



QUANTITATIVE
&
FINANCIAL MODELLING
(QFM)

PROJECT S2 IN DATA SCIENCE

**Application of MLOps practices for
COVID-19 detection form X-ray
images**

prepared by :

TINA Djara Olivier

GUEDAD Mouad

Supervised by :

ERRATTAHI Rahhal

Jury :

KALLOUBI Fahd

Contents

1	Introduction	3
2	State of the Art	4
2.1	Article 1 : Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study	4
2.2	Article 2 :Investigating generalisation in automatic COVID-19 detection using deep learning	5
3	Experimental Set-up	6
3.1	Dataset description	6
3.2	Class distribution	7
4	Result and Discussion	8
4.1	Pre-trained models comparison	8
4.2	Classification layers (Best model)	8
4.2.1	DenseNet169	9
4.2.2	VGG16	9
4.2.3	Comparison	10
4.3	Fine Tuning	10
4.4	Data Preprocessing	14
4.4.1	CLAHE Equalization (Contrast Limited Adaptive Histogram Equalization)	14
4.4.2	Intensity Normalization	15
4.4.3	Image Rescaling	15
4.4.4	Preprocessing Code	15
4.4.5	Preprocessing results	15
5	Conclusion	17

Abstract

This project focuses on developing an automated tool for COVID-19 detection from chest X-ray images using deep learning techniques. The goal is to implement MLOps practices, specifically model tracking using MLflow, to streamline the development, training, validation of the COVID-19 detection model.

The project utilizes pre-trained Convolutional Neural Networks (CNNs) such as VGG-16 and DenseNet169 to analyze chest X-ray images and identify COVID-19 cases. Data augmentation techniques are employed to address the challenge of class imbalance in the dataset. Key contributions of this project include a comparison of pre-trained CNN models for COVID-19 detection, the implementation of MLOps practices using MLflow for model tracking, and the exploration of data augmentation techniques to improve performance on imbalanced datasets.

By developing an accurate and efficient tool for early COVID-19 detection, this project aims to assist healthcare professionals in making informed decisions and contribute to effective pandemic management.

Keywords: COVID-19 detection, chest X-ray images, deep learning, MLOps, MLflow, data augmentation, Transfert Learning.

1 Introduction

The COVID-19 pandemic has created a global health crisis, demanding rapid and accurate diagnostic methods. Traditional diagnostic approaches, such as RT-PCR, have limitations in terms of speed, cost, and availability. As a result, medical imaging techniques, particularly chest X-ray images, have emerged as a potential alternative for screening COVID-19 cases due to their accessibility and relatively fast results.

Deep learning algorithms, specifically Convolutional Neural Networks (CNNs), have demonstrated promising results in medical image analysis, including pneumonia detection in chest X-ray images. Several studies have explored the application of deep learning models for automated COVID-19 diagnosis using X-ray images, achieving high accuracy rates.

This project builds upon the existing research and focuses on implementing MLOps (Machine Learning Operations) practices for the development and deployment of an efficient and accurate COVID-19 detection model. MLOps is a set of practices and tools that aim to streamline the machine learning lifecycle, including model development, training, validation, deployment, and monitoring.

We utilize pre-trained CNN models, such as VGG-16 and DenseNet169, for the detection of COVID-19 cases from chest X-ray images. The performance of these models is evaluated using various metrics, including accuracy, sensitivity, and specificity. MLflow, a tool for managing and tracking machine learning experiments, is employed to monitor and compare the performance of the models across different runs.

The rest of this report is organized as follows: Section 2 provides a review of the state of the art in automated COVID-19 detection using deep learning techniques. Section 3 presents the experimental setup, including the dataset description and class distribution. Section 4 discusses the results and analysis of pre-trained models' performance, classification layers, fine-tuning techniques and data preprocessing. Finally, Section 5 concludes the report and discusses future directions for this research.

2 State of the Art

2.1 Article 1 : Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study

This article discusses the application of deep learning techniques for the detection of COVID-19 cases using chest X-ray images. The COVID-19 outbreak has created a global health crisis, and traditional diagnostic methods such as RT-PCR have limitations in terms of speed, cost, and availability. Medical imaging techniques, particularly chest X-ray images, have been widely used for screening COVID-19 cases due to their accessibility and relatively fast results.

Deep learning algorithms, specifically Convolutional Neural Networks (CNNs), have been successfully applied in medical image analysis, including pneumonia detection in chest X-ray images. Several studies have explored the use of DL models for automated COVID-19 diagnosis using X-ray images, achieving high accuracy rates.

The article proposes a DL-based automated COVID-19 screening method using chest X-ray images. It compares the performance of eight pre-trained CNN models (VGG-16, AlexNet, GoogleNet, MobileNet-V2, SqueezeNet, ResNet-34, ResNet-50, and Inception-V3) by considering various factors such as batch size, learning rate, number of epochs, misclassification rate, and optimization techniques. The models have been validated on publicly available chest X-ray images and the best performance is obtained by ResNet-34 with an accuracy of 98.33

The dataset used for validation includes chest X-ray images collected from the covid-chestxray-dataset and the ChestX-ray8 datasets. Data augmentation techniques are employed to address data scarcity and imbalance issues.

The results of the experiments demonstrate the effectiveness of the pre-trained CNN models in COVID-19 detection. The best performing model is identified, which can serve as a valuable tool for accurate and efficient diagnosis of COVID-19 infection, aiding radiologists in their decision-making process.

The contributions of this research include the comprehensive comparison of different pre-trained CNN models, addressing data scarcity and imbalance problems through data augmentation and normalization, and the development of an end-to-end CNN-based solution for early detection of COVID-19.

Overall, this study highlights the potential of deep learning techniques, specifically CNN models, in the automated detection of COVID-19 cases using chest X-ray images. The proposed approach offers a promising alternative for rapid and accurate diagnosis, assisting healthcare professionals in managing the global pandemic effectively.

2.2 Article 2 :Investigating generalisation in automatic COVID-19 detection using deep learning

Introduction : The COVID-19 virus, belonging to the coronavirus family, was first identified in China in December 2019. It spreads through respiratory droplets and can cause symptoms like fever, cough, and shortness of breath. This study focuses on using transfer learning to develop an automated tool for COVID-19 screening from X-ray images, aiming to assist radiologists and provide preliminary assessments.

BACKGROUND AND RELATED WORK : In the background and related work, various studies on automatic COVID-19 detection from CT-scan and chest X-ray images are discussed. Promising results have been achieved using deep learning and transfer learning models. The focus of this paper is on chest X-ray images due to their lower cost and greater availability compared to CT-scans. Different approaches have been proposed, with deep learning models achieving high accuracies ranging from 88.9 to 98.75. However, there is a lack of systematic evaluation and validation of these approaches by radiologists, and the datasets used are often unbalanced and limited in terms of COVID-19 cases.

MATERIALS AND METHODS : A. The researchers compiled a dataset of 4187 thoracic images from various sources, including the COVID-19 Radiography Database, Chest Imaging data collection, RSNA Pneumonia Detection Challenge, and Chest X-rays Radiopaedia. The dataset is balanced and includes 1227 COVID-19 images, 2910 normal images, and images of other pneumonia cases.

B. To address the training data dependency issue, two datasets were created. The first dataset consists of 3429 images from sources (1) and (2) and is divided into training, development, and test subsets. The second dataset includes 758 images from sources (3) and (4) and serves as an additional evaluation set.

C. The model architecture used in this work is based on the Inception-V3 model pre-trained on the ImageNet dataset. The pre-trained model’s classification part was removed, and a customized architecture was built for the specific number of classes. The customized architecture includes an average pooling layer, two fully connected layers with dropout, and a softmax layer for classification.

D. The performance evaluation of the model includes classification accuracy, sensitivity, and specificity. Sensitivity measures the model’s ability to correctly classify COVID-19 cases, while specificity measures its ability to identify normal cases. These metrics are calculated based on true positive, false negative, true negative, and false positive classifications.

RESULTS AND DISCUSSION: The Inception-V3 model, trained using transfer learning and fine-tuning techniques, demonstrated a remarkable accuracy of 98.98 in detecting COVID-19. Further enhancement was achieved by applying Contrast Limited Adaptive Histogram Equalization (CLAHE) to improve generalization, resulting in an accuracy of 81.27 on a new test set. Notably, the model exhibited a high specificity of 100 while maintaining a moderate sensitivity of 36.67 in multi-class COVID-19 detection. These findings highlight the potential of this approach for efficient primary screening in resource-limited settings where test kits and medical expertise may be limited.

Conclusion : In conclusion, the study explored the effectiveness of transfer learning and fine-tuning using the Inception-V3 model for COVID-19 detection from X-ray images. Fine-tuning the convolutional layers yielded superior performance compared to the baseline model. Additionally, applying CLAHE preprocessing to COVID-19 data significantly improved performance, particularly on the second test set, increasing accuracy from 16.62 to 81.27. These findings suggest that the proposed approach holds promise for primary COVID-19 screening in resource-limited settings

3 Experimental Set-up

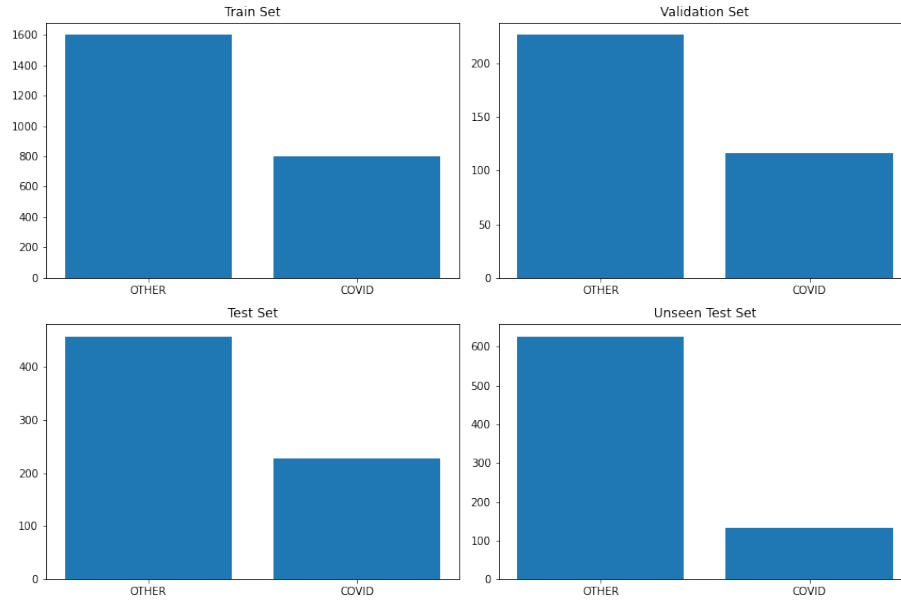
3.1 Dataset description

The dataset used in this project comprises medical imaging data obtained from four different sources: Radiopaedia, RSNA, SIRM, and Twitter. It is divided into four distinct sets, namely the train set, validation set, test set, and unseen test set.

The train set consists of 2,401 image filenames, and the validation set contains 343 image filenames. Both sets are labeled with two classes for binary classification. The test set contains 685 image filenames, and the unseen test set contains 758 image filenames. Similar to the train and validation sets, the test sets are also categorized into the same two classes for binary classification.

Originally, the dataset encompass three classes: COVID, pneumonia, and normal. However, for the purpose of this project, the pneumonia and normal classes have been combined, resulting in a binary classification task.

3.2 Class distribution



We noticed that the dataset exhibits class imbalance across each set, indicating that the distribution of samples among the different classes is uneven.

Firstly, it can lead to biased model performance, as the model may favor the majority class due to the larger number of samples. This can result in lower accuracy, precision, recall, and other performance metrics for the minority class.

Secondly, the model may struggle to learn and generalize patterns from the minority class due to limited exposure to its samples. As a result, the model's ability to correctly classify instances from the minority class may be compromised.

Furthermore, class imbalance can impact the training process, potentially causing longer training times and unstable convergence. The model may require more iterations to find an optimal solution due to the dominant influence of the majority class.

4 Result and Discussion

4.1 Pre-trained models comparison

	MODELS PERFORMANCES								
	Test			Unseen test			Dev		
Models	Accuracy	Sensitivity	Spec	Accuracy	Sensitivity	Spec	Accuracy	Sensitivity	Spec
InceptionV3	98.54	99.34	96.92	31.79	24.6	65.9	56.56	67.84	34.48
VGG16	94.16	96.93	88.59	70.31	70.92	67.42	55.39	66.51	33.62
VGG19	91.38	94.09	85.96	63.19	66.61	46.96	54.81	65.19	34.48
ResNet50	91.67	93.43	88.15	36.14	29.55	67.42	56.27	67.4	34.48
DenseNet169	98.83	99.34	97.8	72.42	77.15	50	56.27	66.96	35.34
DenseNet121	98.1	99.34	95.61	34.56	29.39	59.09	56.56	67.4	35.34
MobileNetV2	98.97	99.56	97.8	63.45	73	18.18	57.73	69.16	35.34
Xception	95.91	97.37	92.98	39.7	41.05	33.33	56.27	67.4	34.48
InceptionRestNetV2	98.83	99.34	97.8	27.04	21.08	55.3	56.56	67.84	34.48

To assess the performance of the different models, we focused on the accuracy score, sensitivity and specificity of unseen test data. Two models stood out with the best results in this metric:

VGG16: With an accuracy score of 70.31, , a sensitivity score of 70.92 and a specificity score of 67.42 on unseen test data, the VGG16 model demonstrated its ability to generalize and perform well on unknown data. Its solid precision indicates a high capacity to correctly classify samples into the appropriate categories.

DenseNet169: With an accuracy score of 72.42, a sensitivity score of 77.15 and a specificity score of 50 on unseen test data, the DenseNet169 model also displayed impressive performance in terms of generalization.

4.2 Classification layers (Best model)

We used MLflow to track the performance of our two models: VGG16 and DenseNet169 which are our two best models according 4.1. We created separate experiments for each model and conducted multiple runs.

With MLflow, we were able to keep track of various configurations, hyper-parameters, and metrics associated with each run of our models. It provided us with a framework to manage and organize our machine learning experiments effectively. MLflow helped us compare and analyze the performance of these models across different runs, enabling us to identify the most effective configuration for our specific use case. Both of the experiences were conducted with the following fixed values of parameters

batch_size	16
epochs	30
learning_rate	0.0001
num_classes	2

In our transfer learning context, we defined a set of specific neurons to freeze within each "mixed" layer of our model. Freezing neurons means that their weights and parameters remain fixed during the training process, allowing us to leverage pre-trained knowledge while adapting the model to our target task.

Here is the breakdown of the frozen neurons for each "mixed" layer:

mixed0: 40 neurons
mixed1: 63 neurons
mixed2: 86 neurons
mixed3: 100 neurons
mixed4: 132 neurons
mixed5: 164 neurons
mixed6: 196 neurons
mixed7: 228 neurons
mixed8: 248 neurons
mixed9: 279 neurons
mixed10: 310 neurons

By freezing specific subsets of neurons within these layers, we aimed to strike a balance between leveraging the pre-trained knowledge encoded in the earlier layers and allowing the later layers to adapt to our target task. This selective freezing strategy helped us optimize the model's performance and prevent over-fitting, particularly when working with limited training data for our specific task.

4.2.1 DenseNet169

Run ID:	c006ff63738a4badaf8f62d981edb849	38f358444a8c4b359770477bf2fc5030	d0d7e68be41949b18ba789cfd625dbab	21487210a6c84c028829f9d6f9be1bbe
Run Name:	Model Mixed9 without data preprocessing	Model Mixed7 without data preprocessing	Model Mixed5 without data preprocessing	Model Mixed3 without data preprocessing
Test Accuracy_Score	99.27	98.98	99.71	99.42
Test Sensitivity_Score	99.56	99.12	100	100
Test Specificity_Score	98.68	98.68	99.12	98.25
Unseen Sensitivity_Score	46.01	51.76	53.83	63.74
Unseen Specificity_Score	46.21	43.18	26.52	15.91
Unseen Test Accuracy_Score	46.04	50.26	49.08	55.41

4.2.2 VGG16

Run ID:	5023da3c117146c9a9b2394b6820d46d	90146856a0a54f06a4e2ea3f3636204e	588753f395a046fe8b83b6fde8023df6	c40fac0d38f940bba3fdce14a34c1009
Run Name:	Model Mixed3 without data preprocessing	Model Mixed5 without data preprocessing	Model Mixed7 without data preprocessing	Model Mixed9 without data preprocessing
Test Accuracy_Score	93.14	92.55	93.28	93.58
Test Sensitivity_Score	96.5	97.37	96.5	97.59
Test Specificity_Score	86.4	82.89	86.84	85.53
Unseen Sensitivity_Score	70.93	75.56	66.61	78.43
Unseen Specificity_Score	57.58	51.52	62.12	58.33
Unseen Test Accuracy_Score	68.6	71.37	65.83	74.93

4.2.3 Comparison

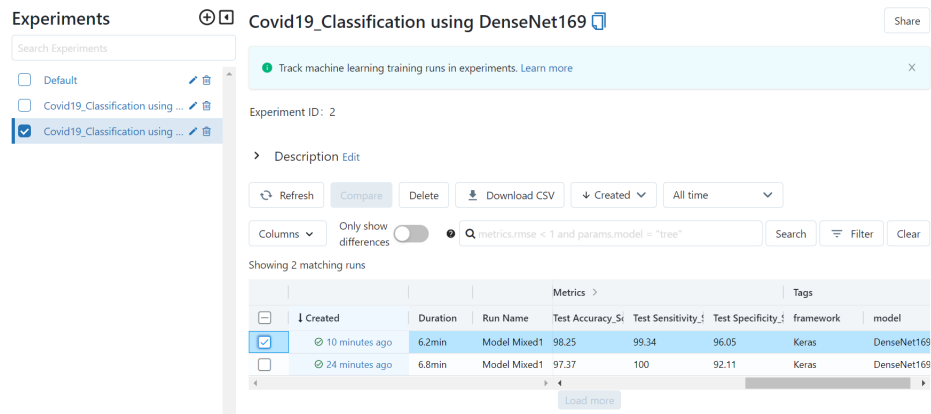
A comparison of the different runs from both experiments allows us to conclude that VGG16 performs better, as it achieves the highest accuracy on the unseen test dataset, which is 74% with mixed9 (276 neurons). At this level, VGG16 outperforms DenseNet169. Now let see what will be the effect of fine tuning and data preprocessing on our model.

4.3 Fine Tuning

In this section, our objective is to implement fine-tuning techniques in our code to determine if the performance of our model will improve, either significantly or relatively. We have a high level of confidence that the subsequent step, data preprocessing, will yield improvements.

Densenet169

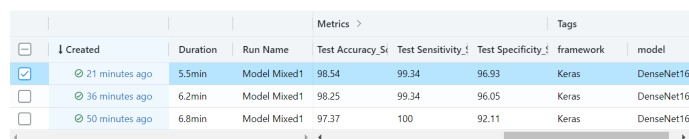
To begin, we will employ dropout with a 50 percent rate, deactivating certain neurons to maintain data balance and mitigate the risk of overfitting. Additionally, we will adjust the number of mixtures applied to 310.



The screenshot shows the 'Experiments' interface with a sidebar on the left containing a search bar and a list of experiments. The main panel displays details for the experiment 'Covid19_Classification using DenseNet169'. It includes a description, filters, and a table of runs. The table has columns for 'Created', 'Duration', 'Run Name', 'Test Accuracy_Si', 'Test Sensitivity_!', 'Test Specificity_!', 'framework', and 'model'. Two runs are shown, both with 'Model Mixed1' as the name and 'DenseNet169' as the model.

Created	Duration	Run Name	Test Accuracy_Si	Test Sensitivity_!	Test Specificity_!	framework	model
10 minutes ago	6.2min	Model Mixed1	98.25	99.34	96.05	Keras	DenseNet169
24 minutes ago	6.8min	Model Mixed1	97.37	100	92.11	Keras	DenseNet169

After implementing the dropout phase with a 50% rate for each layer in the fully connected network, we introduce an additional layer consisting of 64 neurons. This augmentation enables us to observe the resulting output below.



This screenshot shows the same 'Experiments' interface but with an additional run added to the table. The new run, created 21 minutes ago, shows improved metrics: Test Accuracy_Si of 98.54, Test Sensitivity_! of 99.34, and Test Specificity_! of 96.93.

Created	Duration	Run Name	Test Accuracy_Si	Test Sensitivity_!	Test Specificity_!	framework	model
21 minutes ago	5.5min	Model Mixed1	98.54	99.34	96.93	Keras	DenseNet169
36 minutes ago	6.2min	Model Mixed1	98.25	99.34	96.05	Keras	DenseNet169
50 minutes ago	6.8min	Model Mixed1	97.37	100	92.11	Keras	DenseNet169

As we can see, certain metrics have increased, while others have shown a decrease.

In the next step, we will replace average pooling with max pooling with the flatten layer and observe the changes in the output. We will assess if there are any increases in the performance metrics as a result of this modification.

Showing 4 matching runs

	Created	Duration	Run Name	Models	Metrics >		
	↓ Created				Test Accuracy_Si	Test Sensitivity_	Test Specifi
<input checked="" type="checkbox"/>	9 minutes ago	6.2min	Model Mixed1	keras	98.1	99.56	95.18
<input type="checkbox"/>	42 minutes ago	5.5min	Model Mixed1	keras	98.54	99.34	96.93
<input type="checkbox"/>	57 minutes ago	6.2min	Model Mixed1	keras	98.25	99.34	96.05
<input type="checkbox"/>	1 hour ago	6.8min	Model Mixed1	keras	97.37	100	92.11

Load more

Now, we are going to make the final modification by adding BatchNormalization layers after each Dense layer. The purpose of BatchNormalization is to stabilize the learning process and potentially enhance the model's generalization ability. We will observe the effects of this modification and determine if it leads to any significant improvements.

	Created	Duration	Run Name	Models	Metrics >		
	↓ Created				Test Accuracy_Si	Test Sensitivity_	Test Specificity_
<input checked="" type="checkbox"/>	11 minutes ago	5.9min	Model Mixed1	keras	97.23	98.47	94.74
<input type="checkbox"/>	27 minutes ago	6.2min	Model Mixed1	keras	98.1	99.56	95.18
<input type="checkbox"/>	1 hour ago	5.5min	Model Mixed1	keras	98.54	99.34	96.93
<input type="checkbox"/>	1 hour ago	6.2min	Model Mixed1	keras	98.25	99.34	96.05
<input type="checkbox"/>	1 hour ago	6.8min	Model Mixed1	keras	97.37	100	92.11

Load more

As observed, after adding BatchNormalization layers after each layer, the desired improvements were not achieved. In fact, all the parameters of our metrics have decreased, indicating that there are no advancements in terms of increasing the metrics.

Conclusion: After conducting all of these experiments, we have noticed that our primary goal, which is to increase the unseen score, unseen sensitivity, and unseen specificity, has not been achieved. Despite our efforts, we continue to encounter the same problem where some metrics decrease while others increase. As a result, we cannot draw any definitive conclusions from this DenseNet196 model.

However, we can still analyze and consider the most favorable result obtained is :

Name	Value
Test Accuracy_Score 🔗	97.37
Test Sensitivity_Score 🔗	100
Test Specificity_Score 🔗	92.11
Unseen Sensitivity_Score 🔗	96.01
Unseen Specificity_Score 🔗	19.7
Unseen Test Accuracy_Score 🔗	82.72
Validation Accuracy_Score 🔗	98.25
Validation Sensitivity_Score 🔗	68.28
Validation Specificity_Score 🔗	32.76

VGG16

We are going to apply the same modifications we performed on the DenseNet169 model to the VGG16 model. Once completed, we will proceed with a detailed comparison between the two models.

To begin, we will apply the same modifications to the VGG16 model as we did in the DenseNet169 model. We will employ dropout with a 50 percent rate, deactivating certain neurons to maintain data balance and mitigate the risk of overfitting. Additionally, we will adjust the number of mixtures to 310. These changes aim to enhance the performance and generalization of the VGG16 model.

				Metrics >			Tags	
<input type="checkbox"/>	↓ Created	Duration	Run Name	Test Accuracy_Si	Test Sensitivity_!	Test Specificity_!	framework	model
<input type="checkbox"/>	🕒 16 minutes ago	2.3min	Model Mixed1	67.59	100	2.632	Keras	VGG16

After implementing the dropout phase with a 50% rate for each layer in the fully connected network of the VGG16 model, we introduce an additional layer consisting of 64 neurons. This augmentation allows us to observe the resulting output below. These modifications are intended to analyze the impact on the model's performance and assess any improvements.

					Metrics >		
<input checked="" type="checkbox"/>	↓ Created	Duration	Run Name	framework	Test Accuracy_Si	Test Sensitivity_!	Validation Accuracy_Si
<input checked="" type="checkbox"/>	🕒 4 minutes ago	2.3min	Model Mixed1	keras	66.72	0	66.18
<input checked="" type="checkbox"/>	🕒 33 minutes ago	2.3min	Model Mixed1	keras	67.59	2.632	66.76

In the next step, we will replace average pooling with max pooling in the VGG16 model, specifically with the flatten layer, and observe the changes in the output. We will assess if there are any increases in the performance metrics as a result of this modification.

Afterward, we will proceed with the final modification of incorporating Batch-Normalization layers after each Dense layer in the VGG16 model. The inclusion of BatchNormalization aims to stabilize the learning process and potentially improve the model’s generalization ability. Through careful observation, we will evaluate the impact of this adjustment and determine if it yields any substantial improvements.

13

Name	Value
Test Accuracy_Score ↗	93.58
Test Sensitivity_Score ↗	93.44
Test Specificity_Score ↗	93.86
Unseen Sensitivity_Score ↗	27.16
Unseen Specificity_Score ↗	85.61
Unseen Test Accuracy_Score ↗	37.34
Validation Accuracy_Score ↗	93
Validation Sensitivity_Score ↗	60.35
Validation Specificity_Score ↗	34.48

Upon careful analysis of the VGG16 model, we found that the desired improvements were not achieved after the addition of BatchNormalization layers following each layer. Surprisingly, the introduction of these layers resulted in a decrease in all the parameters of our metrics. Consequently, it is evident that no significant advancements have been made in terms of enhancing the metrics' performance with this modification to the VGG16 model.

Conclusion: After conducting an extensive series of experiments on the VGG16 model, we regret to conclude that our primary objective of increasing the unseen score, unseen sensitivity, and unseen specificity has not been met. Despite our diligent efforts, we have encountered the persistent challenge of certain metrics exhibiting a decrease while others show an increase. In comparison to the DenseNet169 model, it is evident that the VGG16 model performs worse in achieving our desired metrics. Given these findings, we hesitate to recommend the VGG16 model, particularly after considering the preprocessing steps. Further exploration and analysis are necessary to determine alternative approaches or models that may yield more favorable results.

4.4 Data Preprocessing

In this section, we describe the different transformations applied to the images before their usage in our project. These transformations aim to improve image quality and comparability, as well as facilitate the detection of relevant features.

4.4.1 CLAHE Equalization (Contrast Limited Adaptive Histogram Equalization)

CLAHE equalization is a technique that enhances the contrast of an image by adaptively adjusting its histogram. Firstly, the image is converted from the RGB space to the LAB space. Then, the L channel (luminosity) of the LAB space undergoes CLAHE equalization. This is performed using the `clahe_equalize` function, where we specify the `clip_limit` parameter to limit contrast amplification and the `grid_size` parameter to determine the size of blocks used for contrast adjustment. After equalization, the image is converted back to the

RGB space. This transformation enhances details and makes structures more distinct.

4.4.2 Intensity Normalization

Intensity normalization is a step that aims to normalize the pixel values of the image to improve comparability across different images in the dataset. Although the exact details of the `normalize_intensity` function have not been provided, it is common to normalize pixel values within a specific range, such as $[0, 1]$. This step ensures that the images have a consistent intensity range, facilitating further processing.

4.4.3 Image Rescaling

Image rescaling is performed to standardize the size of the images in our dataset. This facilitates processing by ensuring that all images have the same size. The exact details of the `scale_image` function have not been specified, but rescaling can be achieved using techniques such as bilinear or bicubic interpolation, with the desired output size specified. This transformation normalizes the spatial resolution of the images and avoids distortions caused by different sizes.

By combining these transformations in the `preprocess_input_generator` function, we create a preprocessing pipeline that enhances visual quality, normalizes intensity, and standardizes image size.

4.4.4 Preprocessing Code

Here is the code snippet representing the preprocessing transformations:

```
def clahe_equalize(image, clip_limit=4.0, grid_size=(16, 16)):
    lab = cv2.cvtColor(image, cv2.COLOR_RGB2LAB)
    lab_planes = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=clip_limit, tileGridSize=grid_size)
    lab_planes[0] = clahe.apply(lab_planes[0])
    lab = cv2.merge(lab_planes)
    clahe_image = cv2.cvtColor(lab, cv2.COLOR_LAB2RGB)
    return clahe_image

def preprocess_input_generator(img):
    img = clahe_equalize(img, clip_limit=4.0, grid_size=(16, 16))
    # Other preprocessing techniques can be added here
    img = normalize_intensity(img)
    img = scale_image(img)
    return img
```

4.4.5 Preprocessing results

We present a comparison between the results obtained from the preprocessed data and the raw, unprocessed data using VGG16. This analysis highlights the










impact of the preprocessing steps on the data quality and provides insights into the benefits of applying these transformations. Parameters values

batch_size	16	16
epochs	50	50
learning_rate	0.0001	0.0001
num_classes	2	2

VGG16 After applying the data preprocessing techniques, we observed a substantial improvement in the previously unseen test accuracy. The accuracy increased from 70.18% to 75.59%, indicating a significant positive impact of the preprocessing steps on the performance of the model as we can verify it in the following table.

Run ID:	11bf7cdfb75b495bbe311a9bcb1929ed	fb2539ad7d7042ed8da16790b1e05247
Run Name:	Model Mixed9 with data preprocessing	Model Mixed9 without data preprocessing
Start Time:	2023-07-04 16:24:42	2023-07-04 15:31:17
End Time:	2023-07-04 16:42:11	2023-07-04 15:49:04
Duration:	17.5min	17.8min
Test Accuracy_Score	93.72	93.87
Test Sensitivity_Score	97.81	97.16
Test Specificity_Score	85.53	87.28
Unseen Sensitivity_Score	79.87	71.73
Unseen Specificity_Score	55.3	62.88
Unseen Test Accuracy_Score	75.59	70.18

DenseNet169 After applying the data preprocessing techniques, we didn't observe a substantial improvement in the previously unseen test accuracy. The accuracy remain the same, indicating a null impact of the preprocessing steps on the performance of the model.

Name	Value
Test Accuracy_Score 	97.37
Test Sensitivity_Score 	100
Test Specificity_Score 	92.11
Unseen Sensitivity_Score 	96.01
Unseen Specificity_Score 	19.7
Unseen Test Accuracy_Score 	82.72
Validation Accuracy_Score 	98.25
Validation Sensitivity_Score 	68.28
Validation Specificity_Score 	32.76

5 Conclusion

After conducting a comprehensive evaluation of various models, including Inception V3, VGG16, VGG19, ResNet50, and DenseNet169 ..., it has been unequivocally established that two models stand out for their exceptional performance and accuracy. These models are VGG16 and DenseNet169, demonstrating remarkable superiority, especially in terms of unseen test accuracy, which proves significantly higher in comparison to the other models.

Following the identification of these top-performing models, we proceeded with extensive tests during the fine-tuning phase. Through meticulous parameter adjustments, we made a compelling discovery: the DenseNet169 model consistently outperformed VGG16, exhibiting outstanding performance. This revelation has empowered us to assert the unwavering reliability of the DenseNet169 model for forthcoming experiments.

Moreover, after conducting the preprocessing phase, our confidence in the DenseNet169 model's capabilities was further bolstered. We witnessed a notable increase in terms of unseen test accuracy—a highly anticipated outcome, as succinctly stated in the conclusion of the fine-tuning phase. This significant advancement reaffirms our conviction in the robustness and potential of the DenseNet169 model for subsequent experimentation.

Given these exceptional findings, we confidently recommend the DenseNet169 model for the detection of COVID-19 through chest X-ray images. Its outstand-

ing performance, accuracy, and reliability make it a valuable tool in the field of COVID-19 diagnosis.

References

- [1] Soumya Ranjan Nayak, Deepak Ranjan Nayak, Utkarsh Sinha, Vaibhav Arora, Ram Bilas Pachori *Application of deep learning techniques for detection of COVID-19 cases using chest X-ray images: A comprehensive study*
- [2] Rahhal Errattahi ,Salmam Fatima Zahra† Asmaa El Hannani , Aqqal Abdelhak , Hassan Ouahmane , Sadik Mohamed , El Hillali Yassin§ Chouaib Doukkali University of El Jadida, National School of Applied Sciences, Laboratory of Information Technologies, El Jadida, Mohammed 5 University, ENS, Rabat, Morocco Hassan II University of Casablanca, NEST Research Group ENSEM, Casablanca, Morocco Polytechnique hauts-de-france University, France *Investigating generalisation in automatic COVID-19 detection using deep learning*