NOVA IMS Information Management School

MAA

Mestrado em Métodos Analíticos Avançados
Master Program in Advanced Analytics

Image Classification for COVID-19 Chest X-Ray Diagnosis

Deep Learning

Fábio Lopes (20200597)

Felipe Costa (20201041)

Jorge Pereira (20201085)

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

1. INTRODUCTION

Starting on the last year, the world has been suffering one of the most difficult times in the recent history. The COVID-19 pandemic exposed situations where the governments were not prepared to deal with. The high transmission rate almost led to a total breakdown of the health system in some countries. It was clear that identify and diagnose the cases earlier is a key point to stop the disease spread.

Several authors started to study this field to help with the current situation, and to be prepared to deal with further moments like this one. Image Recognition systems were one part of some of these studies: since the virus affects the lungs, researchers started to create Deep Learning models to diagnose COVID-19 with Chest Lungs X-Ray.

This project aims to create a Deep Learning model to diagnose COVID-19 using chest lung x-ray. The following data sources were used to conduct the experiments:

- Figure 1 COVID-19 Chest X-Ray Dataset Initiative from Chung [1]: A public dataset available to research purpose with 56 images which 36 of them are positive COVID-19 cases.
- COVID-19 image data collection from Cohen [2]: Public dataset of chest X-Ray and CT images of patients which are positive or suspected of COVID-19 or other viral and bacterial pneumonias. The images are collected from public sources and from hospitals and physicians.
- COVID-19 Radiography Database from Rahman [3]: Dataset created by a team or researchers from Qatar University, Doha, Qatar and the University of Dhaka, Bangladesh along with collaborators from other countries in collaboration with medical doctors. The database contains data with positive COVID-19 cases and with other diseases: 3616 positive COVID-19 cases along with 10,192 Normal, 6012 Lung Opacity (Non-COVID lung infection), and 1345 Viral Pneumonia images.

Python Program Language was used to implement the model. The Keras library implementation was used to all experiments. The study compared and analyzed different approaches to find the one that had the highest performance among the others. A transfer learning approach was also considered using already trained networks. The first set of experiments were composed by a CNN (Convolutional Neural Network) build from scratch and trained with different combinations of parameters. The other set of experiments was composed by a transfer learning approach using previously trained networks.

After testing different models and different combination of parameters one model of each group was selected to be analyzed with more details. The accuracy, auc and loss were the metrics used to evaluate the model. At the end the model with best performance was selected.

The source code of the project is available on Github: https://github.com/felipe-costa/CovidChestXRay.

2. LITERATURE REVIEW

2.1. TRANSFER LEARNING

2.1.1. What is Transfer Learning?

In a classic supervisioned learning task, models are built with the assumption that we are provided labelled samples of a particular task and domain. Using the models built, with a certain degree of confidence, one can expect the model to perform well on unseen samples of the same task and domain. But, when there is a need to perform learning on a different task from a different domain, one needs to repeat the processed of training a new model. This illustration can be seen on Figure 1 where, for each task and domain, a new model needs to be created and then applied on unseen samples. This approach is valid, if one has sufficient data to be able to train the model.

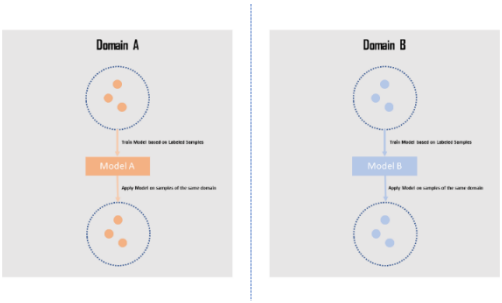

Figure 1 - Classic Supervisioned Learning

By contrast, in Transfer Learning the knowledge obtained by training a model to a specific task and domain is applied on a different problem of interest, as seen on Figure 2. For Sakar [4], "Transfer Learning is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones".

This process tries to mimic the process of learning in nature, and one can relate to the approach by understanding that the more knowledge one possesses on a range of domains, the easier it is to acquire new skills from related domains. For example, one can apply knowledge of math and statistics to learn machine learning.

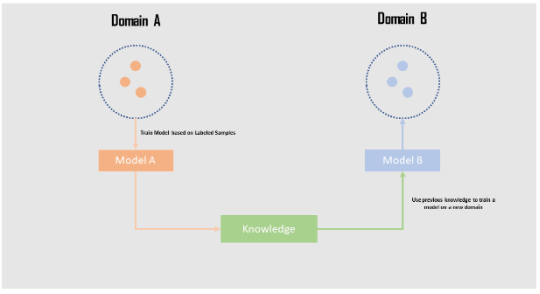

Figure 2 -Transfer Learning Process

This approach is especially important for Deep Learning models since they require vast amounts of data to be able to solve complex problems and, acquiring and labelling all this data would take considerable effort and time. As reference, the "*Imagenet*" dataset, [5], contains

millions of images labelled in different categories but it is the result of years of effort from the authors at Standard University. Obtaining such a dataset for every task that one tries to solve is not realistic so, we can use the state-of-the-art models which have high accuracy on their domain and leverage that knowledge to solve new tasks.

There are three possible benefits that Transfer Learning can bring to a new learning task [6]:

1. **Higher Start**: The initial skill of the origin model is much higher than an untrained new model.
2. **Higher Slope**: The rate of improvement as training continues is steeper.
3. **Higher Asymptote**: The converged skill of the trained model is higher.

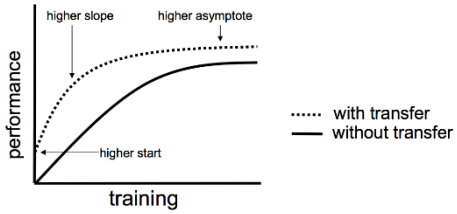

Figure 3 - Three benefits of using Transfer Learning, taken from [7]

2.1.2.Using Pre-Trained Models as Feature Extractions

One of the possible applications of Transfer Learning is to include a pre-trained model (which has been trained on millions of samples) as a Feature Extractor. The goal of this approach is to take advantage of the generalizing capabilities of a pre-trained model on a specific domain and use them to perform different tasks.

Deep Learning models have a layered architecture in which, at each layer, learn different features from the input data. These layers are finally connected to a last layer, in this case fully connected, to be able to produce an output. This layered characteristic allows a user to use a pre-trained model without its final layer and create a shallow classifier layer in its place, essentially using the pre-trained model as features to a classifier, as seen in Figure 4.

The only weights updated in the training process are those of the final layer, since the weights of the pre-trained model are frozen, thus retaining its knowledge and applying it to a new model.

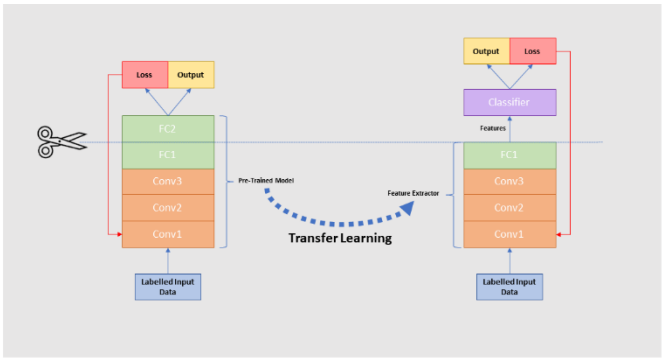

Figure 4 - Using Transfer Learning for Feature Extraction

3. METHODOLOGY

3.1. BUILDING TRAINING AND TESTING DATASET

The three datasets acquired have a mix of classifications. Some focus on Covid vs Non Covid, while others contain other diseases like Pneumonia. To simplify the task, the metadata was analyzed, and all the images were split between 2 categories, COVID vs No COVID (where besides healthy x rays we can find examples of other diseases).

3.2. EXPERIMENTS

On this experiment a network with a basic architecture was created. This network has four convolutional layers and four pooling layers. The idea is to train this network with different combination of parameters to find out the best possible combination.

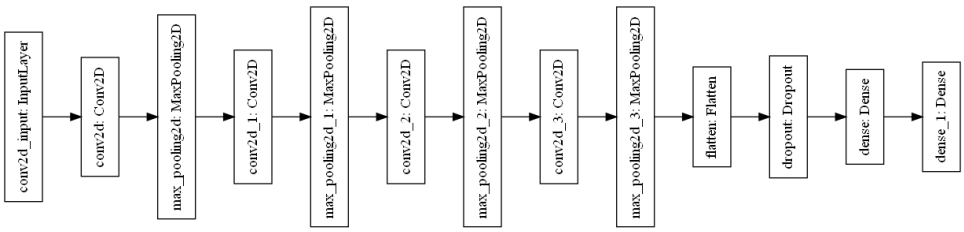

Figure 5 – Custom Convolutional Neural Network

For Transfer Learning, 4 pre-trained models will be considered: VGG16 [8], XCeption [9], DenseNet169 [10], InceptionV3 [11].

3.3. EVALUATION METRICS

To evaluate the models being trained, 3 metrics will be taken into consideration.

1. **Minutes per epoch during training**: Having a sense of how long it takes to train each epoch will allow for a performance comparison between the models.
2. **Accuracy**: Accuracy is one of the most common metrics used on classification problems. Models with high accuracy will be preferred.
3. **Area under the ROC curve (AUC)**: Since the dataset is unbalanced, accuracy might provide an incorrect representation of the true performance of the model since it will naturally be better at identifying samples of the most represented class. For that reason, AUC is chosen to understand how much the model can separate the 2 classes of the task.

3.4. TESTING APPROACH

This process was separated in two different groups:

1. Custom CNN: A list of parameters were defined, and different values were configured in 15 groups. The network was trained with each one of these 15 group of parameters and the results were collected to further analysis.
2. Transfer Learning Models: For each one of 4 the pre-trained a new model was created using one of them as the baseline. The network was trained, and the results were collected as well to further analysis.

4. RESULTS AND DISCUSSION

4.1. INITIAL EXPERIMENTS

The training process took in average 2 hours per network. To speed up this process the groups were divided in two different machines that ran the training process in parallel. On [Table 1] is possible to observe the results of the training process with each set of parameters. The best model for the Custom CNN model was the Run nº 9. This model has the best accuracy on validation and training sets. For the Transfer Learning group, the best model was the VGG16.

# Run	model_name	execution_time	minutes_per_epoch	accuracy	val_acc	auc	val_auc	loss	val_loss	input_shape	filter	batch_size	activation_output	epochs	optimizer	learning_rate
1	Custom	37.1506	1.8575	0.8723	0.8829	0.8948	0.9190	0.3015	0.2743	(150, 150, 3)	(3, 3)	50	sigmoid	20	RMSprop	0.0001
2	Custom	74.7839	3.7392	0.8712	0.8891	0.8991	0.9228	0.2990	0.2705	(256, 256, 3)	(3, 3)	50	sigmoid	20	RMSprop	0.0001
3	Custom	80.1755	4.0088	0.8099	0.8100	0.4963	0.5000	0.4870	0.4868	(256, 256, 3)	(3, 3)	50	sigmoid	20	RMSprop	0.0100
4	Custom	91.1122	4.5556	0.8465	0.8308	0.8486	0.8847	0.3604	0.3814	(256, 256, 3)	(3, 3)	50	sigmoid	20	RMSprop	0.0010
5	Custom	142.1538	7.1077	0.8426	0.8470	0.8355	0.8446	0.4006	0.3623	(256, 256, 3)	(5, 5)	50	sigmoid	20	RMSprop	0.0010
6	Custom	108.7613	5.4381	0.8096	0.8098	0.6132	0.6560	0.4968	0.4860	(256, 256, 3)	(8, 8)	50	sigmoid	20	RMSprop	0.0010
7	Custom	99.6399	4.9820	0.8091	0.8095	0.5708	0.5651	0.5179	0.4890	(256, 256, 3)	(12, 12)	50	sigmoid	20	RMSprop	0.0010
8	Custom	155.1138	5.1705	0.8870	0.8977	0.9152	0.9317	0.2769	0.2759	(256, 256, 3)	(3, 3)	50	sigmoid	30	RMSprop	0.0001
9	Custom	268.7368	6.7184	0.9034	0.9124	0.9342	0.9511	0.2468	0.2193	(256, 256, 3)	(3, 3)	50	sigmoid	40	RMSprop	0.0001
10	Custom	147.3953	7.3698	0.8570	0.8553	0.8699	0.8701	0.3285	0.3318	(256, 256, 3)	(8, 8)	50	sigmoid	20	adam	0.0010
11	Custom	162.2690	8.1135	0.8099	0.8100	0.5031	0.5000	0.4867	0.4862	(256, 256, 3)	(8, 8)	50	sigmoid	20	adam	0.0010
12	Custom	148.9797	7.4490	0.8099	0.8103	0.5000	0.5000	0.3977	0.3785	(256, 256, 3)	(8, 8)	50	softmax	20	adam	0.0010
13	Custom	150.1118	7.5056	0.8099	0.8095	0.5000	0.5000	0.3833	0.3695	(256, 256, 3)	(8, 8)	50	softmax	20	adam	0.0010
14	Custom	170.6974	8.5349	0.1901	0.1903	0.5000	0.5000	0.0000	0.0000	(256, 256, 3)	(5, 5)	50	sigmoid	20	RMSprop	0.0010
15	XCeption	107.3823	3.5794	0.9043	0.9017	0.9672	0.9624	0.2392	0.2530	(224, 224, 3)	NaN	100	softmax	30	adam	0.0001
16	InceptionV3	104.1918	3.4731	0.9074	0.8983	0.9675	0.9624	0.2355	0.2510	(224, 224, 3)	NaN	100	softmax	30	adam	0.0001
17	DenseNet169	115.5197	3.8507	0.9315	0.9309	0.9800	0.9789	0.1837	0.1876	(224, 224, 3)	NaN	100	softmax	30	adam	0.0001
18	VGG16	128.3601	4.2787	0.9368	0.9365	0.9834	0.9804	0.1644	0.1772	(224, 224, 3)	NaN	50	softmax	30	adam	0.0001
19	InceptionV3	394.4333	3.9443	0.9402	0.9085	0.9863	0.9671	0.1513	0.2403	(224, 224, 3)	NaN	100	softmax	100	adam	0.0001

Table 1 – Training Process Results

Other models also had a good performance, however with overfitting. The [Fig. 6] show the difference between the accuracy on validation and training set for the InceptionV3 model for each epoch. At epoch 15, the validation accuracy stops improving, while the training accuracy keeps increasing until reaching a plateau. Which means that the model stopped improving the ability to generalize to unseen data.

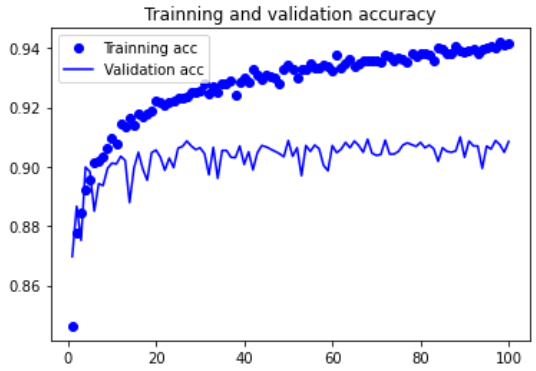

Figure 6 - Overfitting while training the InceptionV3 network

4.2. DEEPER DIVE ON THE CHOSEN MODELS

4.2.1.Custom CNN

The value of accuracy and auc increased with the duration of the training, remaining comparable through it. In the end, the model did not overfit (Figure 5 and Figure 6) and achieved an accuracy of 0.9034% and auc of 0.9342% in the training data and an accuracy of 0.9124% and auc of 0.9511% in the validation data. To train this model, it took 268.74 minutes with the average duration of each epoch being 6.72 minutes.

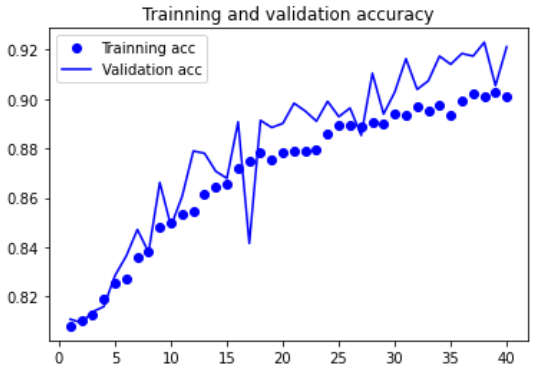

Figure 5 - Custom CNN, Training and Validation Accuracy

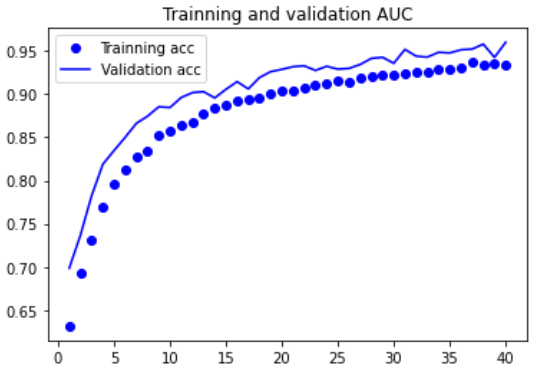

Figure 6 - Custom CNN, Training and Validation AUC

This model contains a total of 17,019,073 parameters and, all of them are trainable, since the network is being trained from scratch, without any previous knowledge acquired.

4.2.2.Transfer Learning with VGG16

The value of accuracy and auc increased with the duration of the training, remaining comparable through it. In the end, the model did not overfit (Figure 7 and Figure 8) and achieved an accuracy of 0.9368% and auc of 0.9863% in the training data and an accuracy of 0.9365% and auc of 0.9804% in the validation data. To train this model, it took 128.36 minutes with the average duration of each epoch being 4.27 minutes.

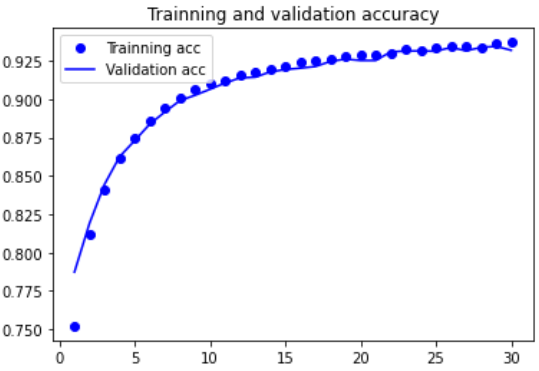

Figure 7 - VGG16, Training and Validation Accuracy

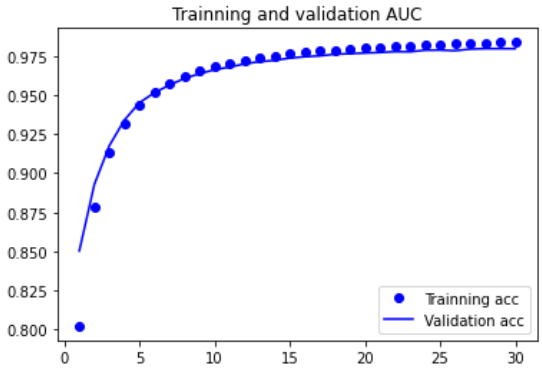

Figure 8 - VGG16, Training and Validation AUC

This model has depth of 26 and contains a total of 14,715,714 parameters and, of those, only 1026 are trainable, since the origin parameters of the VGG16 model remained frozen and untrainable.

4.2.3.Comparison

The VGG16 model took almost half of the time of the custom model to be trained with less epochs. When looking at the advantages of using Transfer Learning described in section 2.1.1, and analysing Figure 9, the training process closely follows the expected behaviour of using Transfer Learning versus a custom-built network. From the 3 possible benefits of Transfer Learning, this experiment shows 2, higher asymptote of skill which means that the knowledge pre-acquired by the network is allowing the network to better predict classes and, there is a higher slope in the training process resulting in less time spent until the network reaches that level of skill.

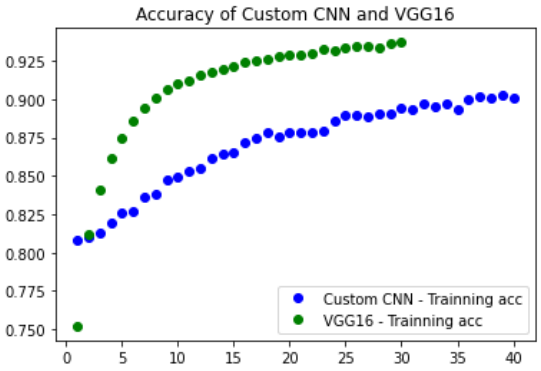

Figure 9 - Training Accuracy comparison between Custom CNN and VGG16

One aspect of the training which was particularly interesting was the GPU usage while training both models. While the Custom CNN saw the GPU hovering around 30% utilization, for the VGG16 network this value rose to the high 90 on the same GPU. This means that using the VGG16 network as a base is better at fully utilizing the hardware available while the training is being performed.

Both models were trained with the same small dataset available but, since the VGG16 network was previously trained on millions of imagens (which occurred for weeks) it is more capable on identifying the patterns in the image for classification. With a significant increase of the number of imagens there is a possibility of the Custom Model approaching the values of accuracy of the VGG16 network but, due to the nature of the task, acquiring millions of imagens is near impossible. Besides the problem of acquisition of data, one would need ample time to let the training of such network take its course.

In summation, the use of a pre-trained network, like the VGG16, allowed the model to perform better, with less time to train and with better use of the hardware available. All of it, having access to the same limited input data.

Dealing with the long training time was one of the most challenging aspects on this project. To be able to finish the experiments in a reasonable time, was necessary to use two computers in parallel. For future works the idea is to classify other diseases besides COVID-19, and, add a finetuning of the VGG16 parameters by unfreezing the parameters and retraining the model.

5. BIBLIOGRAPHY

[1] Chung A., " COVID-19 Chest X-ray Dataset Initiative". https://github.com/agchung/Figure1-COVID-chestxray-dataset, 2020.

[2] Cohen J. P., "COVID-19 image data collection". https://github.com/ieee8023/covid-chestxray-dataset, 2020.

[3] Tawsifur, R. (2020, March). COVID-19 Radiography Database, Version 4. https://www.kaggle.com/tawsifurrahman/covid19-radiography-database

[4] Sarkar, D. (2018, November 17). A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. Medium. https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

[5] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

[6] Brownlee, J. (2019, September 16). A Gentle Introduction to Transfer Learning for Deep Learning. Machine Learning Mastery. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[7] Soria Olivas, E., Guerrero, J. D. M., Martinez Sober, M., & Benedito, J. R. M. (Eds.). (2009). Handbook of research on machine learning applications and trends: Algorithms, methods and techniques. Hershey, PA: Information Science Reference.

[8] Karen Simonyan, & Andrew Zisserman. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition.

[9] (2017). Xception: Deep Learning with Depthwise Separable Convolutions.

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, & Kilian Q. Weinberger. (2018). Densely Connected Convolutional Networks.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, & Zbigniew Wojna. (2015). Rethinking the Inception Architecture for Computer Vision.

[12] Sebastian Ruder, "Transfer Learning - Machine Learning's Next Frontier". http://ruder.io/transfer-learning/, 2017.

[13] Pan, S.J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22, 1345-1359.