Lebanese University, Faculty Of Science Section 5

2020-2021

COVID-19 X-RAY DETECTOR X

Device that detect COVID-19 by classifying chest x-ray images using AI

Student:

Hadi Jaber

Supervisor:

Dr. Mohamad Chaitou

University:

Lebanese University, Faculty Of Science Section 5

COVID-19

COVID-19 (coronavirus disease 2019) is an infectious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), previously known as 2019 novel coronavirus (2019-nCoV), a strain of coronavirus. The first cases were seen in Wuhan, China, in late December 2019 before spreading globally. The current outbreak was officially recognized as a pandemic on 11 March 2020.No effective treatment or vaccine exists currently (April 2020).

Description

Since reverse transcription polymerase chain reaction (RT-PCR) test kits are in limited supply, there exists a need to explore alternative means of identifying and prioritizing suspected cases of COVID-19. The average time to get (RT-PCR) test result is about 2 days. Moreover, large scale implementation of the COVID-19 tests which are extremely expensive cannot be afforded by many of the developing & underdeveloped countries hence if we can have some parallel diagnosis/testing procedures using Artificial Intelligence & Machine Learning and leveraging the historical data, it will be extremely helpful. This can also help in the process to select the ones to be tested primarily.

Why Study X-Rays?

CT scans, despite being used to contribute to the clinical workup of patients suspected for COVID-19, are costly. Therefore, smaller centers may have limited access to CT scanners. X-ray machines are more affordable and can be portable and therefore a viable alternative. To be clear, We don't claim that CXRs should be relied on in the diagnosis of COVID-19. Direction from the Canadian Association of Radiologists recommend following the CDC’s guidelines that state that CXR and CT should not be used to diagnose COVID-19. The recommendations state that viral testing is currently the only means to diagnose COVID-19, even if a patient has suggestive features on CXR or CT. That being said, some research has suggested that radiological imaging (CT in particular) may be just as sensitive as RT-PCR. Such findings exhibit why we've decided to explore this avenue and build open source tools to enable further investigation. It is also important to note that normal imaging does not imply that a patient has not been infected by SARS-CoV-2, only that they aren't showing signs of COVID-19 respiratory disease.

Goals

1. Build a machine learning model that can detect COVID-19 from the XRAY images of chest.
2. Convert this model to TensorFlow lite to optimize the size and the speed of it.
3. Implement the final model in the NVIDIA Jetson Nano.
4. Build a GUI using Python Tkinter to easier the testing of images.

Data

Dataset

For the purpose of this experiment, data (total 120k images) was taken from many repositories like :

1. The COVID-19 image data collection repository on GitHub and Kaggle is a growing collection of deidentified CXRs and CT scans from COVID-19 cases internationally. We would like to take this opportunity to thank Joseph Paul Cohen and his fellow collaborators at the University of Montreal for their hard work in assembling this dataset.(5000 image)
2. The Pneumonia Images dataset available on Kaggle contains several deidentified CXRs and includes a label indicating whether the image shows evidence of pneumonia. We would like to take this opportunity to thank the Paul Mooney Developer Advocate at Kaggle Boulder, United States and all other involved entities for creating this dataset.(5000 image)
3. The Gender images collected from Kaggle too from NIH Chest X-ray Dataset.(110k image)

Data Preprocessing

Prior to training, preprocessing was implemented on the images themselves. ImageDataGenerator (from tensorflow.keras) was employed to perform preprocessing of image batches prior to training. The following transformations were applied to images:
Images were resized to have the following shape: 100×100×3. By reducing the image size, the number of parameters in the neural network was decreased.
images are normalized by scaling them so their pixel values are in the range [0, 1].
We pre-process all images before training and we re-doing that for all models (gender-pneumonia-covid19)

Binary Classification

We first considered a binary classification problem where the goal was to detect whether an X-ray shows evidence of COVID-19 infection. The classifier was to assign X-ray images to either a non-COVID-19 class or a COVID-19 class. A deep convolutional neural network architecture was trained to perform binary classification. The model was trained using the RMSprop optimizer and binary cross-entropy loss. All training code was written using TensorFlow 2.2.0.

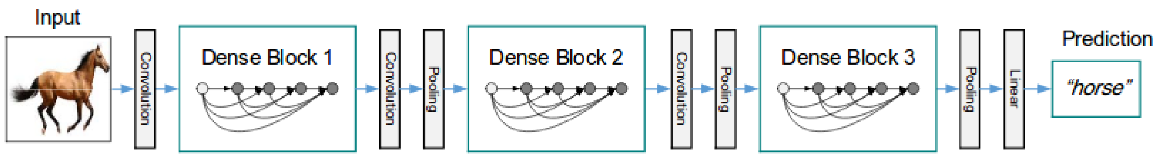
Architecture

Densenet

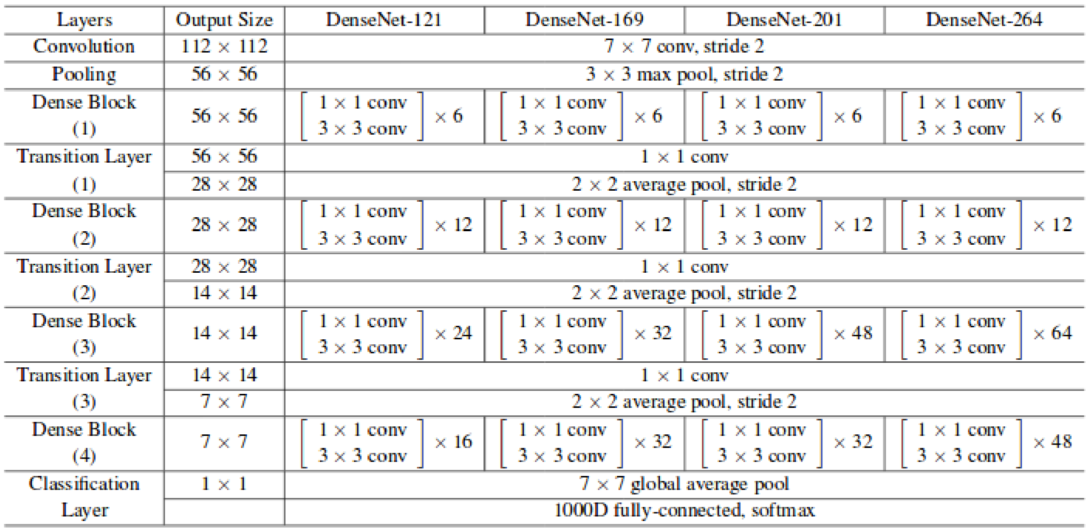
By this time around, Deep learning kind of started as replicating human vision has turned into an engineering practice, people are trying different things looking at the accuracies, network weights and visualizing features. There was a saying around this period, that NIPS conference paper acceptance has been degraded. Researchers are keeping lot of effort in engineering these networks instead of trying something different. Having said all of this, this paper is still a good read and has improved the accuracy of image classification challenges.
DenseNet is very similar to ResNet with two important changes.

1. Instead of adding up the features as in ResNet, they concat the feature maps.
2. Instead of just adding one skip connection, add the skip connection from every previous layer.

Meaning, In a ResNet architecture, we add up (l-1) and (1) layer features. In DenseNet, for Lth layer we concat all the features from [1….(l-1)] layers.
The author quotes "DenseNet layers are very narrow (e.g. 12 filters per layer) adding only a small set of feature maps to the "collective knowledge" of the network and keep the remaining feature maps unchanged- and the final classifier makes a decision based on all feature-maps in the network."



Each block contains a set of conv blocks (Called bottleneck layer if contains a 11 conv layer to reduce features followed by 33 conv layers) depending on the architecture depth. The following graph shows DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-264.

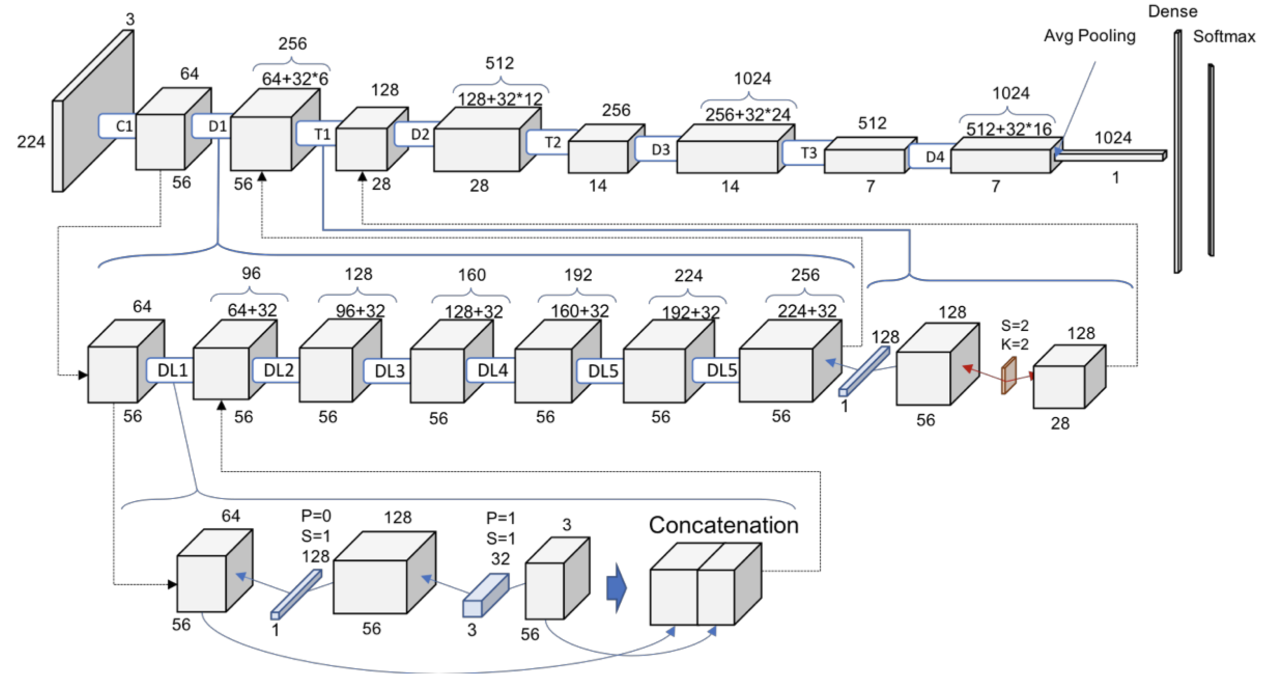


Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1\times1 \text{ conv} \\ 3\times3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Growth rate defines the number of features each dense block will began with. The above diagram uses 32 features. Having this kind of architectures reduces the parameters we train by a lot. For example a resnet conv block may contain [96, 96, 96] feature maps inside a block, where as DenseNet contains [32, 64, 96]. The author claims and has been proven by some other paper called stochastic depth network (where they train a ResNet architecture by dropping a few layers randomly every iteration) that most of the features learned by ResNet are redundant. The author basic intuition is also that connecting layers in this way where each layer

contains the features of all the previous layers, will not allow it learn redundant features.

The transition layers in the network reduces the number of features to x m, where takes values between (0, 1). x m is also used in bottleneck layers. when <0 for both transition layers and bottleneck layers it is called DenseNet-BC, when <0 for only transition layers it is called DenseNet-C.



Results on Architecture

1. It uses 3x less parameters compared to ResNet for similar number of layers.
2. Using the same set of parameters used for ResNet architectures and replacing the bottleneck layers of resnet with Dense blocks, the authors have seen similar performance on ImageNet dataset.
3. On CIFAR-10, CIFAR-100 and other datasets, DenseNet blocks have shown incremental performance.

Measures Against Overfitting

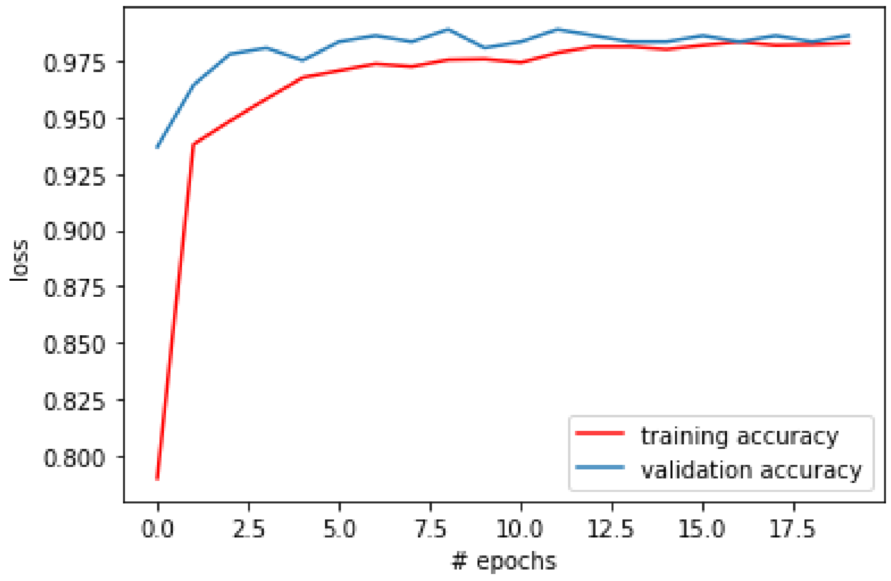
We applied multiple strategies to combat overfitting, such as dropout regularization and data augmentation (varying brightness of image by +10% and -10%).

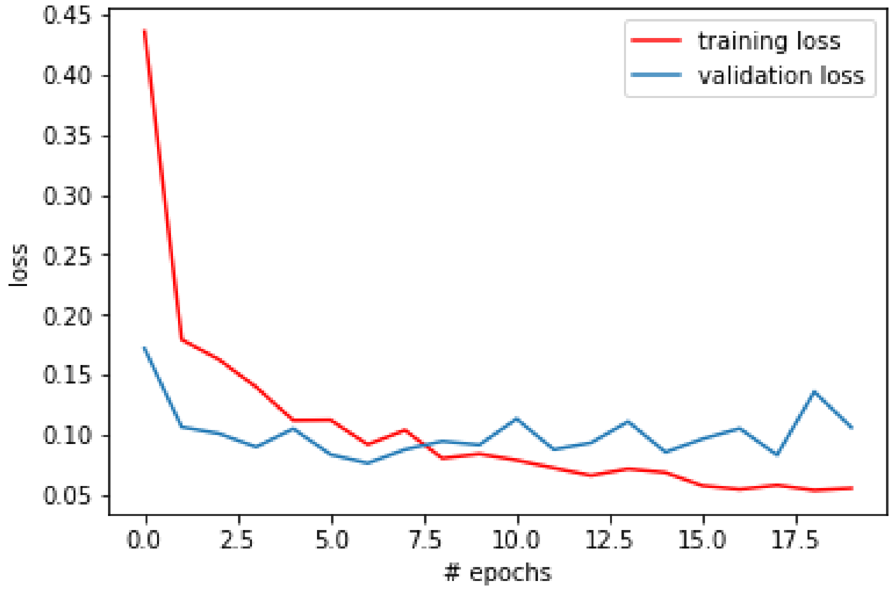
Class Imbalance

Due to the scarcity of publicly available CXRs of severe COVID-19 cases, we were compelled to apply class imbalancing methods to mitigate the effects of having one class outweighing several others. If you are versed in machine learning, you know that accuracy can be misleadingly high in classification problems when a class is underrepresented. We were left with a dilemma, as we also did not want to limit our training data to only 189 non-COVID-19 examples, as there would have been about 400 images total. As above said, we apply Image Augmentation to overcome this issue.

Training Result

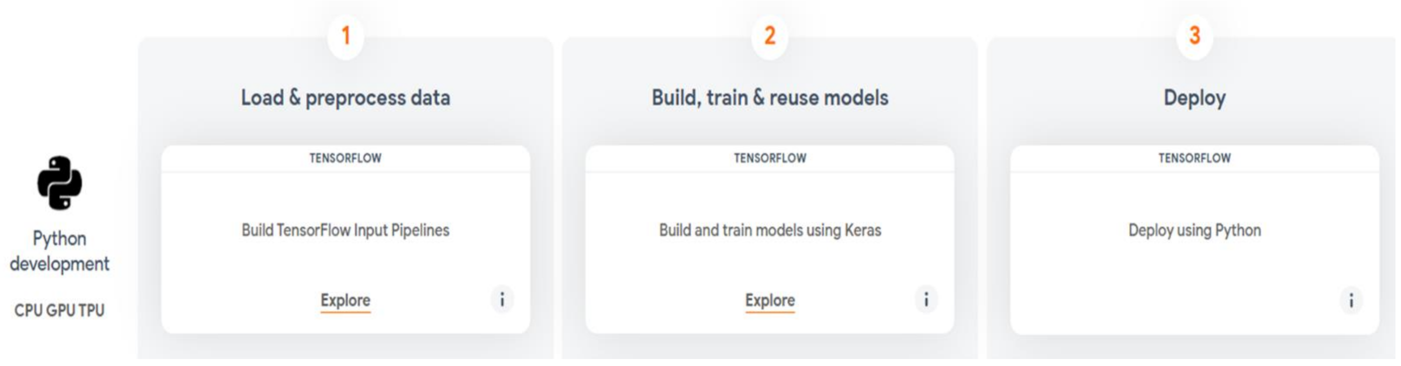
Covid-19 Results :



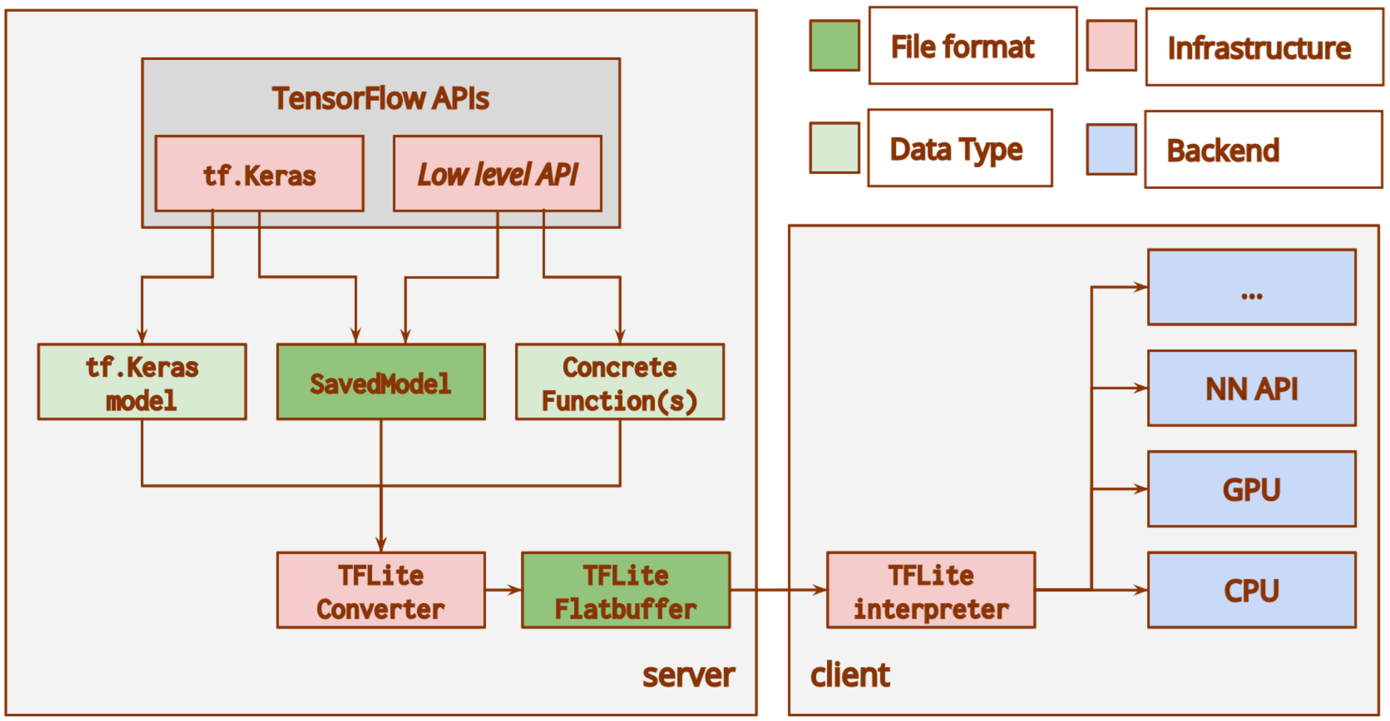


TensorFlow

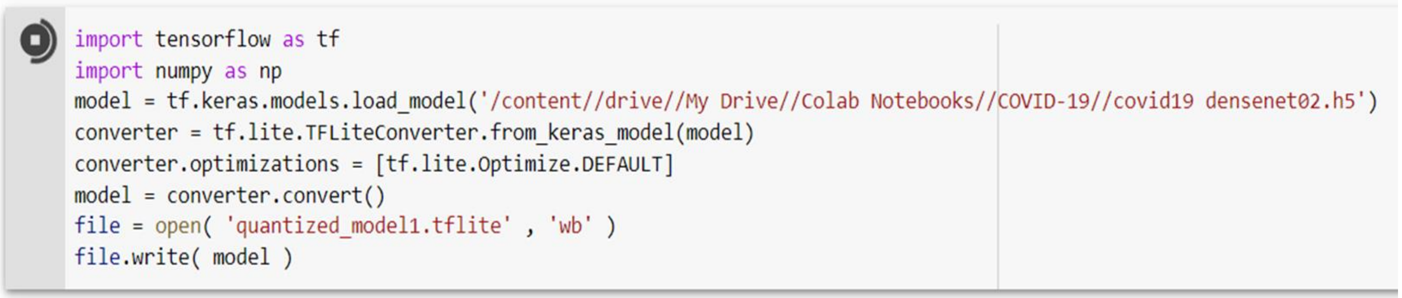
TensorFlow provides a collection of workflows to develop and train models using Python, JavaScript, or Swift, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.



TensorFlow Converter



Converting To QuantizedTFLite



```
import tensorflow as tf
import numpy as np
model = tf.keras.models.load_model('/content//drive//My Drive//Colab Notebooks//COVID-19//covid19 densenet02.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
model = converter.convert()
file = open( 'quantized_model1.tflite' , 'wb' )
file.write( model )
```

- *The advantages of float16 quantization are as follows:*

1. It reduces model size by up to half (since all weights become half of their original size).
2. It causes minimal loss in accuracy.
3. It supports some delegates (e.g. the GPU delegate) which can operate directly on float16 data, resulting in faster execution than float32 computations.

- *The disadvantages of float16 quantization are as follows:*

1. It does not reduce latency as much as a quantization to fixed point math.
2. By default, a float16 quantized model will "dequantize" the weights values to float32 when run on the CPU. (Note that the GPU delegate will not perform this dequantization, since it can operate on float16 data).

NVIDIA Jetson Nano Implementation

- ## What is NVIDIA Jetson Nano ?

  NVIDIA® Jetson Nano™ Developer Kit is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. All in an easy-to-use platform that runs in as little as 5 watts.

- ## Implementation

The first step to implement the model on NVIDIA Jetson is the installing of the following packages :

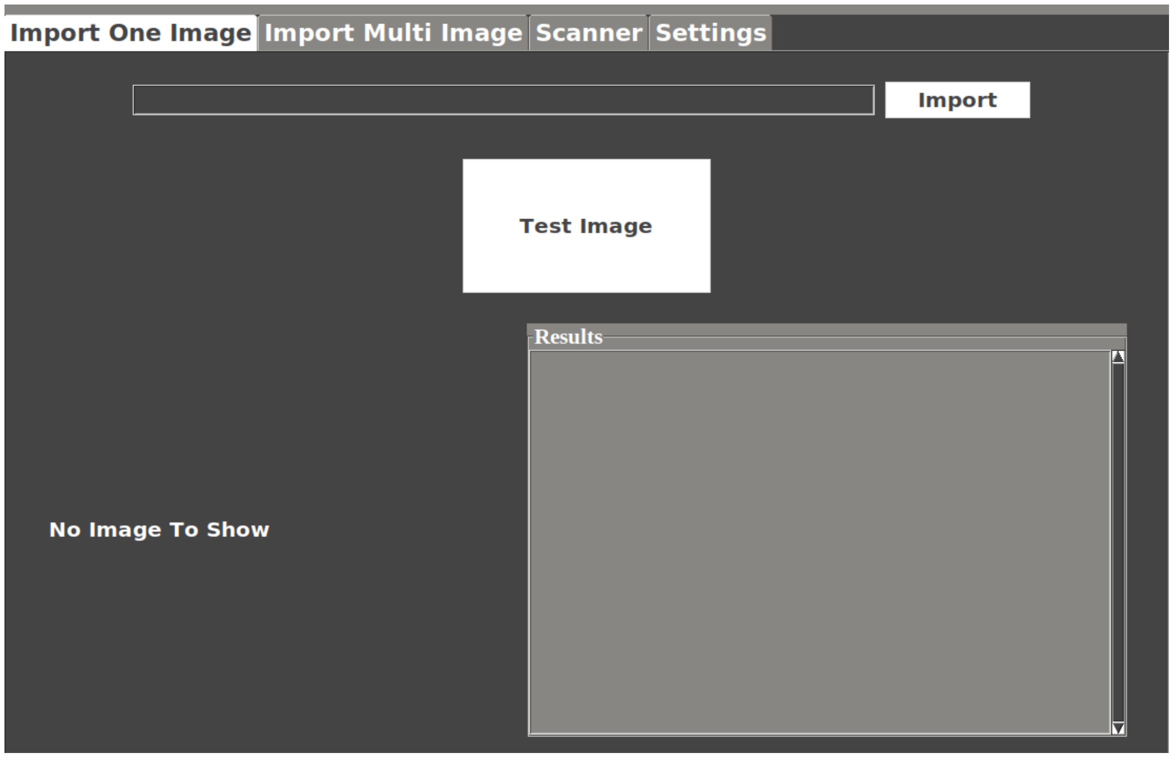
1. Python3.8
2. TensorFlow
3. Numpy
4. Tkinter
5. OpenCV

We used Tkinter to build the graphical user interface in python, it contain a menu with 3 options.

- Import One Image :

  You can import just one image and press Test Image to show the result. The result that will be see is :

  1- Gender of patient
  2- Covid-19 test result (Positive and Negative)
  3- Pneumonia test result (Positive and Negative)
  4- Time elapsed of testing



- ## Import Multi Images :

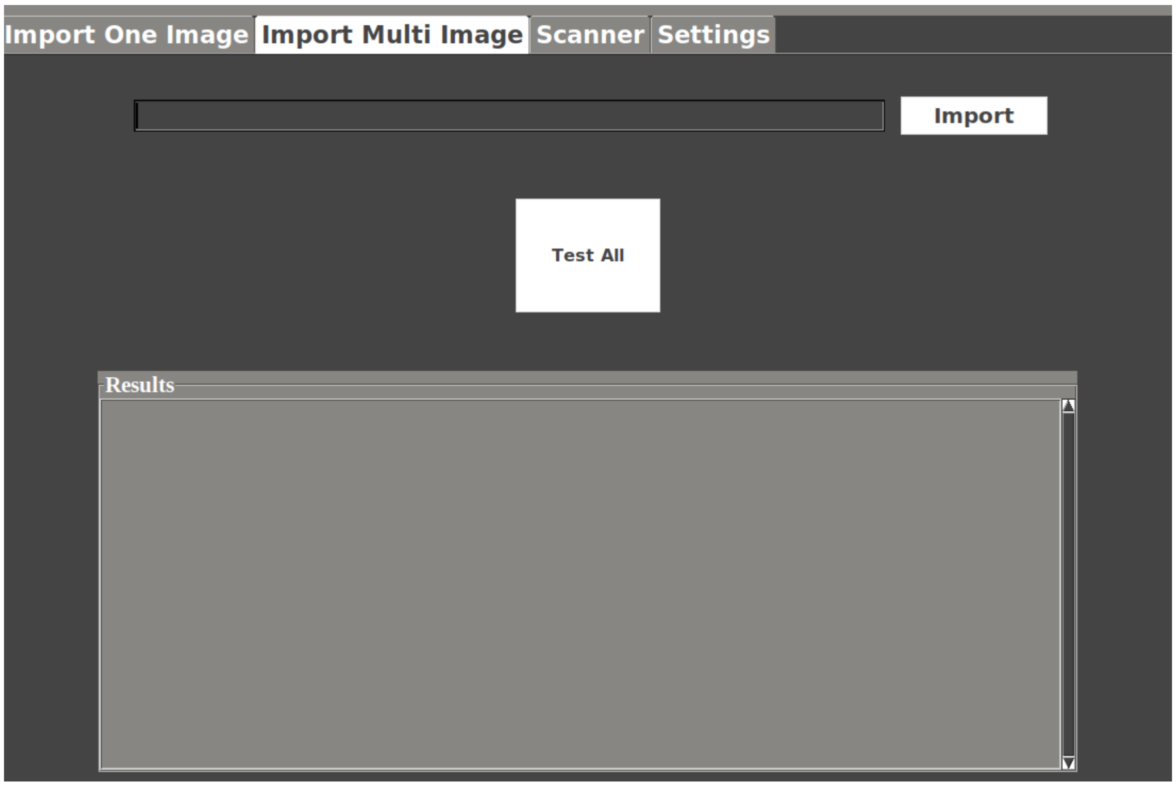
  Choose the directory that contain many images and press Test All to show the results of all images in the choosed directory. The result that will be see for each patient is :

  1- Gender of patient

  2- Covid-19 test result (Positive and Negative)

  3- Pneumonia test result (Positive and Negative)

  4- Time elapsed of testing

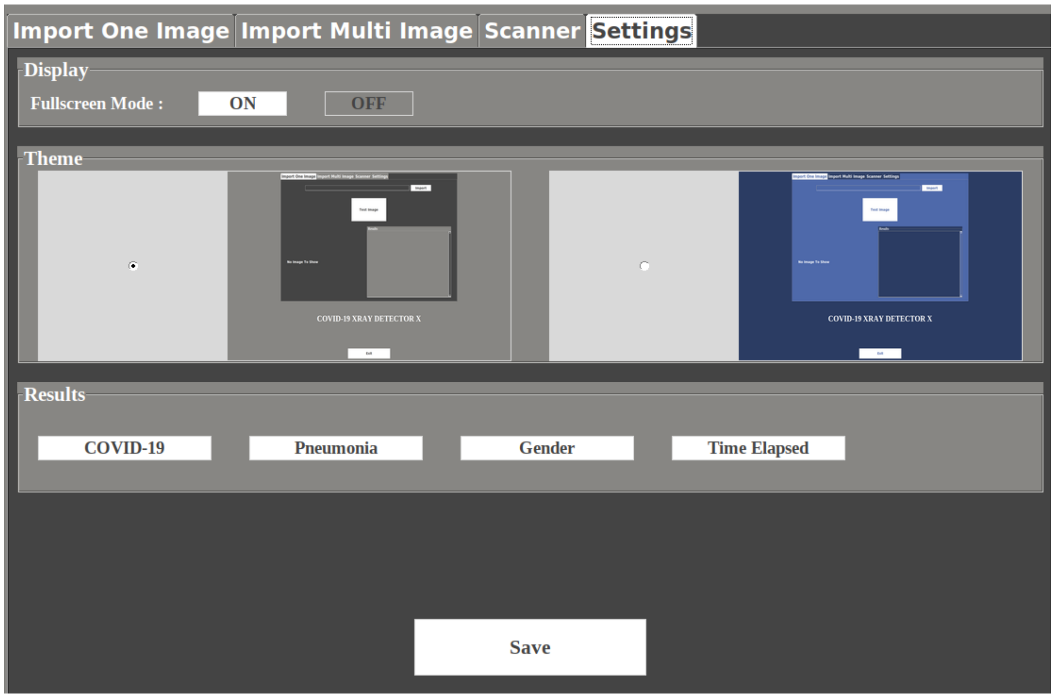


• Scanner :

Coming Soon

• Settings Tab :

Of course like all application we need a settings to manage our software, so we build a simple settings tab in this software contains some settings like Fullscreen mode and themes and results management, that can help user to manage what he want in the software.



Testing

After finish everything related to implementation, it’s time to test our program on some images (Not a trained images) to make sure that it’s results are accurate and fast in diagnosis and we can use it on real life. We've done multiple Tests to make sure that. We got the accuracy and the detection rate for each test.

- ### Test 1 :

  Give the software any single chest x-ray image (Positive or Negative). To test the COVID-19 model.

  Probability : **more than 98%**

  Time : **0.5 second**

- ### Test 2 :

  Give the software any single chest x-ray image (Positive or Negative). To test Pneumonia model.

  Probability : **more than 96%**

  Time : **0.4 second**

- ### Test 3 :

  Give the software 867 Positive images and 1311 Negative images. (Tested on COVID-19 model)

  Accuracy : **99.5%**

  Detection Rate : **98.9%**

  Time : **16.61 minutes**

- ## Test 4 :

  Give the software 1345 chest x-ray images of pneumonia but not a COVID-19. (Tested on COVID-19 model)

  Negative images : **1191 image**

  Positive images : **154 image**

  Accuracy : **99%**

  Detection Rate : **98%**

  Average Probability : **94%**

- ## Test 5 :

  Give the software 2000 chest x-ray images of pneumonia. (Tested on Pneumonia model)

  Accuracy : **98.4%**

  Detection Rate : **99.1%**

  Time : **7.6 minutes**

- ## Test 6 :

  We tested the gender model and the result come true 100% for any x-ray image.

Conclusion

- After working this project, we got accurate and impressive results. Of course we can not give up PCR testing completely and adopt this device as an alternative, but these results will enable us to reduce the number of these tests significantly, so that many countries adopt PCR tests for all people in cities, villages, hospitals and especially airports, and as we know that this examination is expensive and requires quarantine for about 72 hours and then this test must be re-examined again to confirm the result once and for all. But using this device we will be able to know the negative results more accurately through the x-ray image of the chest which can be obtained in only 10 minutes, the advantages of this device are that it does not give a negative result unless it is 100% sure , but if there is a possibility that the person is infected, the result will appear positive. This device deems a PCR unnecessary for a lot of people, as well as dividing people into two groups: First group consists of people who are 100% negative, and these people have no need to do a PCR test, second group consists of people who are possibly infected, with that possibility being 90% and above, and the capabilities of this device were tested on multiple x-ray pictures of people with infections or diseases in their respiratory organs but nothing COVID-19 related, and the results were 94% accurate. With these benefits, this device can be used in places like airports or hospitals, 2 places were a PCR tests result can be the difference between containing the virus, or failing to control it. And of course after seeing this results of testing on pneumonia model we can now approximately know if the patient have covid-19 or just a pneumonia.

References

1. https://www.pyimagesearch.com/2020/03/16/detecting-covid-19-in-x-ray-images-with-keras-tensorflow-and-deep-learning/

2. https://github.com/frogermcs/TFLite-Tester/tree/558d0dca39df476afbeb8dcd1689c174d6416439

3. https://medium.com/edureka/tensorflow-image-classification-19b63b7bfd95

4. https://medium.com/towards-artificial-intelligence/testing-tensorflow-lite-image-classification-model-e9c0100d8de3

5. https://thinkmobile.dev/automate-testing-of-tensorflow-lite-model-implementation/

6. https://blog.tensorflow.org/2018/03/using-tensorflow-lite-on-android.html

7. https://www.tensorflow.org

8. https://www.tensorflow.org/lite/

9. https://github.com/lindawangg/COVID-Net

10. https://www.tensorflow.org/lite/performance/post_training_quantization

11. https://github.com/arjunparmar/COVID-19

12. https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning

13. https://developer.nvidia.com/embedded/jetson-nano-developer-kit

14. https://www.kaggle.com/nih-chest-xrays/data

15. https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia