Data Challenge #1 Report:

COVID-19 cases classification from chest X-ray images

STAT946 – University of Waterloo

Mohammad Zarei (Student ID: 20858700)

In this document, the process of model training and main results are briefly described. The training steps are based on the code sequences of the attached notebook ("*DL1_MZarei_majorCode.ipynb*") which has been written in Google Colab-Pro environment.

Data Loading and Preparation

The first two steps of the notebook include loading data and preparing train/validation sets. First, it will create a set of new directories "MZarei/" in which the data will be unzipped and split into train/validation sets. Also, it has been used for saving final trained model. Several transform function have been tested but finally just `transforms.Grayscale(3)` and `transforms.RandomHorizontalFlip(p=0.5)` are kept. Also, to alleviate the unbalanced nature of data a weighted sampler (8 to 1 ratio based on the class ratio of train set) has been used through data loader.

Define Train/Evaluate Functions and Model Architecture

In the steps 3, *Train* and *Evaluate* functions are defined based on the uploaded tutorial in Pizza. *Evaluate* function provides accuracy, f1 score, recall and precision of model over validation set (10% of data). I tested the impact of using weighted loss to manage unbalanced classes but better models were obtained without it. Also, I used Mixup augmentation method which shows a great improvement in the model but requires more epoch number.

In step 4, a classifier head is added to a pretrained *AlexNet* (several pretrained models like Vgg16, GoogleNet, SqueezeNEt have been tested but AlexNet was chosen finally). Up to epoch= 6, only the weights of classifier head will be updated during training and after that all weights will be adjusted based on backpropagation.

Train Model

Model training initiates in step 5 and the best model is selected based on F1 criteria. Running the training process several times has shown that we can get $F1 \approx 0.93 - 0.95$ and around 99% accuracy on validation set. I tried to fix random seeds so that the trained models can be reproducible, but it seems it is impossible as there are too many randomness in Pytorch so during each run new models will be obtained. So the final trained model may have around 1-2% lower accuracy than my best models (*model1 to model 10* in Models folder) that come with this document (those model are being trained with exactly same process but are selected after several runs and monitoring model performance over validation set).

Use Model for Test Data

In step 6, the best trained model on the basis of largest F1 score over validation set can be used for predicting test data. Due to randomness of training process the final accuracy of best model over test set may be a bit lower than leader board scores (it should be around 95-97%). In the following, the performance of 10 best models over my validation set are presented (all are available in *MZareiFolder/DL1_MZarei/Models*). These models have exactly same architecture/hyperparameters and have been trained with same train set.

Model	1	2	3	4	5	6	7	8	9	10		Bag-All
loss	0.043	0.044	0.037	0.040	0.049	0.043	0.039	0.039	0.036	0.051		-
accuracy	98.9%	98.8%	98.9%	98.8%	99.0%	99.0%	98.9%	98.6%	98.6%	98.8%		99.3%
f1	94.5%	94.1%	94.6%	94.3%	95.0%	95.1%	94.4%	93.3%	93.5%	94.3%		96.4%
recall	92.4%	96.2%	95.5%	95.5%	91.7%	93.0%	91.1%	93.0%	95.5%	94.3%		94.9%
precession	96.7%	92.1%	93.8%	93.2%	98.6%	97.3%	97.9%	93.6%	91.5%	94.3%		98.0%
Kaggle test			95.75%		94.0%	97.0%	96.0%		96.5%			96.0%

An interesting observation is that using bagging method over all models provides better results than each individual model (this can be tested in the last two parts of the notebook). However, the performance of bag-model is worse than the performance of best individual model over the test data in Kaggle. The best model in terms of Kaggle test

is *model6.pt* with 97% accuracy. The reason for this may be because of relatively low number of test data.

Note that I attached another code (*DL1_MZarei_noMixup.ipynb*) in which I have not used *Mixup* and *RandomFlip,* and the model seems to be reproducible (at least in my own system). With the trained model using that code, you should be able to get the accuracy of 96.75% over Kaggle test data and 98.62% over its validation set.

Thank you for your efforts.