Dataset

During an AI development, we always rely on data. From training, tuning, model selection to testing, we use three different data sets: the training set, the validation set ,and the testing set. We might think that the gathering of data is enough, but it is the opposite.

In every AI projects, classifying and labeling data sets takes most of our time , especially data sets accurate enough to reflect a realistic vision of the world. Over here I want to introduce to the first two types of data sets we would be needing for this project — the training data set and test data set. They both have different purposes during our AI project and the success of a project depends a lot on them.

- Training Data-The training data set is the one used to train an algorithm to understand how to apply concepts such as neural networks, to learn and produce results. It includes both input data and the expectedoutput.

- Testing Data-The test data set is used to evaluate how well your algorithm was trained with the training data set. In AI projects, we can't use the training data set in the testing stage because the algorithm will already know in advance the expected output which is not ourgoal.

In our project we collected different images for person with mask, person without mask and not person. We tried our best to collect images of person (with mask and without mask) of different colour, gender and age. We also collected the different images which are not human e.g. dog, cat, plane, monkey etc. Our dataset contains 1000 sample images in each theclass.

For certain images we just use datasets where we were able to find plenty of image sample. But for some cases we had to really come up with image samples which we couldn't find in any dataset repositories. We tried to make sure to have balanced dataset between the classes as well as categories of samples considered for a singleclass.

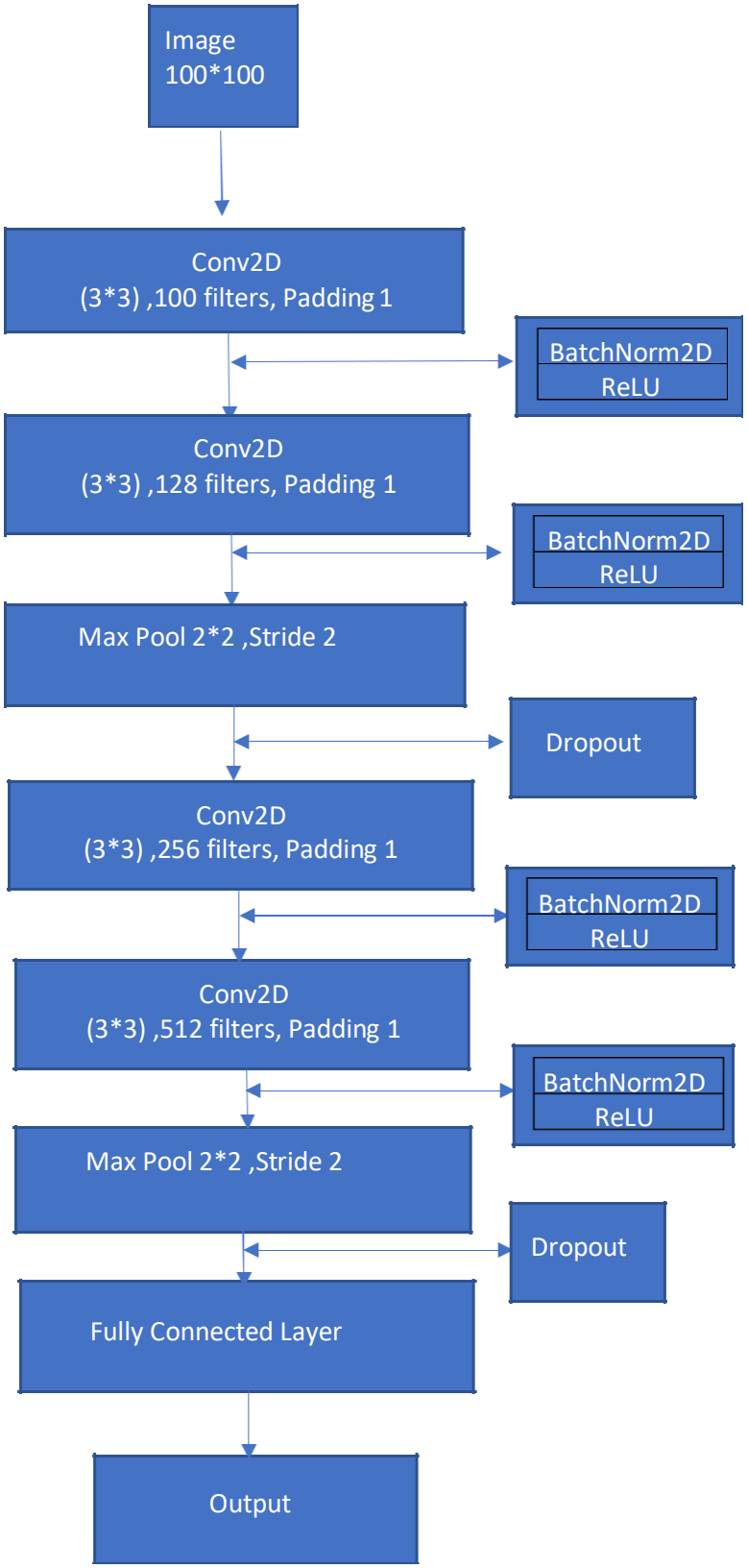
CNNArchitecture

Before training the dataset, we have to preprocess the data set. Number of images for differentclassesand the sources of the images are given previously. So, we set the three-dataset path. First will be Masked human dataset, second will be unmasked and the last will be not a human. After setting up path for three different dataset, we iterate through these three different data set and resize all the images to 100*100 pixels. We also append it to a NumPy array along with thelabels.

So, the label "0" represents masked human "1" represents human and "2 " represents non-human. We then divide the dataset into trainVal and test dataset. 5% of the data goes to testing. 10 % for the validation and the rest for testing. Then we try to normalize the weights of the images. This is because we have unbalanced number of images for different dataset. When dealing with unbalanced data, we need to pass this information to the loss function to avoid unproportioned step sizes of the optimizer. We do this by assigning a weight to each class, according to its representability in the dataset. We assign more weight to classes with a small number of samples so thatthenetwork will be penalized more if it makes mistakes predicting the label of these classes. While classes with large numbers of samples, we assign to them a smaller weight. Following is the formula for normalising theweight.

Class weight=1-((class cardinality)/sum(all class cardinality))

After this, we load our training, validation and testing dataset and convert it to tensor along with the labels. After converting it we load using data loader with the batch size of 32 to load the data set to train Data loader, validation Data loader and testing Data loader



Now, according to the CNN diagram we take an input image of size 100*100*3. And this image is passed to a Convolution layer(1st Convo Layer) with an input channel of 3 (RGB channel) output channel of 100, kernel size or filter size of 3 and padding of 1. Now this output of convolution layer will be normalised and then passed to the activation function ReLU (ReLU takes the maximum of the input between 0 and1). It is passed then again passed to next Convolution layer(2nd Convo Layer) where the normalisation and the activation function takes place(just like1st Convo Layer). The input channel to this layer will be 100 as it was output of the previous channel. The output channel will be 128 . The kernel size and the padding will remain the same just like previous layer. Then after that it is fed to maxpooling layer of stride 2. This will help in pertaining the information of the image making it smaller in size. We also introduce the Regularization method that is Dropout after maxpooling . This is used to avoid overfitting or under fitting of the data. In dropout we deactivate 5 percent of neurons while connecting it to another hiddenlayer.

After dropout we connect it to the convolution layer(3rd Convo Layer) with input channel of 128(the output channel of previous layer will become the input channel of the next layer). The output channel will be 256. Padding and kernel size will remain as it is. In this layer it will follow the same step of normalization and ReLU activation function. Just like the previous layer which we discussed, the output from the ReLU will be fed to 4th convolution layer with input and output channel of 256 and 512. Again, passing it to the normalization function and activation function. After that maxpooling will be applied. Again, there will be a dropout of 5% in the neurons to avoid over or underfitting of data.

After the convolution layer , the output is passed to the fully connected layer where we apply similar normalization function, activation function and a dropout of 20%. The output of from the fully connected network will be 3. This is because we have to produce any of the 3-output label. That is “masked” “non masked human” or “not human”

K fold Cross Validation

Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modeling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

K fold cross validation procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of kinthe reference to the model, such as k=10 becoming 10-foldcross-validation.

In this model the total number of our dataset is divided into equal number of folds that is provided by k. Then in each iteration one of the fold is considered as validation set and other folds are considered as training set. Figure 4 demonstrates the following concept.

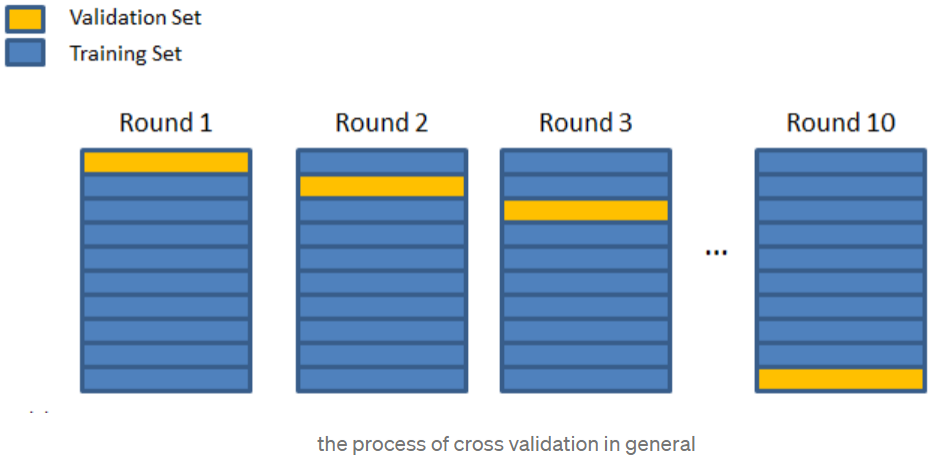


Figure 4: K-Cross Validation (Raheel,Shaikh 2018)

In this process we will be using the same CNN architecture which we used in the first phase. Before training the image we have to pre process the image. This procedure is similar to the first phase of the project which we have discussed before. For implementing the phase 2 of our project we have used Skorch and GridSearchCVlibraries.

Skorch is one of the useful libraries in Pytorch to work on machine learning models especiallyneuralnetworks.It is a robust library that works in the combination of PyTorch and Scikit-learn. Skorch enables programmers to implement code using the customizability scikit-learn and power of PyTorch. When we call model.fit() and we don't have to worry about writing our own callback functions, skorch will handle everything for us. Skorch is an inbuilt library in PyTorch used for Neural networks. In building deep neural networks, we are required to train our model, evaluate, dividing the train & validation data, and many other operations required to build a model. Here skorch library will reduce the effort of the programmer similar to scikit learn library in machine learning. Along with scorch we also use GridSearchCV.

GridSearchCV is also a library function that is a member of sklearn's model_selection package. It also helps to loop through predefined hyperparameters and fit our model on the training set. So, in the end, we

can select the best parameters from the listed hyperparameters. In addition to that, we can specify the number of times for the cross-validation which is required for our project.

In Phase 2, we train our model using k-fold validation with 95% of the images from the dataset. We reserve the remaining the 5% of the images for testing the trained model. Now we split the 95% of the data into 10-fold and train the model using k-fold validation. We preprocess the images to 100*100*3 pixels. Then we pass the training data set to the neural net classifier using scorch library . In the neural net classifier, we pass the parameter such as the learning rate , optimizer , CNN and the entropy we want to pass. Then we also define the GridSearchCV, where we pass the other parameter such as the CNN, folds for the cross validation, scoring ,verbose and return test score. We train our model using model.fit(data,label). This will automatically train our model without any need of writing code for training or validation loop. After this we save our best estimator model and use it to test our model with the test data which we reserved earlier and then print the confusion matrix and classification report.

Once the GridSearchCV completes the training, it returns the best model based on the best score value. The best score indicates the maximum accuracy value obtained by the model during each fold while testing the model on the validation data. The best model returned by the GridSearchCV had an mean accuracy value of 87%, and this was obtained while we performed the training for 15 epochs.

Figure 5 gives the accuracy for each fold for both testing and training data. We do it for two Epochs and save the best model.

Folds	10 Epochs		15 Epochs	
	Training	Testing	Training	Testing
1	0.902	0.870	0.947	0.891
2	0.935	0.874	0.951	0.856
3	0.923	0.884	0.943	0.877
4	0.912	0.867	0.965	0.884
5	0.913	0.856	0.967	0.881
6	0.906	0.807	0.953	0.870
7	0.889	0.828	0.945	0.849
8	0.909	0.874	0.970	0.905
9	0.911	0.839	0.954	0.832
10	0.923	0.888	0.954	0.895

Figure 5: Accuracy for each Folds

As discussed above we can see that model with 15 Epochs has a better accuracy compared to the first one. So, we save the second model and use it for testing our unseen dataset.

Using the best model returned by the grid search we tested the model again on 95% of the images from the dataset which we used for the initial k fold training. The Following are the classification report and confusion matrix obtained for the testing we performed. The accuracy of the model is 96%. As discussed before our model uses scorch and GridSearchCV to train our data and for crossvalidation

Figure 6 represents the classification report and confusion matrix for the training data we discussed above.

Train Classification Report

	precision	recall	f1-score	support
0	0.99	0.92	0.96	950
1	1.00	0.98	0.99	950
2	0.91	0.99	0.95	950
accuracy			0.96	2850
macro avg	0.97	0.96	0.96	2850
weighted avg	0.97	0.96	0.96	2850

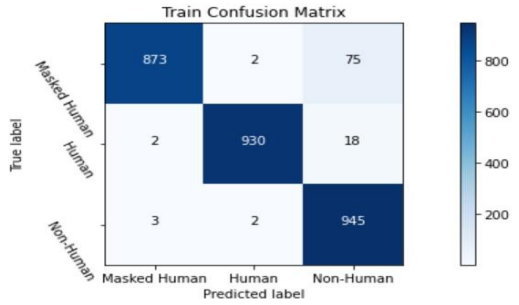


Figure 6:Confusion Matrix for training data

Once we trained our model using k-fold validation, we test the performance of our model on the 5% of the images that we reserved from the dataset which our model has never seen before. Figure 7 is the classification report and confusion matrix we obtained while we evaluated our model on the test images. The accuracy of the model is92%.

Build DataLoader

Test Classification Report

	precision	recall	f1-score	support
0	0.98	0.84	0.90	50
1	0.93	1.00	0.96	50
2	0.87	0.92	0.89	50
accuracy			0.92	150
macro avg	0.92	0.92	0.92	150
weighted avg	0.92	0.92	0.92	150

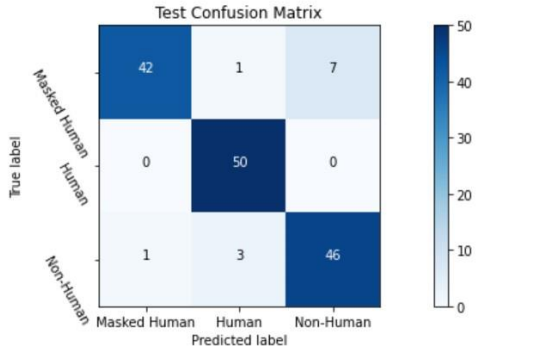


Figure 7: Confusion Matrix for testing data

We have created separate function to predict individual images. Those images are stored in a separate folder called “Test”. To predict individual image our model uses the best saved model to predict the label. The results are stored in a separate folder called “Result”. It will be appending the images along with the label. We are attaching some of the images for which our model predicted the labels correctly and incorrectly. Following are the numbers along with the labels. 0:Masked Human 1:Human 2:Non-Human.

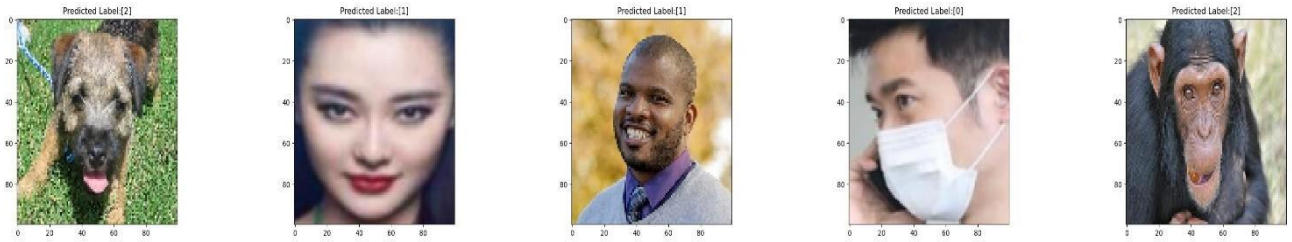


Figure 8: Labels predicted correctly by the model

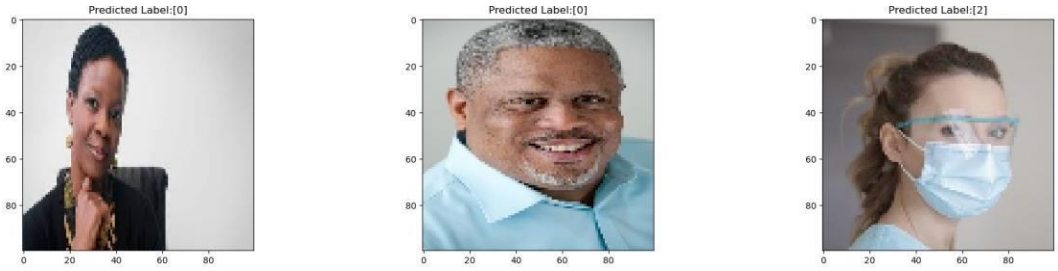


Figure 9: Labels predicted incorrectly by the model

References

- Haddad,J 2020 May 24. How I built a Face Mask Detector for COVID-19 using PyTorch Lightning. Retrieved from https://towardsdatascience.com/how-i-built-a-face-mask-detector-for-covid-19-using-pytorch-lightning-67eb3752fd61
- Saha, A 2018 June 5. Deep Learning with PyTorch. Retrieved from https://learning.oreilly.com/videos/deep-learning-with/9781788475266/9781788475266-video3_6
- Waqas, A. 2020 June 20. Mask Detection Project. Retrieved from https://jovian.ai/aliwaqas333/course-project-mask-detection
- Mayo, M. 2018 May 1. PyTorch Tensor Basics . Retrieved from https://www.kdnuggets.com/2018/05/pytorch-tensor-basics.html
- Mwiti,D.2018Nov10.IntroductiontoPyTorchforDeepLearning.RetrievedFrom https://www.kdnuggets.com/2018/11/introduction-pytorch-deep-learning.html
- Haddad,J.2020May.MaskDetectionusingPyTorchLightning.Retrieved from https://jovian.ai/aliwaqas333/course-project-mask-detection
- Raheel ,Shaikh.2018 Nov 26. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85
- Sharma, Mathanraj.2020 March 20. https://towardsdatascience.com/grid-search-for-hyperparameter-tuning-9f63945e8fec
- Skorch-dev. https://github.com/skorch-dev/skorch
- GridSearchCV. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html