}$$

where λ is the hyperparameter tuning the parameter penalization: the higher the hyperparameter, the more the regularization. If we choose too low hyperparameter values, we can encounter overfitting problems, or underfitting problems for much larger hyperparameter values. Finally, Dropout is a method that trains an ensemble consisting of all sub-networks that can be formed by removing non-output units from a base network. In practice, we define a mask vector μ, representing the probability of switching off a unit by setting its output to zero. In Figure 3 an example of dropout is shown.

Dropout is an ensemble method, for which, at each iteration, we randomly sample a sub-network from the original one, and, at the end, we ensemble the final net by averaging all outputs obtained by single networks. This method, as all other ensemble methods, is a way to reduce variance and for this reason dropout results in a regularization method.

IV. Learning Framework

In the next paragraphs, we introduce the models fitted to predict the patients' classes. In the *Appendix*, specific plots of accuracy and loss for each model, computed for each epoch on the training and validation phases, are reported.

A. Logistic Lasso

Logistic Lasso is a Logistic Model that introduces a norm-1 penalty on model parameters. The norm-1 penalty shrinks the irrelevant model parameters to 0, selecting only important features in terms of predictions. This norm introduces sparsity in parameters. Since we have not two, but three classes, we need to build a logistic regression for multiclass regression. This model is called Multiclass Logistic [15], [16]. The Multiclass Logistic with Lasso, for a problem with K classes, assumes the following form

$$P\left(Y=k, X=x\right)=\frac{e^{\beta_{ok}+\beta_k^T x}}{\sum_{k=1}^{K} e^{\beta_{ok}+\beta_k^T x}}+\lambda\sum_{k=1}^{K}||\beta_k||_1 \quad (6)$$

We aspect from this model that the memory occupation is very small, since we can suppose not all parameters are important to predict the patients' class. In practice, we reshaped the original image tensors in a 1-D vector and we reorganized all this vectors in a data matrix.

B. CNN

The first model is a slight modification of LeNet [17], where we used 3 convolution blocks, each composed of convolution, batch normalization, and max pooling, followed by three dense layers. Figure 4 shows the architecture for this CNN.

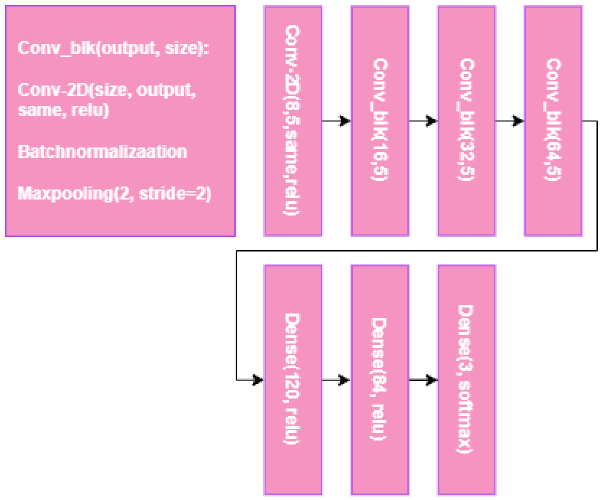

Fig. 4: CNN architecture.

During the trials, we tried to add some regularization, such as Dropout or $L2$-norm, but these models exhibited lower generalization accuracy over time (indicating probable underfitting). Conversely, adding more layers resulted in better training accuracy but lower generalization accuracy (indicating probable overfitting).

C. VGG9

The Visual Geometry Group (VGG) architecture can be partitioned into two parts: the first consisting mostly of convolutional and pooling layers, and the second consisting of fully connected layers that are identical to those in AlexNet [18]. In this study, we implemented a personalized VGG9, whereas the original VGG had 16 layers. The choice of depth was guided by our restricted computational resources and the limited amount of images. The specific VGG9 net comprises 4 convolutional blocks: the first one with 1 convolution and 16 filters, the second one with 1 convolution and 32 filters, the third one with 2 convolutions and 64 filters, and the last one with 2 convolutions and 128 filters. The overall model structure is shown in Figure 5.

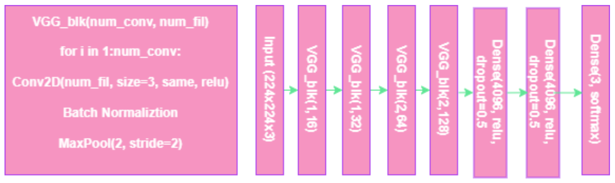

Fig. 5: VGG9 net architecture.

Moreover, we propose a slight modification of VGG9, where we add a Batch Normalization layer after each conventional Max Pooling layer, and then a Dropout layer with a 10% probability. These modification are allowed with the aim to construct robust models that capture the hard true underling function with a small available dataset.

D. CNN-LSTM

In this section, we present a new network architecture that can be divided into two blocks: 1) several convolutional blocks for feature extraction; and 2) a Long Short-Term Memory (LSTM) block to capture temporal dependencies in the data. The recurrence is managed by the LSTM cell, that defines the weight which with the previous layers are considered at the current time. The LSTM cell, is composed by three gates, called, respectively, input (i_t), forget (f_t) and output (o_t) gate, for which the equations are shown below.

$$\begin{aligned} I_t &= \sigma\left(w_i[H_{t-1}, x_t]+b_i\right) \\ F_t &= \sigma\left(w_f[H_{t-1}, x_t]+b_f\right) \\ O_t &= \sigma\left(w_o[H_{t-1}, x_t]+b_f\right) \end{aligned} \quad (7)$$

The input gate tells us what new information we are going to store in the cell state, the forget gate tells the information to throw away from the cell state, and the output gate is used to provide the activation to the final output of the LSTM block at timestamp t. Look at Figure 6 for a graphical example.

This network is a combination of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN), where the recurrence and the vanishing gradient problem are addressed by the LSTM cell. Essentially, this network builds

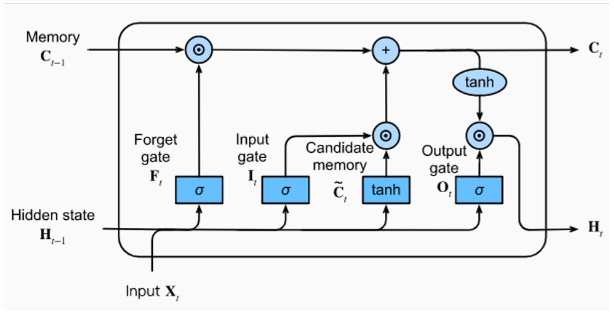

Fig. 6: LSTM cell.

on VGG's concept of blocks and adds a recurrent unit. Figure 7 shows the CNN-LSTM architecture.

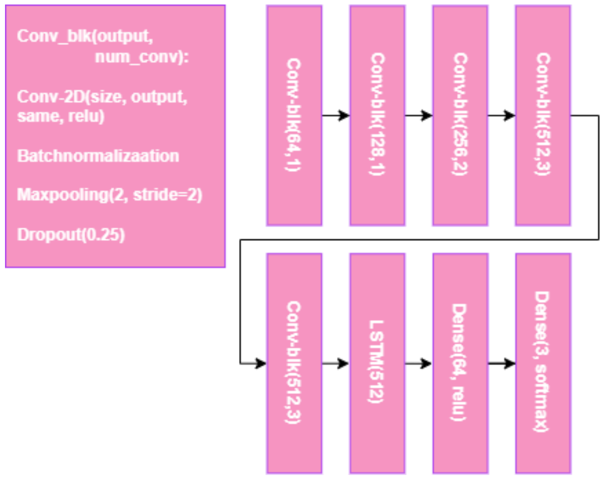

Fig. 7: CNN-LSTM architecture.

E. COVID-Net

Here, we present the COVID-Net implemented by L. Wang et al. [5] to compare its results with the models defined previously on the same dataset. Moreover, we implemented two different models: 1) the original COVID-Net; and 2) a modified version of COVID-Net with added regularization using a dropout of 10%. The latter addresses the issue that the original COVID-Net was designed for a much larger dataset, making its depth excessive for our dataset. The modified version includes a 10% dropout after each convolutional layer to mitigate overfitting, which became evident when fitting the original COVID-Net to our data. The general architecture of COVID-Net consists of block of convolutions, followed by batch normalization, dropout for the regularized version of COVID-Net and max pooling. Moreover, the output of layer l is passed as input to layer $l+1$ and $l+2$. So, the inputs of layer $l+2$ consists of a concatenation of output of layers l and $l+1$. This strategy allows the network to learn on a much more plentiful context. In Figure 8 is shown the general architecture of COVID-Net.

V. Results

In this section, we present the results of the fitting phase for our data. Table 2 shows the accuracy achieved for each model on the test set, the memory occupation in terms of millions of trainable parameters, the time required for fitting and evaluation in terms of minutes[1], and the number of epochs.

Model	Accuracy	Memory	Time	Epochs
Lasso	0.885	0.253	49.41	100
CNN	0.923	4.87	4.57	29
CNN-LSTM	0.895	16.26	33.64	30
VGG9	0.911	119.8	3.63	21
VGG9 Reg	0.930	119.8	10.62	49
COVID-Net	0.881	0.711	29.01	32
COVID-Net Reg	0.918	0.711	75.34	62

TABLE 2: Comparison of Neural Network Models.

From Table 2, we can see that there is no model that stands out in all fields of interest. However, the VGG9 with regularization shows the best performances, but with very high storage requirement. Moreover, the CNN network shows high performances with a much more reasonable memory occupation and time requirement. Finally, we can see that Multiclass Logistic with Lasso shows low accuracy and high time requirement, but in terms of memory occupation it is the best one. In the *Appendix*, Table 4 reports several performance measures for each class and model, such as Precision, Recall, and F1-Score.

In Table 3, we present the average Precision, Recall, and F1-Score for each model, computed across the three classes.

Model	Precision	Recall	F1-Score
Lasso	0.88	0.89	0.89
CNN	0.93	0.92	0.92
CNN-LSTM	0.90	0.90	0.90
VGG9	0.92	0.91	0.91
VGG9 Reg	0.93	0.93	0.93
COVID-Net	0.90	0.88	0.88
COVID-Net Reg	0.92	0.92	0.92

TABLE 3: Average Precision, Recall, and F1-Score for Neural Network Models.

In Figure 9 we show the best three models in terms of accuracy, time requirement and memory occupation. This Figure is useful to understand what model we can use depending on our needs and, at the same time, it underlines that the best model is the shallow CNN.

VI. Discussion

In Section V, we presented the results of fitting models to the data. We observed that no model outperforms the others in terms of generalization, time and memory at the same time. However, depending on our needs, we can choose the best model for the specific task. Given the restricted amount of data we are working with, which can well represent the initial phase faced by a Data Scientist at the beginning of a

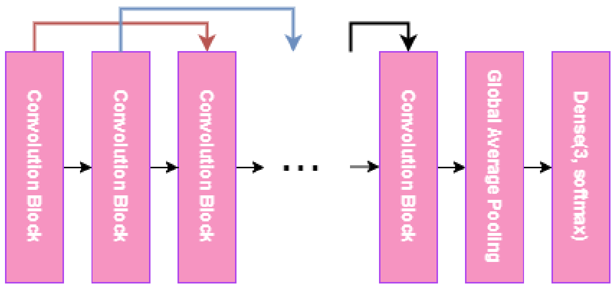

Fig. 8: Original COVID-Net general architecture.

[1]For the time requirement, we exploited a T4 GPU of Google Colab.

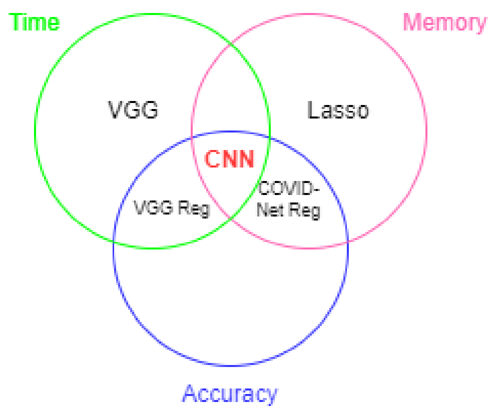

Fig. 9: Best three models in terms of accuracy, time requirement and memory occupation.

global pandemic, we arrive at the following discussion. First, we need a model with good generalization capacity: this is achieved by all models, but the simple CNN and the VGG9 with regularization have slightly better performance. From a computational point of view, we need a model that maintains good performance with reasonable memory occupation: this can be achieved by both versions of COVID-Net and CNN. Finally, the time required for training and evaluation is crucial if we consider a scenario with more data, which may lead us to choose VGG or CNN. As we can see, the best model, found at the intersection of these three criteria, for an initial phase is the CNN architecture. In addiction, we can suppose that, at the growing of image dataset, representing the pandemic advancement over time, a shallow neural network may can fail to capture the real function underling data. For this reason, may a deeper neural network such as COVID-Net without regularization or a deeper VGG can better carry out this task, nevertheless we have to keep in mind that memory occupation and time requirement are very important to ensure good model performance with reasonable resources consuming.

VII. Conclusions

The COVID-19 pandemic has confronted the scientific community, particularly data scientists and data analysts, with the challenge of providing suitable tools to detect patients affected by SARS-COV2. Specifically, we have 4576 CXRs, split into training, validation, and test sets with proportions of 70%, 15%, and 15% respectively. Therefore, we are working with a restricted dataset, which can represent the initial phase of a pandemic when the amount of available data is low. Given this context, we developed several neural networks with different strategies, aiming to predict the patients' disease from CXRs: 1) a simple Convolutional Neural Network, which is a slight improvement of LeNet; 2) a combination of a Convolutional Neural Network for feature extraction, followed by a recurrent cell, represented by a Long-Short-Term Memory cell; 3) a Visual Geometry Group network with 6 layers; and 4) the COVID-Net and a modified version that includes regularization. The results of the analyses led us to choose, in terms of performance, time requirement, and memory occupation, the simple CNN. This model achieved an accuracy of 92.3% on the test set, an average F1-Score of 0.92 across the three classes, a memory occupation of 4.87 million trainable parameters, and required 4.57 minutes for training and evaluation. From this paper, we conclude that not only is there no single model that consistently outperforms the others, but also that the most suitable neural network can vary depending on the phase of the analysis. Additionally, in a progressed phase of the pandemic, other networks, such as the original COVID-Net, may be more suitable than the simple CNN.

References

[1] E. E.-D. Hemdan, M. A. Shouman, and M. E. Karar, "Covid-19 detection through transfer learning using multimodal imaging data," *IEEE Access*, vol. 8, pp. 101302–101323, 2020.
[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[4] A. Khan, A. Sohail, S. Khan, Z. Nawaz, S. Naz, R. Siddique, S. Khalid, D. J. Gomes, and T. Mahmood, "Deep covid detect: An international experience on covid-19 lung detection and prognosis using chest ct," *arXiv preprint arXiv:2004.12537*, 2020.
[5] L. Wang, Z. Lin, and A. Wong, "Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.
[6] D. Buonsenso, D. Pata, A. Chiaretti, and C.-. I. L. S. Group, "Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound," *Journal of Ultrasound in Medicine*, vol. 40, no. 6, pp. 1269–1275, 2021.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.
[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, PMLR, 2015.
[9] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, pp. 1058–1066, PMLR, 2013.
[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[12] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1998.
[13] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
[14] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, 1986.
[15] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
[16] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[18] O. Russakovsky, J. Deng, Z. Huang, A. C. Berg, and L. Fei-Fei, "Detecting avocados to zucchinis: what have we done, and where are we going?," in *Proceedings of the IEEE international conference on computer vision*, pp. 2064–2071, 2013.

Appendix

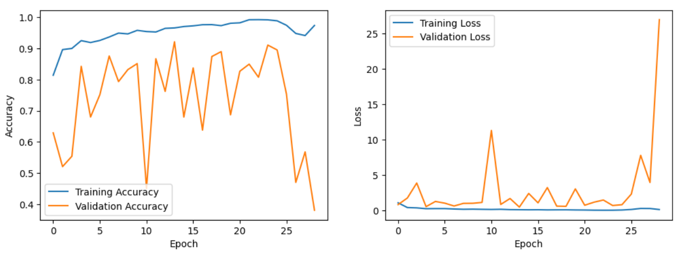

Fig. 10: CNN

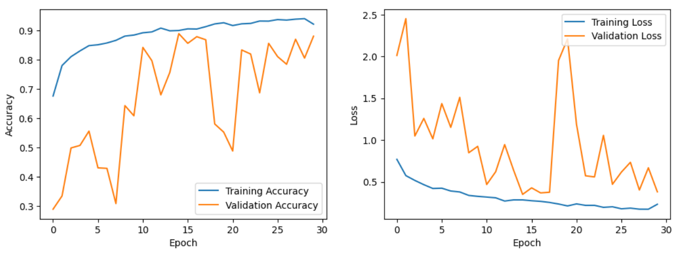

Fig. 11: CNN-LSTM

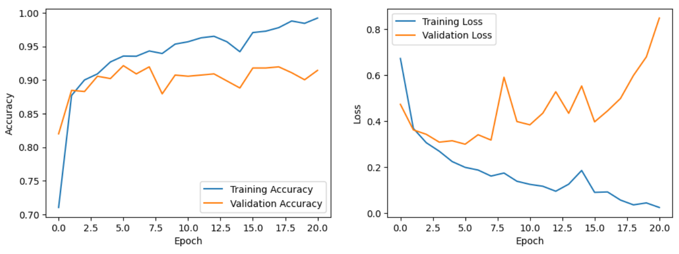

Fig. 12: VGG9

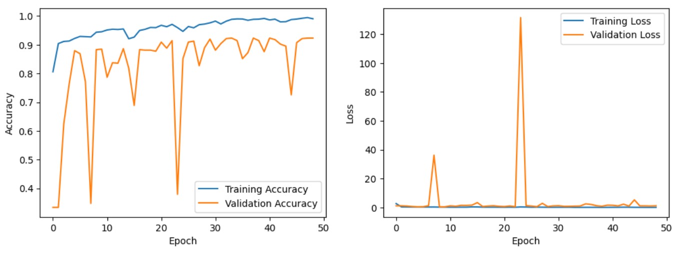

Fig. 13: VGG9 with Regularization.

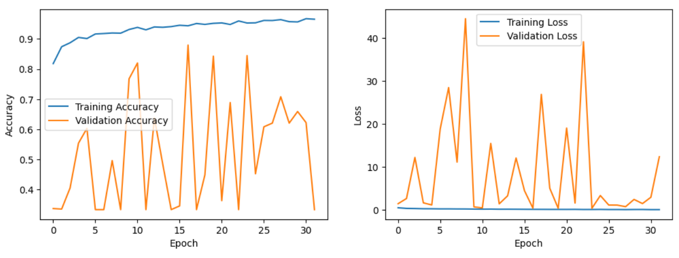

Fig. 14: COVID-Net

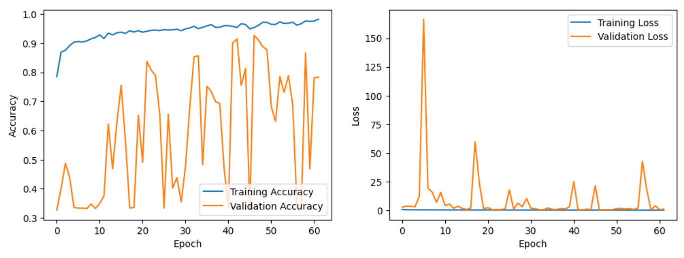

Fig. 15: COVID-Net with regularization.

Model	Class	Prec.	Rec	F1
CCN	Nor	0.83	0.97	0.90
	Pne.	0.99	0.91	0.95
	Cov.	0.97	0.88	0.92
CCN LSTM	Nor	0.84	0.93	0.88
	Pne.	0.87	0.92	0.89
	Cov.	0.99	0.84	0.91
VGG9	Nor.	0.87	0.92	0.89
	Pne.	0.89	0.96	0.92
	Cov.	0.99	0.85	0.92
VGG9 +Reg	Nor.	0.86	0.96	0.90
	Pne.	1.00	0.94	0.97
	Cov.	0.95	0.89	0.92
Covid-Net+Reg	Nor	0.85	0.98	0.91
	Pne.	0.99	0.88	0.94
	Cov.	0.93	0.88	0.91

TABLE 4