



Deep Learning laboratory

Statistical learning

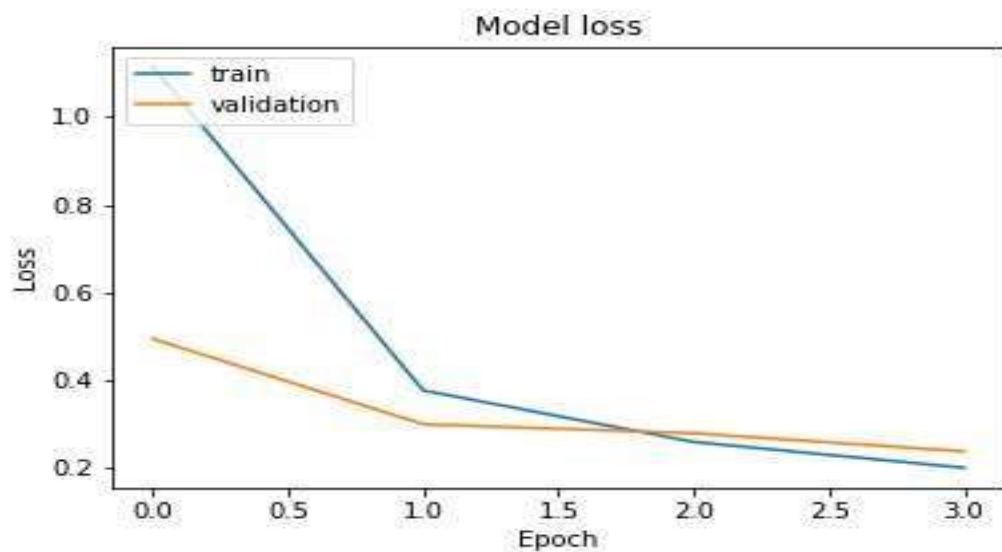
Ankit Rathi

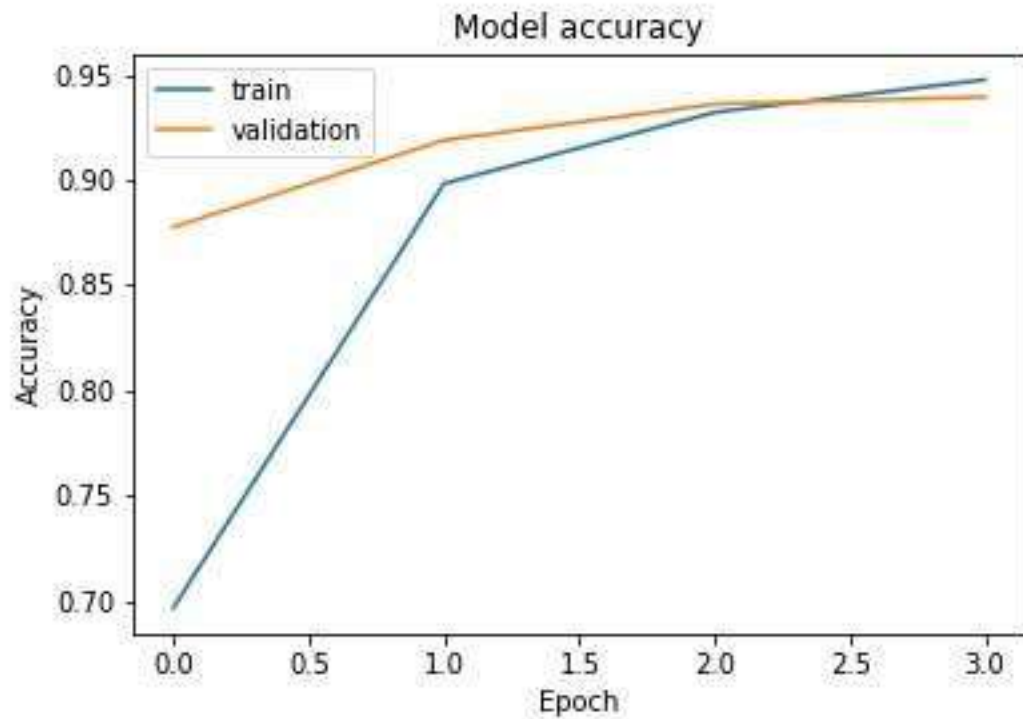
1. Load the Data and train the data containing two convolutional layer, and plot the graphs of loss function calculated for training set and validation set on one graph? What do you see ?

```
model = Sequential()
n_filters=40
model.add(Conv2D(n_filters,
                 (3,3),
                 input_shape=(20,20,3)))
model.add(MaxPooling2D((3,3)))
n_filters=60
model.add(Conv2D(n_filters, (3,3)))
model.add(Flatten())
model.add(Dense(n_filters))
n_filters=80
n_class=43
model.add(Dense(n_class))

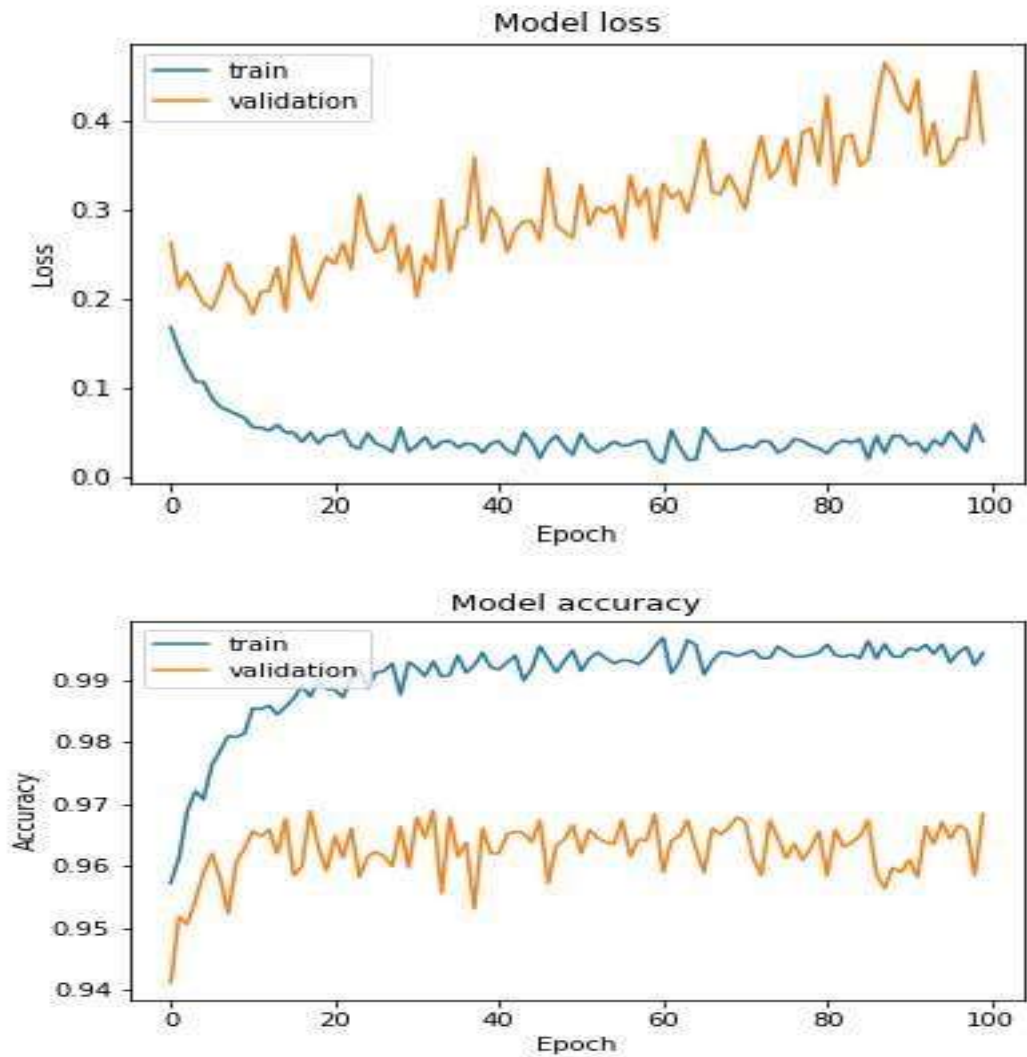
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])

model.build()
model.summary()
```





As we can see from the model loss and model accuracy graph, we can say that loss of training set was higher than validation but after a like epochs the loss of training set became less than the validation set, this can be due to fact that , training loss is measured during each epochs while validation loss is measured after each epochs, I have checked the same by running the model for 100 epochs and I have got the same result.



After running the model for 100 epochs we can see that after few number of epochs the training loss is decreasing subsequently, whereas validation loss is getting higher after every while.

2. Based on your observation from point 1, what kind of regularization we can implement to obtain a better relation of loss on training and validating? Implement these techniques. Comment the corrected result

Regularization applied during training, but not during validation/testing.  
When training a deep neural network we often apply **regularization** to help our model:

1. Obtain higher validation/testing accuracy
2. And ideally, to **generalize better** to the data outside the validation and testing sets

Regularization methods often sacrifice training accuracy to improve validation/testing accuracy — in some cases that can lead to your validation loss being lower than your training loss.

Secondly, we must keep in mind that regularization methods such as dropout are *not* applied at validation/testing time.

Applying dropout during validation/testing time) can make your training/validation loss curves look more similar.

The second reason you may see validation loss higher than training loss is due to how the loss value are measured and reported:

1. Training loss is measured *during* each epoch
2. While validation loss is measured *after* each epoch

Your training loss is continually reported over the course of an entire epoch; however, **validation metrics are computed over the validation set *only once the current training epoch is completed.***

This implies, that on average, training losses are measured half an epoch earlier.

If we shift the validation losses half an epoch to the left you'll see that the gaps between the training and losses values are much smaller.

The final most common reason for validation loss being lower than your training loss is due to the data distribution itself.

3. Check the number of images for each class. What do you see? Write appropriate function to deal with the problem.

```
[4] # Data_generating
training_generator=ImageDataGenerator(rescale=1/255,
                                     rotation_range=20,
                                     zoom_range=0.05,
                                     width_shift_range=0.1,
                                     height_shift_range=0.1,
                                     shear_range=0.05)

validation_generator = ImageDataGenerator(rescale=1/255)

BS=32 #batch size
testing_flow=training_generator.flow_from_directory(directory="/content/drive/My Drive/IABSL/test",
                                                    class_mode="categorical",
                                                    target_size=(20,20),
                                                    color_mode='rgb',
                                                    shuffle=True,
                                                    batch_size=BS)

BS=32 #batch size
validation_flow=validation_generator.flow_from_directory(directory="/content/drive/My Drive/IABSL/val",
                                                         class_mode="categorical",
                                                         target_size=(20,20),
                                                         color_mode='rgb',
                                                         shuffle=True,
                                                         batch_size=BS)

BS=32 #batch size
training_flow=validation_generator.flow_from_directory(directory="/content/drive/My Drive/IABSL/train",
                                                       class_mode="categorical",
```

Total number of images in each classes is as follow –

Total\_train --27446

Total\_test -- 7842

Total\_validation – 3921

Training Dataset --- The actual dataset that we use to train the model (weights and biases in the case of Neural Network). The model *sees* and *learns* from this data.

Validation Dataset -- *The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.*

Test dataset -- *The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.*

4. Train the network and verify its quality on a testing set. Calculate the quality indices such as: accuracy, specificity and sensitivity

```
prediction=model.predict_generator(testing_flow,
steps=total_test//32+1) predicted_classes=np.argmax(prediction,
axis=1)
```

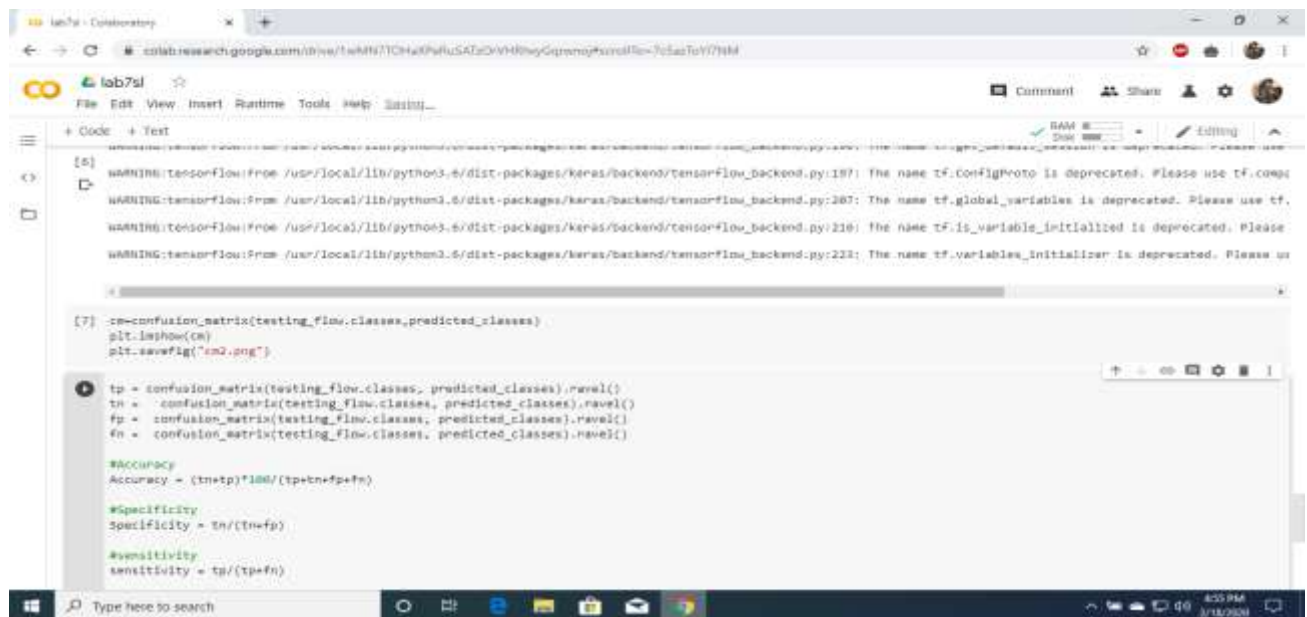
Accuracy –

Sensitivity –

Specificity –

I have used the exact code to check the accuracy / specificity / sensitivity but google.colab didn't work perfectly and kind of didn't give any output/error.

Here is a screenshot from the same –



The screenshot shows a Google Colab notebook interface. The top bar includes the 'lab7al' logo and a search bar. The main area contains a code cell with the following Python code:

```
[5]
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:197: The name tf.configProto is deprecated. Please use tf.compat.v1.config_proto instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:210: The name tf.is_variable_initialized is deprecated. Please use tf.compat.v1.is_variable_initialized instead.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:223: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

[7] cm=confusion_matrix(testing_flow.classes, predicted_classes)
    plt.imshow(cm)
    plt.savefig("cm2.png")

tp = confusion_matrix(testing_flow.classes, predicted_classes).ravel()
tn = confusion_matrix(testing_flow.classes, predicted_classes).ravel()
fp = confusion_matrix(testing_flow.classes, predicted_classes).ravel()
fn = confusion_matrix(testing_flow.classes, predicted_classes).ravel()

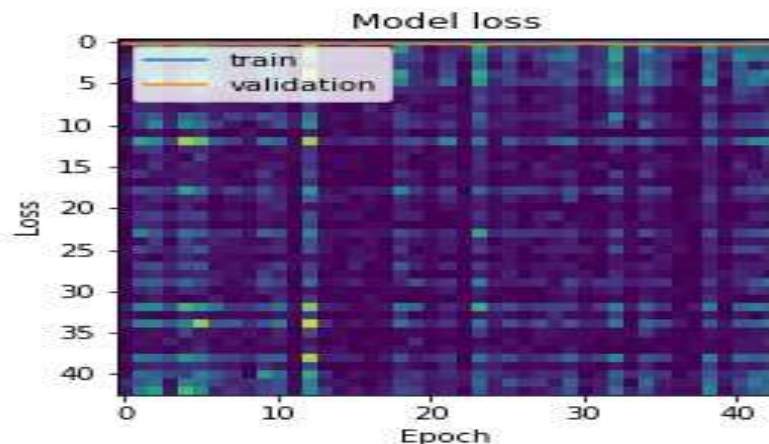
#Accuracy
Accuracy = (tn+tp)/(tp+tn+fp+fn)

#Specificity
Specificity = tn/(tn+fp)

#Sensitivity
Sensitivity = tp/(tp+fn)
```

The bottom of the image shows a Windows taskbar with a search bar and several application icons.

5. Plot the confusion matrix. Identify the data for which network has a problem. Try to explain why?



We have plotted a confusion matrix for the model loss on the test dataset, A confusion matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is  $N \times N$ , where  $N$  is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix.

- **Accuracy** : the proportion of the total number of predictions that were correct.
- **Positive Predictive Value** or **Precision** : the proportion of positive cases that were correctly identified.
- **Negative Predictive Value** : the proportion of negative cases that were correctly identified.
- **Sensitivity** or **Recall** : the proportion of actual positive cases which are correctly identified.
- **Specificity** : the proportion of actual negative cases which are correctly identified.
- Our network has a problem with the training set , we use it basically to avoid overfitting but as we can see in our model we have got high amount of variance , as we have low training set loss and higher validation loss and Unrepresentative Data Sample, The causes of overfitting, underrepresentative data samples, and stochastic algorithms.



- ideas of *large differences* in model performance are relative to our chosen performance measures, datasets, and models. We cannot talk objectively about differences in general, only relative differences that we must interpret ourself.