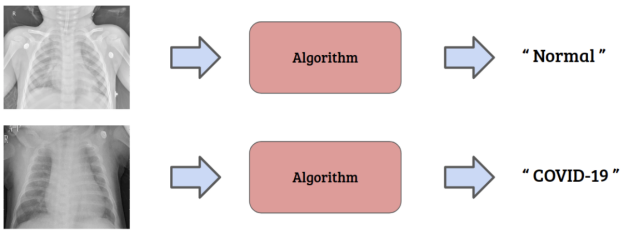
Advanced Algorithms project Report

Momesso Filippo
Università degli Studi di Trento
filippo.momesso@studenti.unitn.it

1. Introduction

In this project the goal is to build a machine learning classifier that is able to recognize presence of illness in a patient by analyzing thoracic x-ray images. The algorithm has to classify the images between four classes:

- **Normal** - the patient is healthy
- **Bacterial** - the patient has a bacterial infection
- **Viral** - the patient has a viral infection
- **COVID-19** - the patient is infected by COVID-19



The data is composed by ˜6000 thoracic images split into *training*, *validation* and *test* sets. Training and validation sets are provided with ground-truth labels, making this problem a task of supervised learning. In addition to the images, the same dataset is also provided as numpy arrays of already extracted features with corresponding labels for training and validation sets.

The table below summarizes how the samples are distributed through the classes and evidences an important characteristic of the dataset: it is highly imbalanced. Samples of class *COVID-19* are few compared to other classes' samples.

Split/Class	Normal	Bacterial	Viral	COVID-19	Total
train	951	1664	909	45	3569
validation	313	553	303	20	1189
test	319	569	292	11	1191
total	1583	2786	1504	76	5949

Table 1. Distribution of samples.

To measure the performances of the diffents algorithms used to solve the problem, I used the following classification metrics:

- Global Accuracy (Acc): $\frac{\text{\# correctly class. img}}{\text{\# img}}$
- Class Averaged Accuracy (mAcc): $\frac{1}{|Y|}\sum_{y\in Y}\frac{\text{\# correctly class. img of class } y}{\text{\# img of class } y}$
- Class averaged Intersection-over-Union (mIoU): $\frac{1}{|Y|}\sum_{y\in Y}\frac{\text{\# correctly class. img of class } y}{\text{\# img of class } y \text{ or predicted of class } y}$

2. Proposed Method

To solve the problem I tried with three different approaches: *random forests*, *support vector machines* and *convolutional neural networks*. Due to the fact that the dataset presented imbalanced data, at training time I used class weights to compensate the imbalances. Other techniques such as data augmentation wouldn't have work for this specific problem because rotations, reflections or other modification to the images would have compromised the chances to recognize the signs of the illness.

Anyway for random forests and support vector machines I didn't use the images provided but I used the features already extracted. Regarding convolutional neural networks I used the 256x256 greyscale images provided.

2.1. Random Forests

The first algorithm I tried was random forests. I trained a random forest of 50 trees using `sklearn.ensemble.RandomForestClassifier` and `entropy` estimator.

2.2. Support Vector Machines

The second algorithm I tried was support vector machines. I trained several `sklearn.svm.SVC` models with different parameters for `kernel` and `C` value. The one with the best overall accuracy (Acc) metric on validation set was the one I chose to compare with the other algorithms in section 3 (`kernel='rbf'` and `C=10`).

2.3. Convolutional Neural Networks

The third algorithm I tried was convolutional neural networks. I defined three convolutional neural network architectures, which differ by number of convolutional layers and/or node per layers. The base architecture for these CNN is the LeNet network built by professor Rota Bulò in one of his lectures. Unlike the first two algorithms, in this case for training and to make predictions I used the images provided.

1. LeNet:
    - 2 convolutional layers (conv2d, relu, maxpool2d)
    - 2 fully connected layers
    - output layer
2. CNN3:
    - 3 convolutional layers (conv2d, relu, maxpool2d)
    - 2 fully connected layers
    - output layer
3. CNN4:
    - 4 convolutional layers (conv2d, relu, maxpool2d)
    - 2 fully connected layers
    - output layer

All the network has been trained for 20 epochs with Adam optimizer and a decreasing learning rate over time. The main difference between these networks is on the number of convolutional layers. This allowed me to measure how the performance changed as I increased the depth of the network. Ideally more depth should lead to better performances.

3. Results

For each classifier i tried with different hyperparameters, here I report only the best performing ones.

3.1. Random Forests

With random forest model the performaces on the validation set were pretty low compared to the performances on the training set. This is a clear sign of overfitting. In addition the model was not able to learn to correctly classify COVID-19 samples even after applying class weighting tecnique as shown by the confusion matrix below. Training of this model is really fast: 1.85 seconds. Evaluation time on validation set: 0.017 seconds.

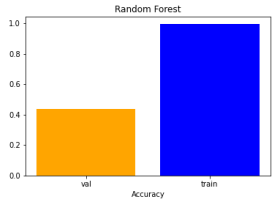


Figure 1. Training and validation accuracy of random forest.

```
[[482.,  18., 268., 270.],
 [  0.,   0.,   0.,   0.],
 [ 42.,   1.,  26.,  22.],
 [ 29.,   1.,  19.,  11.]]
```

Table 2. Confusion matrix random forest. (From left to right: Bacteria, Covid-19, Viral, Normal)

3.2. Support Vector Machines

With support vector machines the performaces were even worse than the random forest model. The problems were pretty much the same: overfitting and no ability to correctly classify COVID-19 samples. Below there are overall accuracy and confusion matrix on validation set. Training of this model is quite slow compared to random forest: 32.24 seconds. Evaluation time on validation set is also slower: 3.76 seconds.

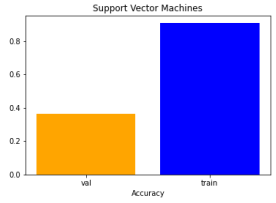


Figure 2. Training and validation accuracy of random forest.

```
[[269.,   7., 139., 130.],
 [  0.,   0.,   0.,   0.],
 [148.,  11.,  81.,  88.],
 [136.,   2.,  93.,  85.]]
```

Table 3. Confusion matrix SVM. (From left to right: Bacteria, Covid-19, Viral, Normal)

3.3. Convolutional Neural Networks

With convolutional neural networks the performaces on validation set were much better compared to the first two algorithms I used. Not only the accuracy has improved a lot but CNNs could recognize COVID-19 samples, making these models the best so far for this specific problem.

3.3.1 LeNet

LeNet model has an overall accuracy of 0.771 on the validation set, class averaged accuracy is 0.786. As the confusion matrix testimonies it is able to correctly classify 16/20 covid-19 samples. Compared to random forest and SVM the performances are astonishing. On the other hand training require much more time: 184.45 seconds. Evaluation time on validation was faster than SVM but slower than random forest: 2.31 seconds.

```
[[428.,   1.,  11., 113.],
 [  1.,  16.,   1.,   2.],
 [  7.,   0., 282.,  24.],
 [ 98.,   2.,  12., 191.]]
```

Table 4. Confusion matrix LeNet.

3.3.2 CNN3

CNN3 model has worse overall accuray than LeNet on the validation set. Correctly classified Covid-19 samples are 15/20. Regarding training and evalution time they are the same as LeNet. Adding one convolutional layer didn't bring concrete performance improvements on any side.

```
[[427.,   1.,  10., 115.],
 [  2.,  15.,   0.,   3.],
 [ 16.,   0., 278.,  19.],
 [113.,   2.,   9., 179.]]
```

Table 5. Confusion matrix CNN3.

3.3.3 CNN4

With CNN4 model I noticed slightly better performances. Overall accuracy has improved to 0.781 as well as class averaged accuracy which is 0.788. Adding two convolutional layer to LeNet brought better performaces, however training and evaluation time got worse. Training time was 318 seconds and evaluation time on validation set was 3.24 seconds. CNN is the model i tried on test set getting overall accuracy of 0.778 and positioning me in sixth place on the leaderboard.

```
[[427.,   1.,   7., 118.],
 [  1.,  12.,   2.,   5.],
 [  9.,   0., 281.,  23.],
 [ 85.,   2.,   7., 209.]]
```

Table 6. Confusion matrix CNN4.

Adding convolutional layers didn't bring the performace improvements expected. The ipothesis that explain this behaviour is that the dataset is not big enough to train the networks and have good generalization performances. However I observed that these models (LeNet and CNN4) didn't encounter overfitting issues because training accuracy and validation accuracy improved over the epochs, but maybe with more epochs of training they could overfit.

Regarding CNN3, comparing training and validation accuracy curve over the epochs shows that a there is a little overfitting. In fact from epoch 6 there is a drop in validation accuracy from 0.77 to 0.74, while training accuracy increases.

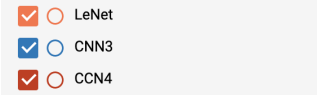


Figure 3. Legend of following plots

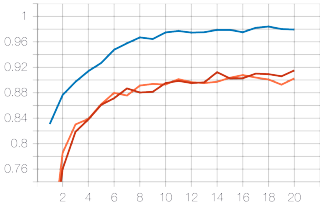


Figure 4. Training accuracy history of the three CNN.

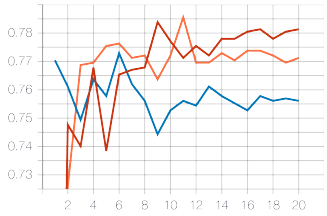


Table 7. Validation accuracy history of the three CNN.