

Covid-19 Classification using CNNs

Data Science for Management – LM91



Università
di Catania



UNIVERSITY OF
LINCOLN



Google Cloud Platform

Introduction

AI-enabled Medical Image Analysis (MIA) Workshop is devoted to medical image analysis, with emphasis on radiological quantitative image analysis for diagnosis of diseases. The focus is on Artificial Intelligence (AI), Machine and Deep Learning (ML, DL) approaches that target effective and adaptive diagnosis; there is also a particular interest in approaches that enforce trustworthiness and automatically generate explanations, or justifications of the decision making process.

COV19D Competition is based on a database of chest CT scan series that is manually annotated with respect to Covid-19/non-Covid-19 diagnosis.

The aim of this Computer Vision project is to build a classifier using some models CNNs on PyTorch to classify ct scans of covid and non-covid patients.

The Workflow

The workflow

Here we describe the cloud workflow to upload to perform the classification



The workflow – Google Cloud Platform



New Project



You have 20 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *

MIA-COVD19-Challenge



Project ID: covid-19-analysis-318015. It cannot be changed later. [EDIT](#)

Location *



No organization

[BROWSE](#)

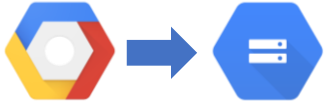
Parent organization or folder

[CREATE](#)

[CANCEL](#)

In Google Cloud create a new project naming it "MIA-COVD19-Challenge"

The workflow – Google Cloud Storage



← Create a bucket

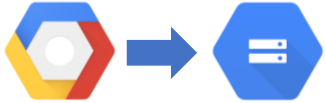
- **Name your bucket**
Pick a globally unique, permanent name. [Naming guidelines](#)

Tip: Don't include any sensitive information
CONTINUE
- Choose where to store your data
- Choose a default storage class for your data
- Choose how to control access to objects
- Advanced settings (optional)

CREATE CANCEL

1. In Google Cloud Storage create a new bucket and naming it 'covid_19_challenge'
2. Leave everything default and click on CREATE to create the bucket

The workflow – Google Cloud Storage



Bucket details

covid_19_challenge

OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE

Buckets > covid_19_challenge

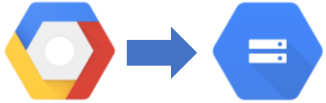
UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter Filter objects and folders

<input type="checkbox"/>	Name	Size	Type	Created time ?	Storage class
No rows to display					

1. In Google Cloud Storage create a new bucket and naming it 'covid_19_challenge'
2. Leave everything default and click on CREATE to create the bucket
3. Click on UPLOAD FILES and select the files of the dataset

The workflow – Google Cloud Storage



Bucket details

covid_19_challenge

OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE

Buckets > covid_19_challenge

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE


Filter by name prefix only Filter Filter objects and folders







<input type="checkbox"/>	Name	Size	Type	Created time ?
<input type="checkbox"/>	dati_palmucci_pred.zip	1.9 GB	application/x-zip-compressed	Jul 18, 2021, 12:18:45 PM
<input type="checkbox"/>	train.zip	32.4 GB	application/x-zip-compressed	Jul 10, 2021, 3:17:01 PM
<input type="checkbox"/>	val.zip	7 GB	application/x-zip-compressed	Jul 10, 2021, 10:13:25 AM

1. In Google Cloud Storage create a new bucket and naming it 'covid_19_challenge'
2. Leave everything default and click on CREATE to create the bucket
3. Click on UPLOAD FILES and select the files of the dataset
4. And wait until files are uploaded to the Bucket


The workflow – Google AI Platform (Notebooks)



 Notebooks + NEW INSTANCE ↻ REFRESH ▶ START ■ STOP ↺ RESET 🗑 DELETE

Create and use Jupyter Notebooks with a notebook instance. Notebook instances have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. [Learn more](#)

 **Filter** Enter property name or value

<input type="checkbox"/>	<input checked="" type="radio"/>	Instance name ↑	Zone	Environment Version	Auto-upgrade
--------------------------	----------------------------------	-----------------	------	---------------------	--------------

1. Create a NEW INSTANCE

The workflow – Google AI Platform (Notebooks)



Notebooks

[+ NEW INSTANCE](#) [REFRESH](#) [START](#) [STOP](#) [RESET](#) [DELETE](#)

Create and use JupyterLab pre-installed frameworks. [Learn more](#)

Filter Enter proper

- Python 3
Includes scikit-learn, pandas and more
- Python 3 (CUDA Toolkit 11.0)
Optimized for NVIDIA GPUs
- TensorFlow Enterprise
Includes Keras, scikit-learn, pandas, NLTK and more
- PyTorch 1.9**
Includes scikit-learn, pandas, NLTK and more
 - Auto-upgrade
 - Without GPUs
 - With 1 NVIDIA Tesla T4
- R 4.0
Includes basic R packages, scikit-learn, pandas, NLTK and more
- RAPIDS 0.18 [EXPERIMENTAL]
Optimized for NVIDIA GPUs
- Kaggle Python [BETA]
Python image for Kaggle Notebooks, supporting hundreds of machine learning libraries popular on Kaggle
- Theia IDE [EXPERIMENTAL]
IDE with notebook support including scikit-learn, pandas, and more
- Smart Analytics Frameworks
BigQuery, Apache Beam, Apache Spark, Apache Hive, and more

1. Create a NEW INSTANCE
2. Select PyTorch 1.9 With 1 NVIDIA Tesla T4

The workflow – Google AI Platform (Notebooks)



New notebook instance

Instance name
pytorch-1-9-20210713-203251

63-char limit with lowercase letters, digits, or '-' only. Must start with a letter. Cannot end with a '-'.
[?](#)

Region *
asia-northeast1 (Tokyo) [?](#)

Zone *
asia-northeast1-a [?](#)

[?](#) Restricted by GPU selection.

Instance properties [✎](#)

Environment ?	PyTorch 1.9 (with Intel® MKL-DNN/MKL)
Machine type	4 vCPUs, 15 GB RAM
GPUs ?	1 NVIDIA Tesla T4
Boot disk	100 GB Standard persistent disk
Data disk	100 GB Standard persistent disk
Subnetwork	default(10.146.0.0/20) ▼
External IP	Ephemeral(Automatic)
Extensions ?	SELECT EXTENSIONS None selected
Permission	Compute Engine default service account
Estimated cost ?	\$321.03 monthly, \$0.440 hourly

☒ Install NVIDIA GPU driver automatically for me [?](#)

ADVANCED OPTIONS

CANCEL

CREATE

1. Create a NEW INSTANCE
2. Select PyTorch 1.9 With 1 NVIDIA Tesla T4
3. Find a Region and Zone with the instance properties required
4. Click on CREATE

The workflow – Google AI Platform (Notebooks)



Notebooks [+ NEW INSTANCE](#) [REFRESH](#) [▶ START](#) [■ STOP](#) [⏻ RESET](#) [🗑 DELETE](#)

Create and use Jupyter Notebooks with a notebook instance. Notebook instances have JupyterLab pre-installed and are configured with GPU-enabled machine learning frameworks. [Learn more](#)

[Filter](#) Enter property name or value

<input type="checkbox"/>	<input checked="" type="radio"/>	Instance name ↑	Zone	Environment Version	Auto-upgrade
<input type="checkbox"/>	<input checked="" type="radio"/>	pytorch-1-9-20210714-154656 OPEN JUPYTERLAB	asia-east1-a	M74	—

1. Create a NEW INSTANCE
2. Select PyTorch 1.9 With 1 NVIDIA Tesla T4
3. Find a Region and Zone with the instance properties required
4. Click on CREATE
5. When the VM is ready on click on OPEN JUPYTERLAB

The dataset

The analysis of the dataset

The dataset consists of ct scans of 1926 patients, split in:

- **Training set**

- 687 Covid patients with 153.680 ct scan slices

- 865 Non-covid patients with 181.991 ct scan slices

- **Validation set**

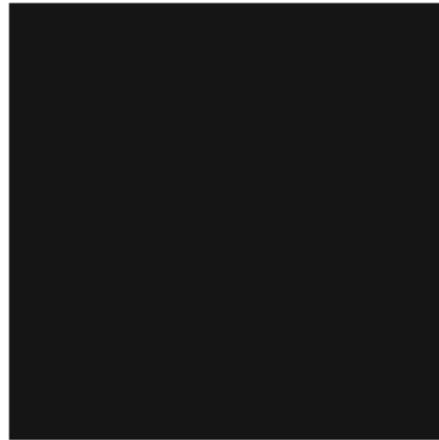
- 165 Covid patients with 35.002 ct scan slices

- 209 Non-covid patients with 40.516 ct scan slices

Please note: Test set will be created using the 10% of the training set

The analysis of the dataset

After the analysis we found that 14 slices in ct_scan_34 in the validation set (covid) and 1 slice in ct_scan_263 in the training set (covid) are black:



Also, there are two files “._9.jpg” and “._92.jpg” in ct_scan_9 in the validation set (non covid) that create problems during the training process.

The analysis of the dataset

At the end we have:

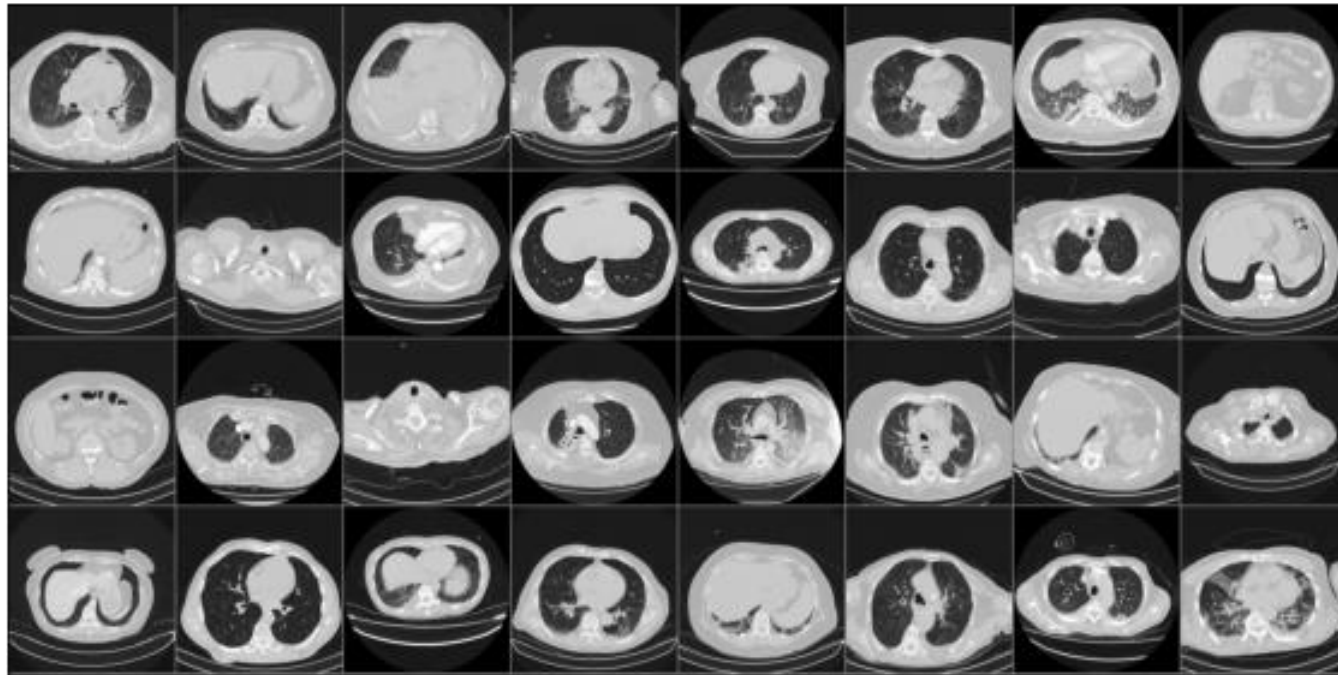
	Set	Class	Number of Patients	Number of slices	Average number of slices
0	train	covid	687	153680	224
1	train	non-covid	865	181991	210
2	val	covid	165	35002	212
3	val	non-covid	209	40516	194

And we decided to create the test set as the 10% of the training set, and we obtain:

- Number of training samples: 302103
- Number of validation samples: 75518
- Number of test samples: 33568

The analysis of the dataset

Image sample (after the preprocessing and loaded in the batch)



Classification

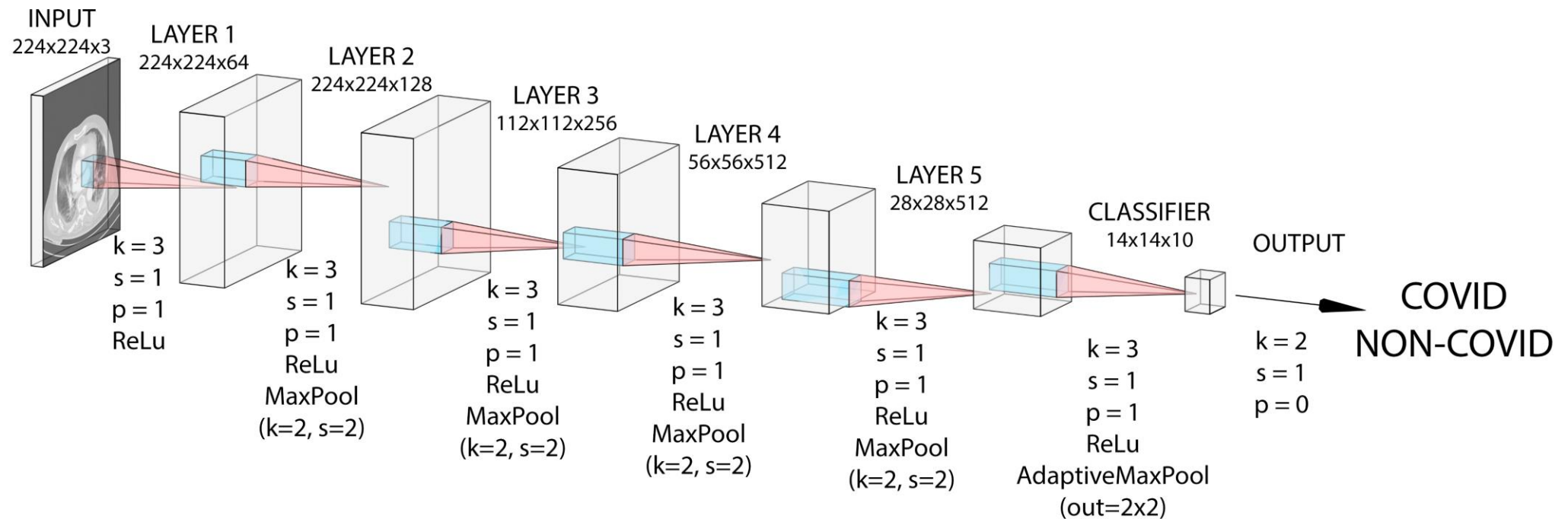
Settings

The dataset is transformed and loaded with:

- Resize of 224x224
- Normalization with mean and std of 0.5
- Batch size of 32 (only for FCN, then 64)
- Cross Entropy for the loss
- Adam with learning rate of 0.0001
- 5 Epochs for training
- Comparison of FCN vs VGG19, ResNet-152 and DenseNet-161 (Pre-trained)

FCN Architecture

- 5 layer, with kernel size 3x3, padding and striding 1.
- The number of channels is doubled at each layer, starting from 64 to 512.
- The activation function used is ReLu with Max pooling of size 2x2 and striding 2.
- Due to the lack of final dense layer as classifier we used a convolutional layer of 512, kernel size 2x2, padding 0 and striding 1, that will be able to classify images in the 2 predefined classes.



FCN Architecture – Results

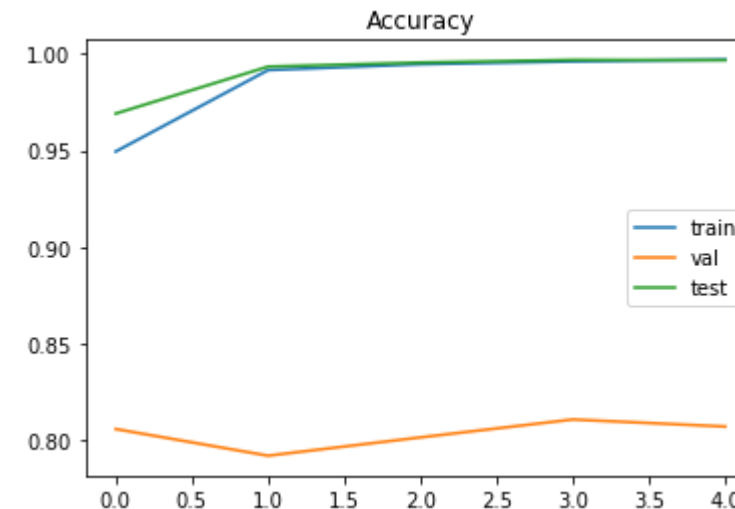
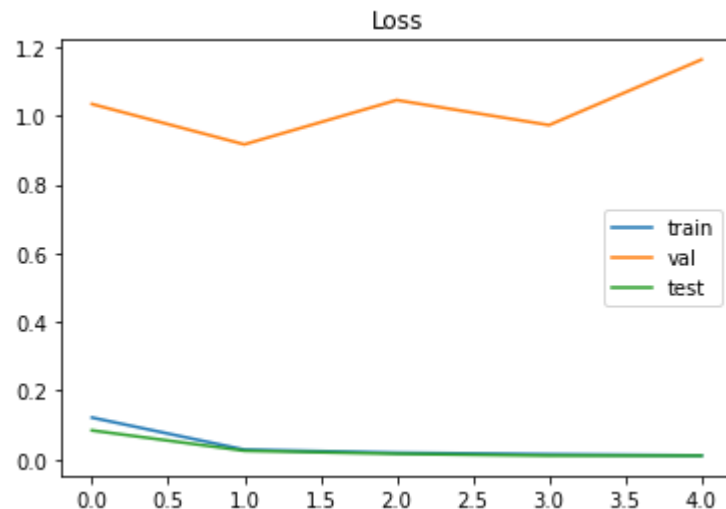
Epoch 1: Training Loss=0.1219, Training Accuracy=0.9495, Validation Loss=1.0336, Validation Accuracy=0.8058, Test Loss=0.0846, Test Accuracy=0.9691

Epoch 2: Training Loss=0.0286, Training Accuracy=0.9916, Validation Loss=0.9163, Validation Accuracy=0.7920, Test Loss=0.0254, Test Accuracy=0.9934

Epoch 3: Training Loss=0.0196, Training Accuracy=0.9947, Validation Loss=1.0451, Validation Accuracy=0.8015, Test Loss=0.0164, Test Accuracy=0.9955

Epoch 4: Training Loss=0.0152, Training Accuracy=0.9961, Validation Loss=0.9725, Validation Accuracy=0.8108, Test Loss=0.0113, Test Accuracy=0.9969

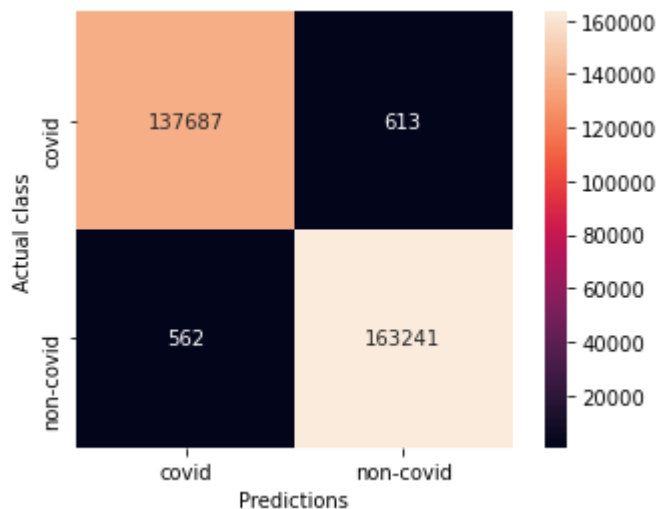
Epoch 5: Training Loss=0.0117, Training Accuracy=0.9971, Validation Loss=1.1629, Validation Accuracy=0.8071, Test Loss=0.0111, Test Accuracy=0.9968



FCN Architecture – Results (Best model)

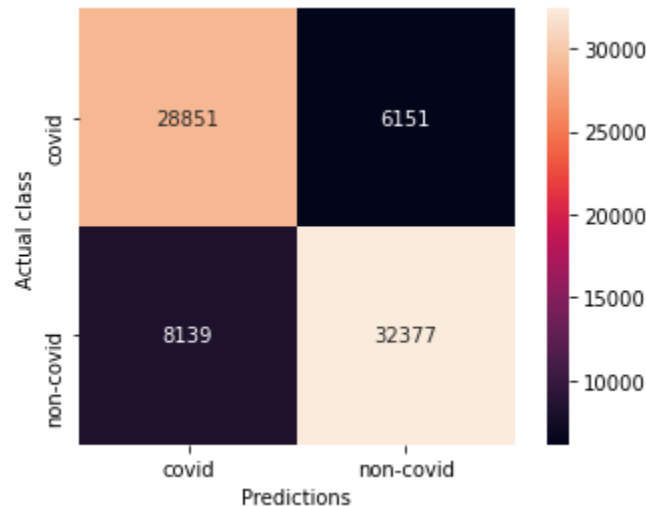
Epoch 4: Training Loss=0.0152, Training Accuracy=0.9961, Validation Loss=0.9725, Validation Accuracy=0.8108, Test Loss=0.0113, Test Accuracy=0.9969

Confusion Matrix train set



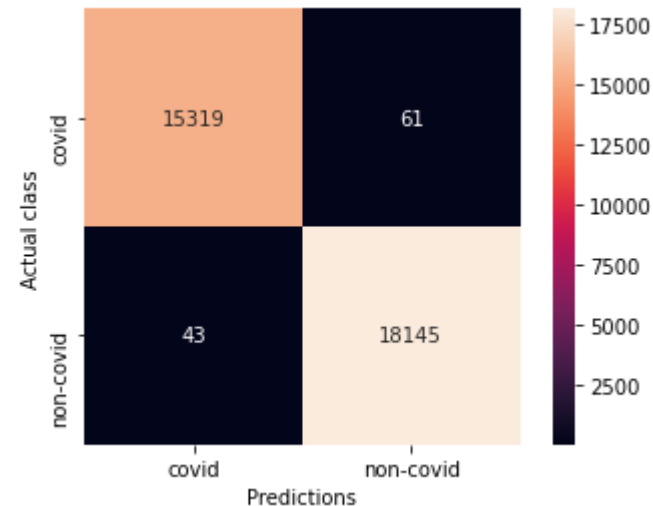
	precision	recall	f1-score	support
covid	1.00	1.00	1.00	138300
non-covid	1.00	1.00	1.00	163803
accuracy			1.00	302103
macro avg	1.00	1.00	1.00	302103
weighted avg	1.00	1.00	1.00	302103

Confusion Matrix val set



	precision	recall	f1-score	support
covid	0.78	0.82	0.80	35002
non-covid	0.84	0.80	0.82	40516
accuracy			0.81	75518
macro avg	0.81	0.81	0.81	75518
weighted avg	0.81	0.81	0.81	75518

Confusion Matrix test set

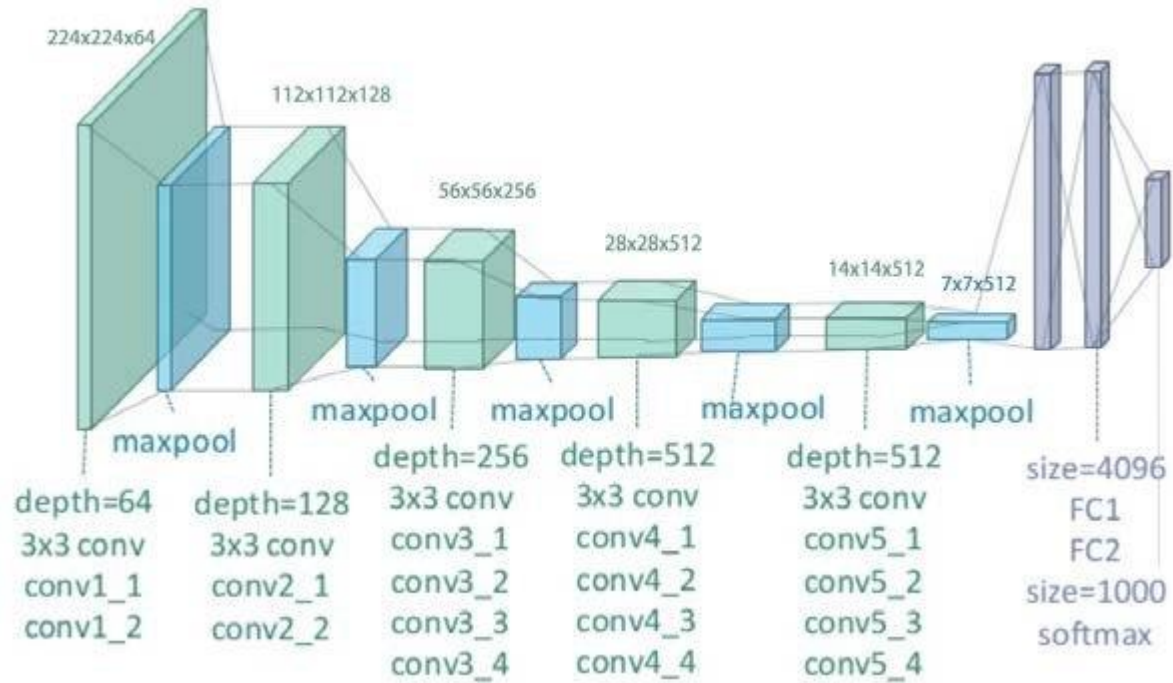


	precision	recall	f1-score	support
covid	1.00	1.00	1.00	15380
non-covid	1.00	1.00	1.00	18188
accuracy			1.00	33568
macro avg	1.00	1.00	1.00	33568
weighted avg	1.00	1.00	1.00	33568

100

VGG stands for Visual Geometry Group from Oxford. It is composed by 19 convolutional layers with a 3 x 3 Kernel, max pool kernel of size 2x2 size with stride 2, and 3 fully connected layers used for classification.

It has a total of about 143 million parameters.



VGG19 Architecture – Results

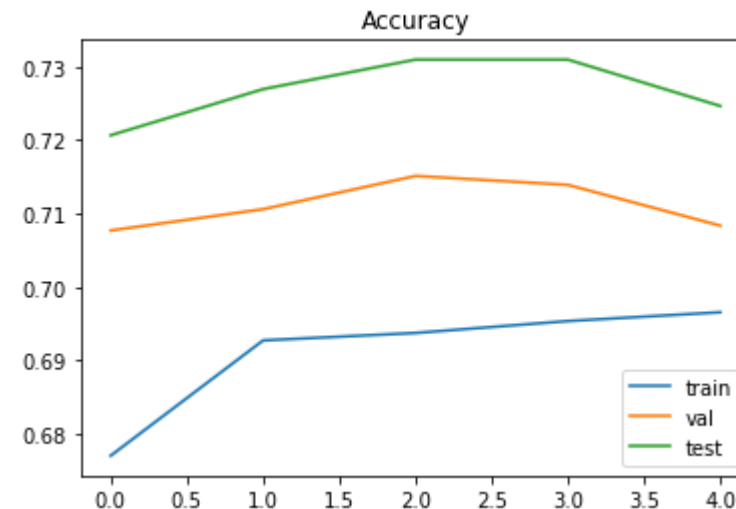
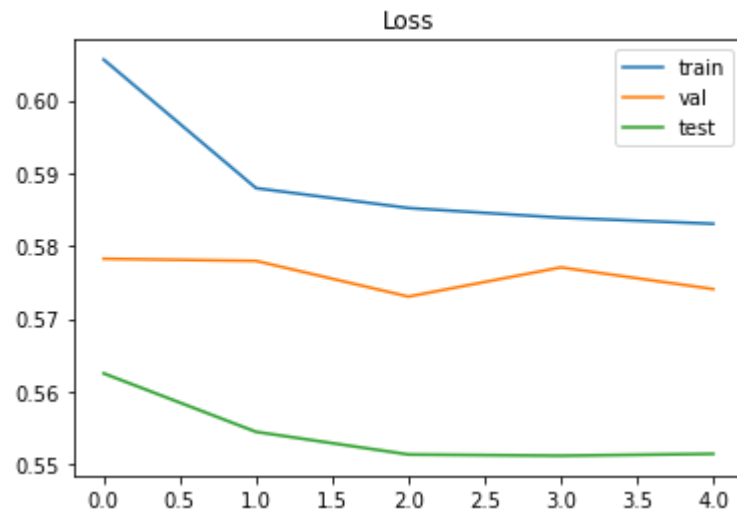
Epoch 1: Training Loss=0.6056, Training Accuracy=0.6771, Validation Loss=0.5782, Validation Accuracy=0.7077, Test Loss=0.5625, Test Accuracy=0.7207

Epoch 2: Training Loss=0.5880, Training Accuracy=0.6927, Validation Loss=0.5779, Validation Accuracy=0.7106, Test Loss=0.5545, Test Accuracy=0.7269

Epoch 3: Training Loss=0.5852, Training Accuracy=0.6938, Validation Loss=0.5730, Validation Accuracy=0.7151, Test Loss=0.5513, Test Accuracy=0.7310

Epoch 4: Training Loss=0.5839, Training Accuracy=0.6954, Validation Loss=0.5771, Validation Accuracy=0.7139, Test Loss=0.5512, Test Accuracy=0.7310

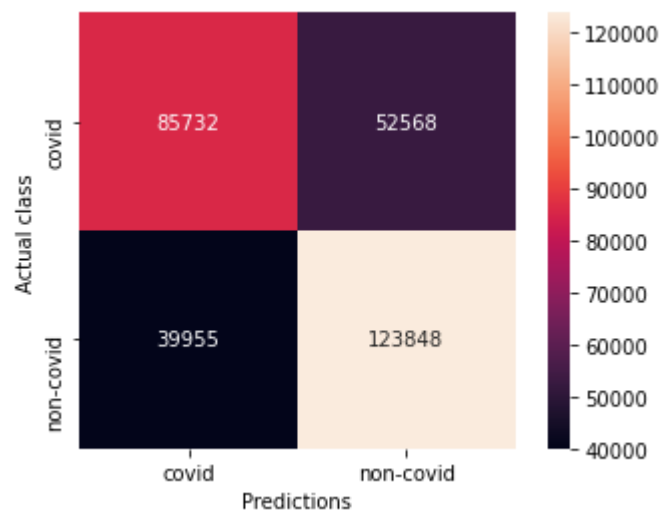
Epoch 5: Training Loss=0.5831, Training Accuracy=0.6966, Validation Loss=0.5741, Validation Accuracy=0.7084, Test Loss=0.5514, Test Accuracy=0.7246



VGG19 Architecture – Results (Best model)

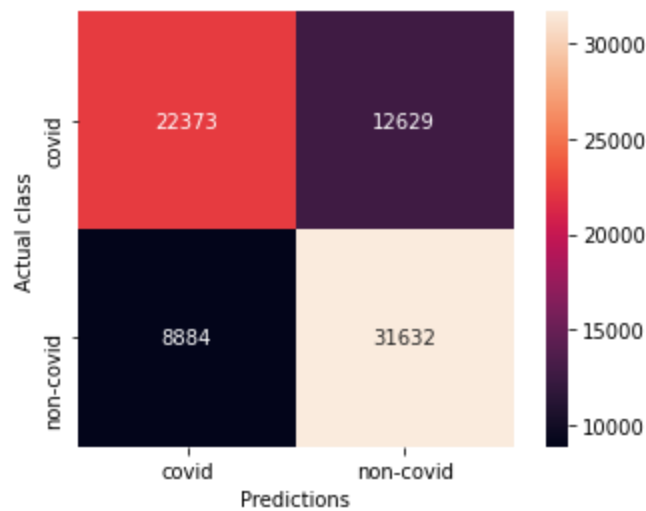
Epoch 3: Training Loss=0.5852, Training Accuracy=0.6938, Validation Loss=0.5730, Validation Accuracy=0.7151, Test Loss=0.5513, Test Accuracy=0.7310

Confusion Matrix train set



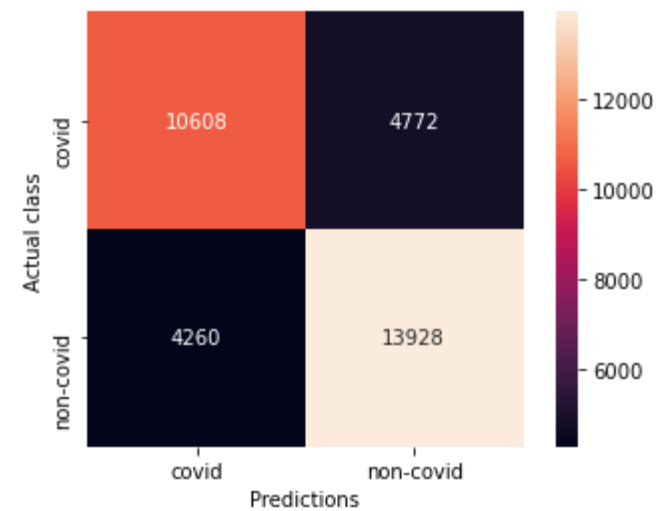
	precision	recall	f1-score	support
covid	0.68	0.62	0.65	138300
non-covid	0.70	0.76	0.73	163803
accuracy			0.69	302103
macro avg	0.69	0.69	0.69	302103
weighted avg	0.69	0.69	0.69	302103

Confusion Matrix val set



	precision	recall	f1-score	support
covid	0.72	0.64	0.68	35002
non-covid	0.71	0.78	0.75	40516
accuracy			0.72	75518
macro avg	0.72	0.71	0.71	75518
weighted avg	0.72	0.72	0.71	75518

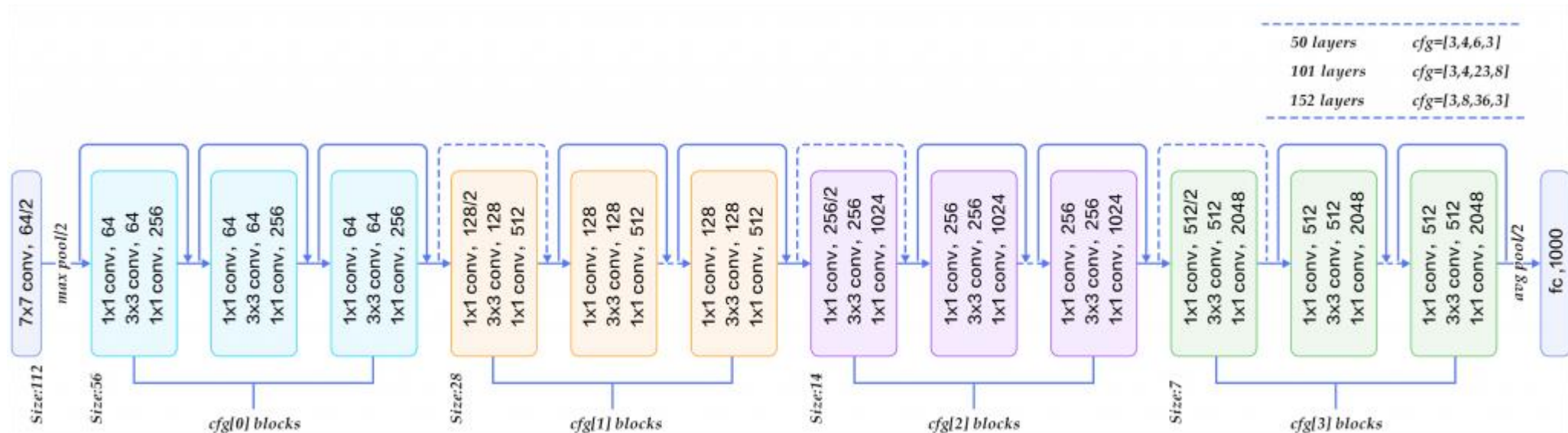
Confusion Matrix test set



	precision	recall	f1-score	support
covid	0.71	0.69	0.70	15380
non-covid	0.74	0.77	0.76	18188
accuracy			0.73	33568
macro avg	0.73	0.73	0.73	33568
weighted avg	0.73	0.73	0.73	33568

ResNet-152 Architecture

To avoid the problem of vanishing/exploding gradients ResNet uses the so Residual Block which is a residual mapping layer copied from the shallower layer. All convolutional layers apply the same convolutional window of size 3×3 , the number of filters increases following the depth of networks, from 64 to 2048. There is only one max-pooling layer with pooling size 3×3 , and a stride of 2 is applied after the first layer. The average pooling layer is applied to replace fully connected layers at the end of the architecture. It has 60 million of parameters.



ResNet-152 Architecture – Results

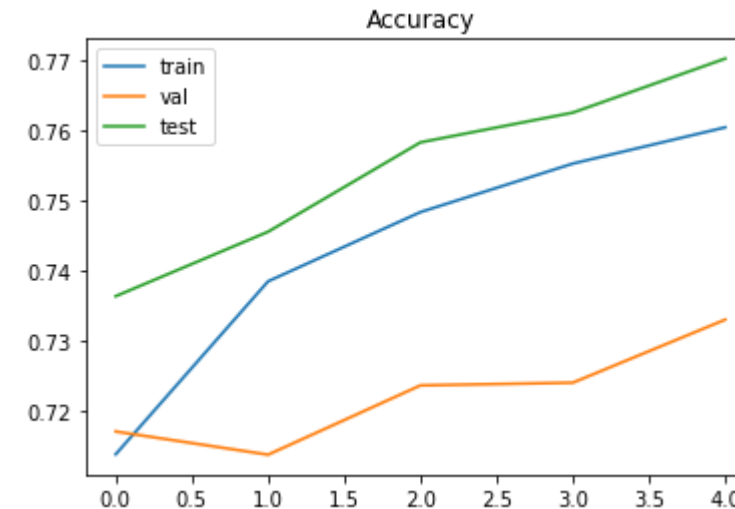
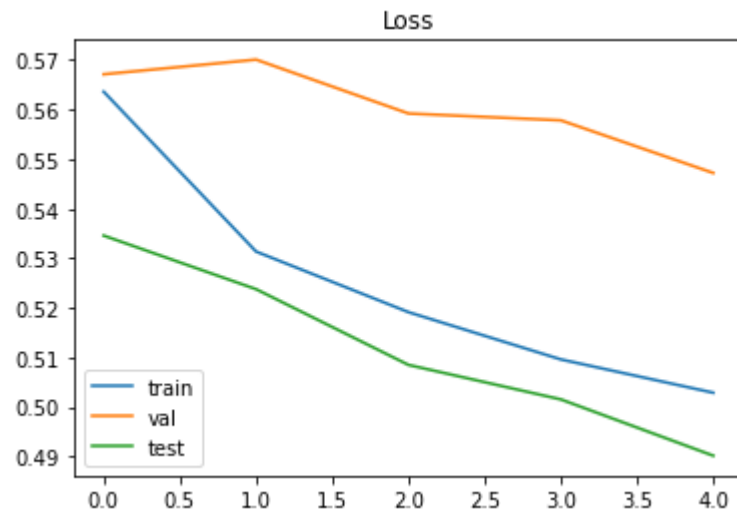
Epoch 1: Training Loss=0.5636, Training Accuracy=0.7139, Validation Loss=0.5671, Validation Accuracy=0.7171, Test Loss=0.5346, Test Accuracy=0.7363

Epoch 2: Training Loss=0.5314, Training Accuracy=0.7385, Validation Loss=0.5701, Validation Accuracy=0.7138, Test Loss=0.5238, Test Accuracy=0.7455

Epoch 3: Training Loss=0.5191, Training Accuracy=0.7483, Validation Loss=0.5592, Validation Accuracy=0.7237, Test Loss=0.5085, Test Accuracy=0.7582

Epoch 4: Training Loss=0.5096, Training Accuracy=0.7552, Validation Loss=0.5578, Validation Accuracy=0.7240, Test Loss=0.5015, Test Accuracy=0.7624

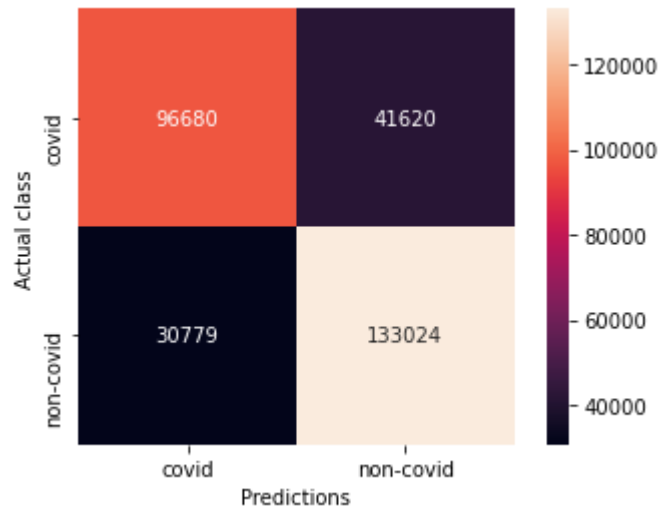
Epoch 5: Training Loss=0.5029, Training Accuracy=0.7603, Validation Loss=0.5472, Validation Accuracy=0.7330, Test Loss=0.4902, Test Accuracy=0.7701



ResNet-152 Architecture – Results (Best model)

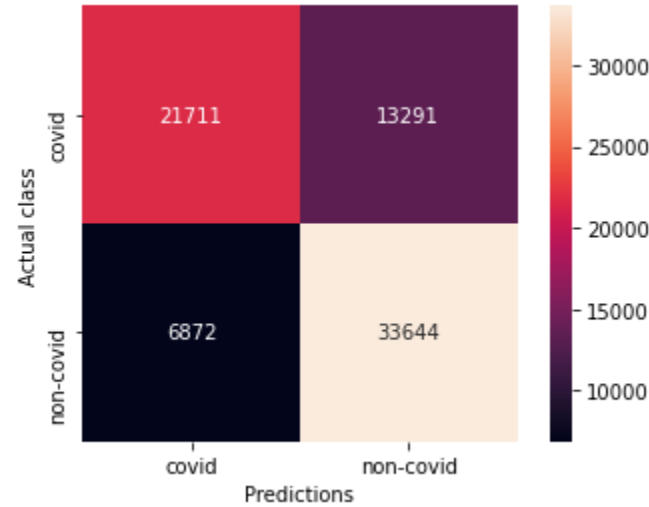
Epoch 5: Training Loss=0.5029, Training Accuracy=0.7603, Validation Loss=0.5472, Validation Accuracy=0.7330, Test Loss=0.4902, Test Accuracy=0.7701

Confusion Matrix train set



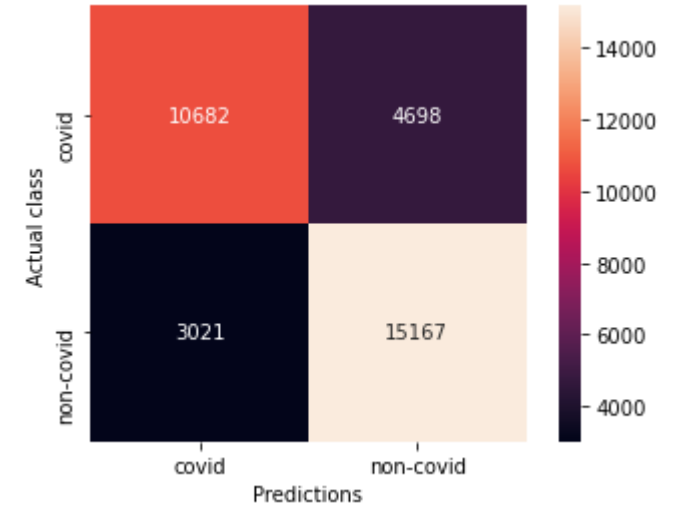
	precision	recall	f1-score	support
covid	0.76	0.70	0.73	138300
non-covid	0.76	0.81	0.79	163803
accuracy			0.76	302103
macro avg	0.76	0.76	0.76	302103
weighted avg	0.76	0.76	0.76	302103

Confusion Matrix val set



	precision	recall	f1-score	support
covid	0.76	0.62	0.68	35002
non-covid	0.72	0.83	0.77	40516
accuracy			0.73	75518
macro avg	0.74	0.73	0.73	75518
weighted avg	0.74	0.73	0.73	75518

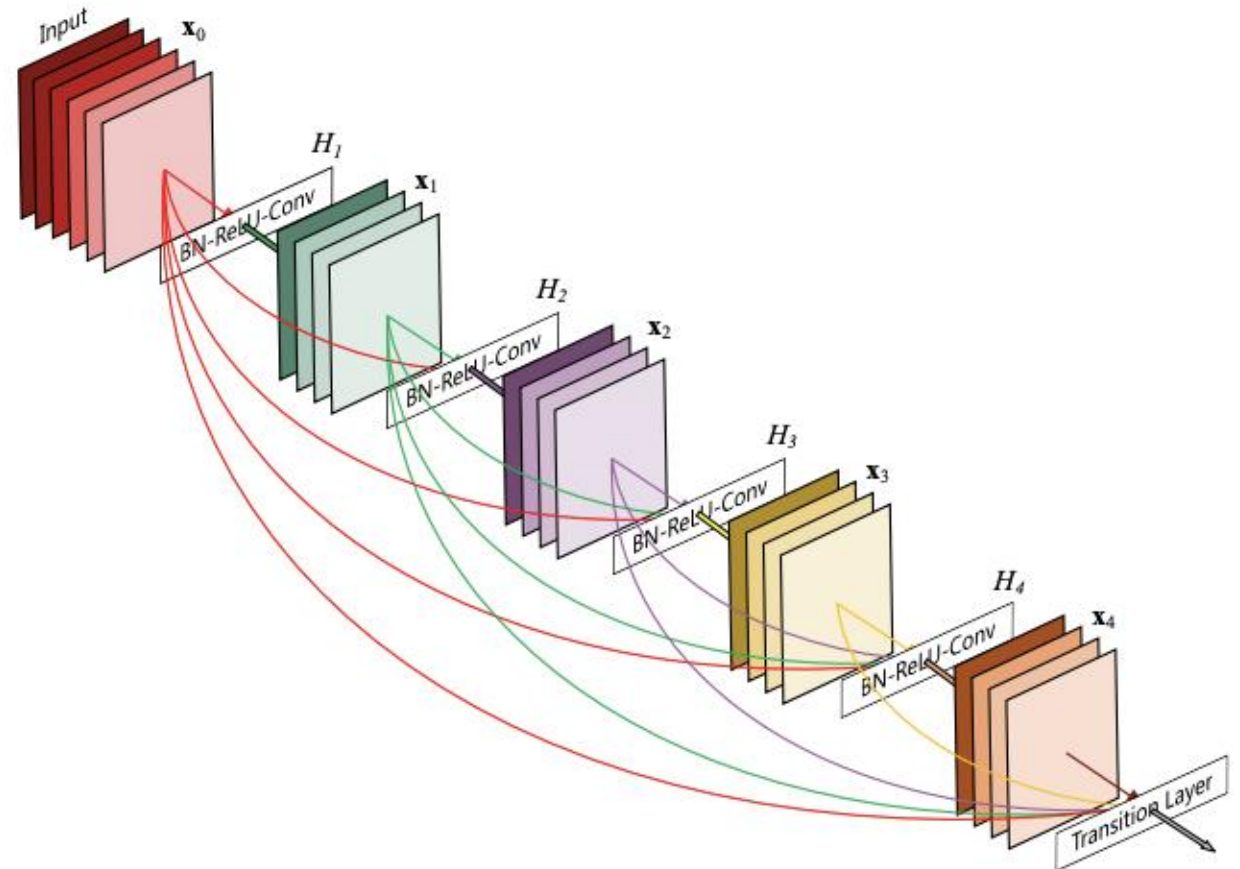
Confusion Matrix test set



	precision	recall	f1-score	support
covid	0.78	0.69	0.73	15380
non-covid	0.76	0.83	0.80	18188
accuracy			0.77	33568
macro avg	0.77	0.76	0.77	33568
weighted avg	0.77	0.77	0.77	33568

DenseNet-161 Architecture

Quite differently from ResNet, in DenseNet for each layer the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. This alleviates the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. It is composed by a 7x7 convolutional layer with stride 2, then max pool 3x3 with stride 2, and then a series of dense blocks and transition layers. The classification layer is created by a 7x7 global average pool with a 1000D fully connected. It has 29 million of parameters.



DenseNet-161 Architecture – Results

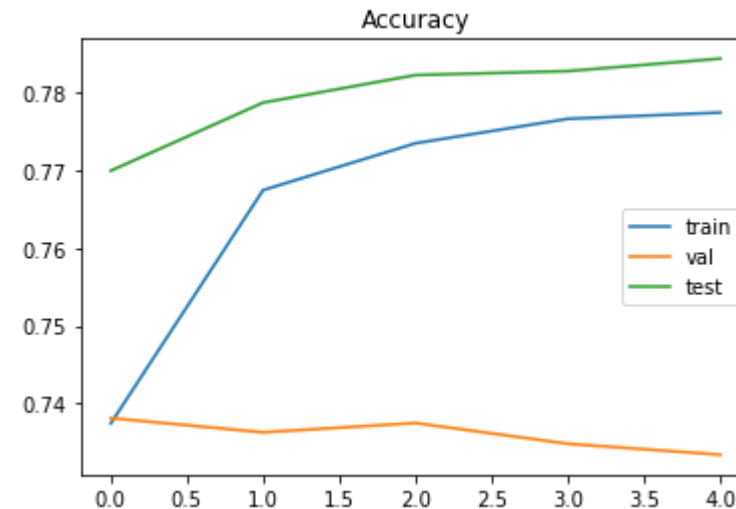
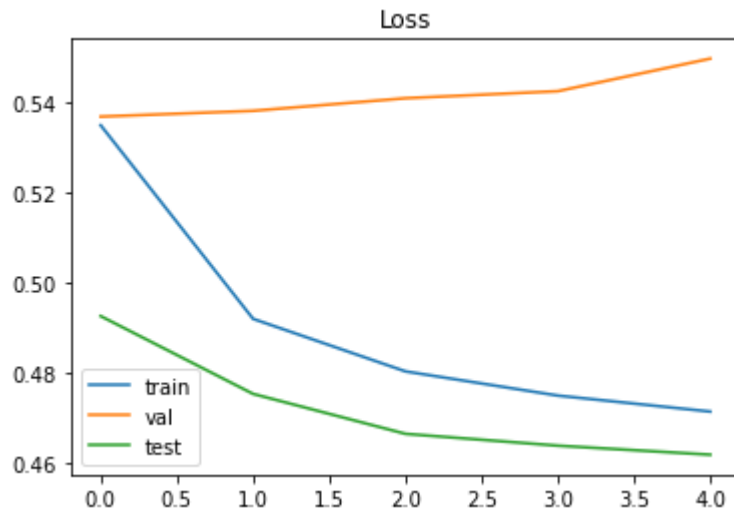
Epoch 1: Training Loss=0.5348, Training Accuracy=0.7375, Validation Loss=0.5367, Validation Accuracy=0.7381, Test Loss=0.4925, Test Accuracy=0.7699

Epoch 2: Training Loss=0.4919, Training Accuracy=0.7674, Validation Loss=0.5380, Validation Accuracy=0.7363, Test Loss=0.4753, Test Accuracy=0.7787

Epoch 3: Training Loss=0.4802, Training Accuracy=0.7734, Validation Loss=0.5408, Validation Accuracy=0.7375, Test Loss=0.4664, Test Accuracy=0.7822

Epoch 4: Training Loss=0.4749, Training Accuracy=0.7766, Validation Loss=0.5423, Validation Accuracy=0.7348, Test Loss=0.4638, Test Accuracy=0.7827

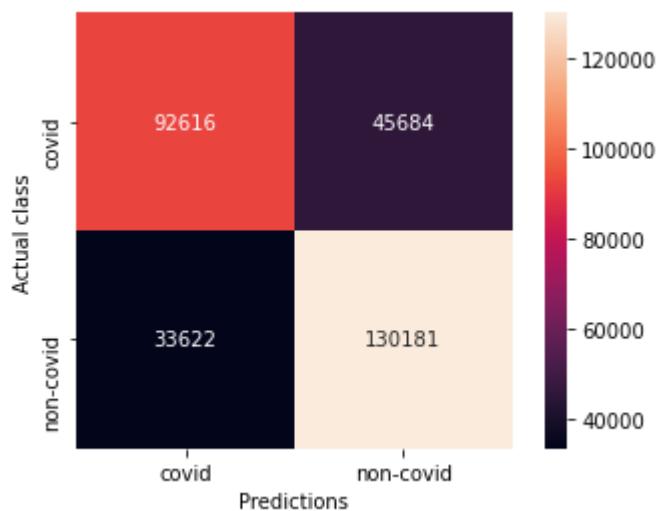
Epoch 5: Training Loss=0.4713, Training Accuracy=0.7774, Validation Loss=0.5495, Validation Accuracy=0.7334, Test Loss=0.4618, Test Accuracy=0.7843



DenseNet-161 Architecture – Results (Best model)

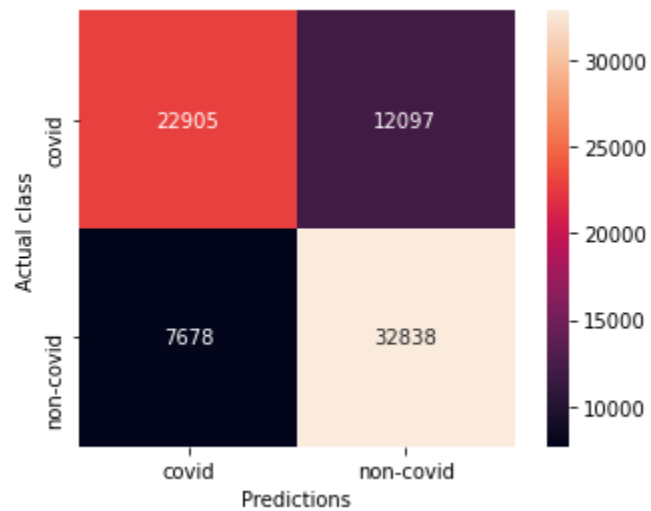
Epoch 1: Training Loss=0.5348, Training Accuracy=0.7375, Validation Loss=0.5367, Validation Accuracy=0.7381, Test Loss=0.4925, Test Accuracy=0.7699

Confusion Matrix train set



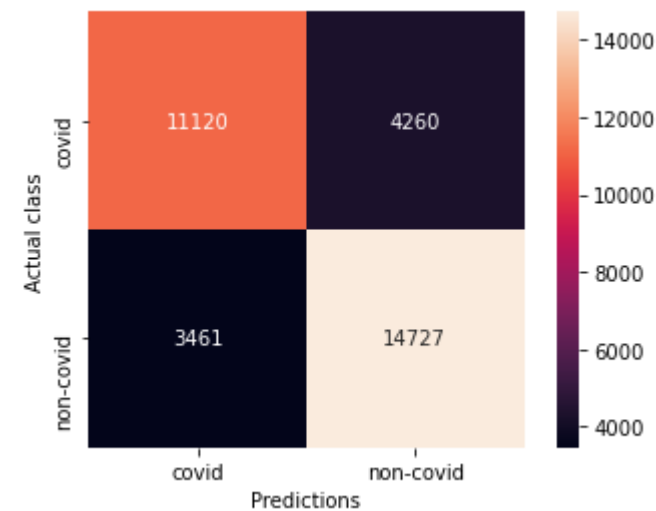
	precision	recall	f1-score	support
covid	0.73	0.67	0.70	138300
non-covid	0.73	0.79	0.77	163803
accuracy			0.74	302103
macro avg	0.74	0.73	0.73	302103
weighted avg	0.74	0.74	0.74	302103

Confusion Matrix val set



	precision	recall	f1-score	support
covid	0.75	0.65	0.70	35002
non-covid	0.73	0.81	0.77	40516
accuracy			0.74	75518
macro avg	0.74	0.73	0.73	75518
weighted avg	0.74	0.74	0.74	75518

Confusion Matrix test set



	precision	recall	f1-score	support
covid	0.76	0.72	0.74	15380
non-covid	0.78	0.81	0.79	18188
accuracy			0.77	33568
macro avg	0.77	0.77	0.77	33568
weighted avg	0.77	0.77	0.77	33568

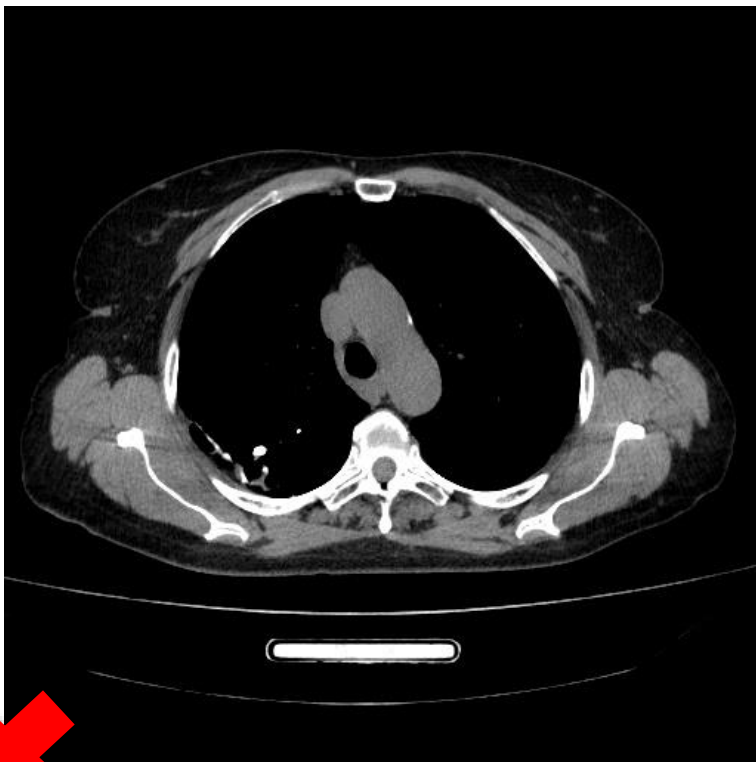
Comparison between architectures

Model	Epoch	Train Accuracy	Val Accuracy	Test Accuracy
My model FCN	4	0.996111	0.810769	0.996902
DenseNet-161	1	0.737452	0.738148	0.769911
ResNet-152	5	0.760347	0.733011	0.770119
VGG19	3	0.693755	0.715129	0.730952

Inference

The analysis of the Inference dataset

Data inconsistencies (same patient but different ct scans)



ct_scan_2 slice 100.jpg



ct_scan_3 slice 100.jpg

Inference on the Inference set

Prediction set consists in:

- Number of patients: 49
- Total number of slices: 14635
- Average number of slices per patient: 299

Patient: "ct_scan_21"

Number of Covid slices: 63/166

Number of Non-Covid slices: 103/166

Slice: 3.jpg - covid

