Written Report (40 points): The report should be delivered as a separate pdf file, and it is recommended for you to use the NIPS template to structure your report. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the report. The report should describe your results, experimental setup and comparison between the results obtained from different settings of the algorithm and dataset.**Again, the questions in the Assignment PDF and here are the same (for the written report), we just put them in both places for convenience.**

Part 1. loading dataset

Found 4176 images belonging to 10 classes.
Found 1392 images belonging to 10 classes.
Found 1392 images belonging to 10 classes.
No. of sampled in train folder 4176
No. of sampled in validation folder 1392
No. of sampled in test folder 1392

Part 2.1 - Build your Neural Network and Train

Build a Convolutional Neural Network with 2 or 3 hidden layers without regularization methods, which includes Conv2D layer, activation Layer. please use training dataset and validation dataset for training process, and plot the training process with Loss trend and accuracy trend (30 Points).

Part 2.2 - Test

Test your machine learning model on the testing set: After finishing all the above steps, fix your hyper-parameters(learning rate, number of neurons per layer) and model parameter and test your model's performance on the testing set. This shows the effectiveness of your model's generalization power gained by learning. For the test dataset, the performance should be more than 80% (10 Points).

Ans: The hyperparameters that were used to find the best results are as follows:-

1. The batch size for the image generator was increased from 16(default) to 32, which worked best for this case. Batch size equal to 64 did not improve the model's performance, because the weight updates weren't as frequent as in the case of 32.
2. Also the image's target size from the generator was too large. When the image size was reduced the model seemed to pick up better features during convolution operations.

When reduced below 64x64 , the image obtained at the end without padding was too small as a result of convolutions; dropout further made the model too simple.

3. The optimizer used was RMSProp with learning rate 0.001 gave the best performance. Other smaller values for learning rates were tried but had poor performance for smaller numbers of epochs. And when trained for a larger number of epochs, the model did not seem to improve after some epochs.
4. Three convolutions layers along with 2 max pooling layers to extract position independent features were used. The model seemed to overfit a lot with the absence of the dropout layer. This was done so that l1 and l2 regularizations in the next steps would further help to reduce overfitting.

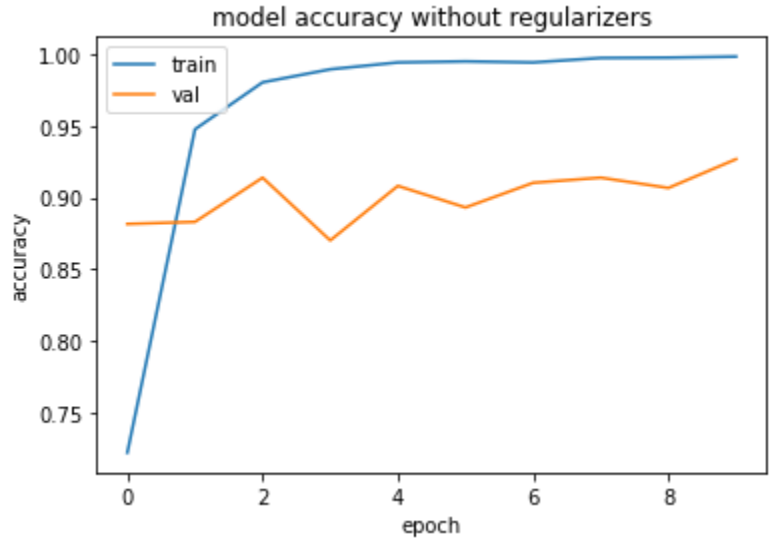
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 64)        1792

 max_pooling2d (MaxPooling2D  (None, 31, 31, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        36928

 flatten (Flatten)           (None, 9216)              0

 dropout (Dropout)           (None, 9216)              0

 dense (Dense)               (None, 10)                92170

=================================================================
Total params: 167,818
Trainable params: 167,818
Non-trainable params: 0
_________________________________________________________________
```

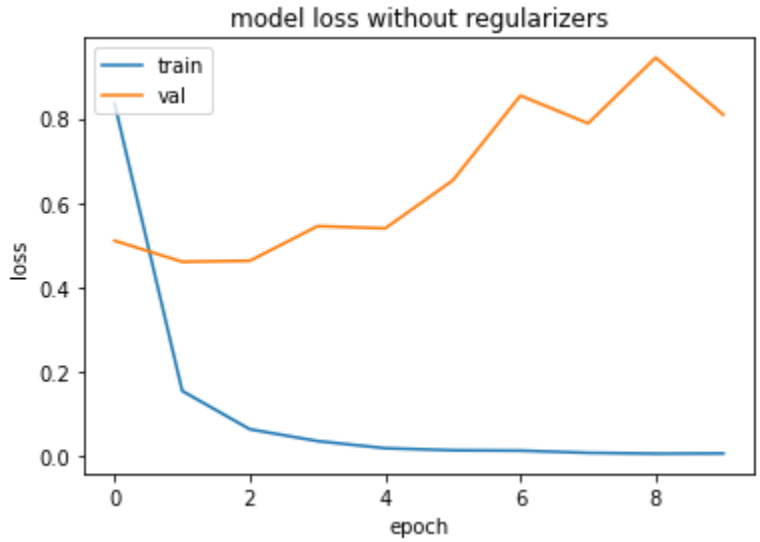
Testdata Evaluation

44/44 [==============================] - 3s 79ms/step - loss: 1.3473 - accuracy: 0.8714

Loss without regularizers: **1.3473137617111206**

Accuracy without regularizers: **0.8714080452919006**





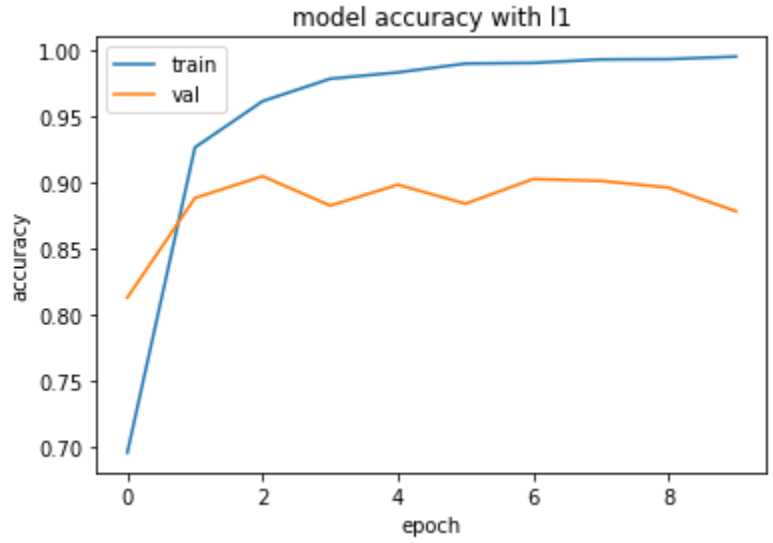
Part 2.3 - L1 Regularization. Please add the L1 regularization setting in your Conv2D layer. Then, train your new model separately, and plot the training process including loss and accuracy. (10 points)

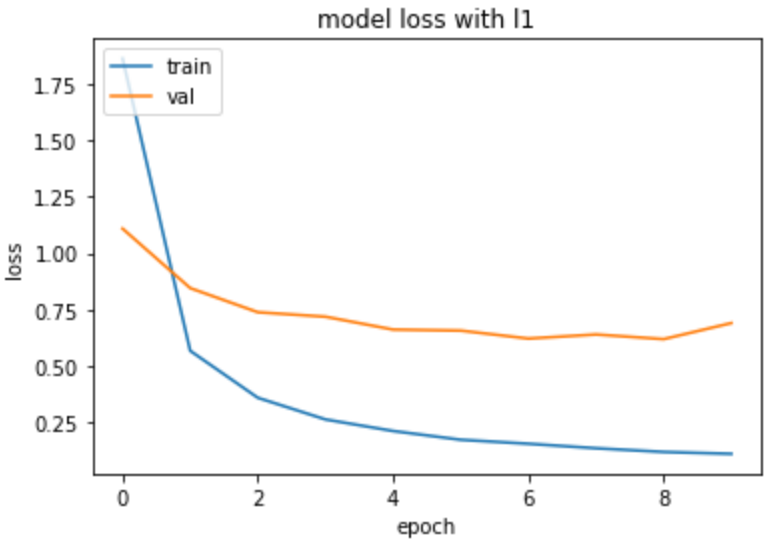
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 62, 62, 64)        1792

 max_pooling2d_2 (MaxPooling  (None, 31, 31, 64)       0
 2D)

 conv2d_4 (Conv2D)           (None, 29, 29, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 12, 12, 64)        36928

 flatten_1 (Flatten)         (None, 9216)              0

 dropout_1 (Dropout)         (None, 9216)              0

 dense_1 (Dense)             (None, 10)                92170

=================================================================
Total params: 167,818
Trainable params: 167,818
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
131/131 [==============================] - 15s 109ms/step - loss: 1.8612 - accuracy: 0.6959 - val_loss: 1.1085
- val_accuracy: 0.8132
Epoch 2/10
131/131 [==============================] - 14s 107ms/step - loss: 0.5679 - accuracy: 0.9270 - val_loss: 0.8453
- val_accuracy: 0.8886
Epoch 3/10
131/131 [==============================] - 14s 107ms/step - loss: 0.3599 - accuracy: 0.9619 - val_loss: 0.7384
- val_accuracy: 0.9052
Epoch 4/10
131/131 [==============================] - 14s 107ms/step - loss: 0.2641 - accuracy: 0.9789 - val_loss: 0.7186
- val_accuracy: 0.8829
Epoch 5/10
131/131 [==============================] - 14s 106ms/step - loss: 0.2130 - accuracy: 0.9837 - val_loss: 0.6617
- val_accuracy: 0.8987
```

Epoch 6/10
131/131 [==============================] - 14s 106ms/step - loss: 0.1741 - accuracy: 0.9904 - val_loss: 0.6571 - val_accuracy: 0.8843
Epoch 7/10
131/131 [==============================] - 14s 107ms/step - loss: 0.1565 - accuracy: 0.9909 - val_loss: 0.6219 - val_accuracy: 0.9030
Epoch 8/10
131/131 [==============================] - 14s 107ms/step - loss: 0.1367 - accuracy: 0.9935 - val_loss: 0.6401 - val_accuracy: 0.9016
Epoch 9/10
131/131 [==============================] - 16s 121ms/step - loss: 0.1202 - accuracy: 0.9938 - val_loss: 0.6195 - val_accuracy: 0.8966
Epoch 10/10
131/131 [==============================] - 14s 107ms/step - loss: 0.1120 - accuracy: 0.9957 - val_loss: 0.6904 - val_accuracy: 0.8786

Testing Accuracy

44/44 [==============================] - 3s 79ms/step - loss: 0.8524 - accuracy: 0.8570
Loss with l1: 0.8524300456047058
Accuracy with l1: 0.8570402264595032




model loss with l1
train
val
1.75
1.50
1.25
1.00
0.75
0.50
0.25
loss
0
2
4
6
8
epoch

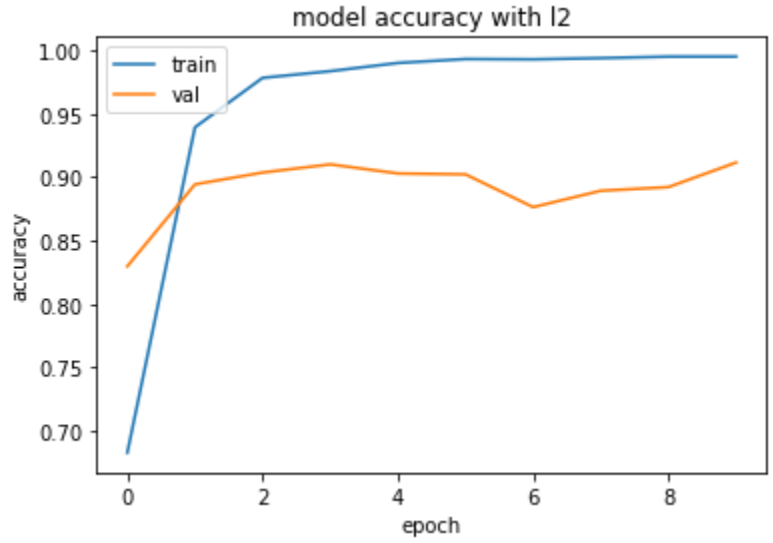
Part 2.4 - L2 Regularization. Please add the L2 regularization setting in your Conv2D layer. Then, train your new model separately, and plot the training process including loss and accuracy. (10 points)

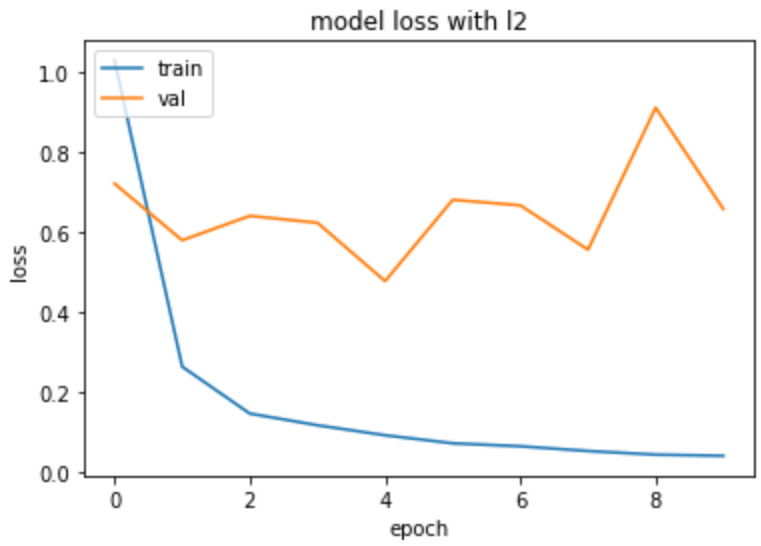
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 62, 62, 64)        1792

 max_pooling2d_4 (MaxPooling  (None, 31, 31, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 29, 29, 64)        36928

 max_pooling2d_5 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_8 (Conv2D)           (None, 12, 12, 64)        36928

 flatten_2 (Flatten)         (None, 9216)              0

 dropout_2 (Dropout)         (None, 9216)              0

 dense_2 (Dense)             (None, 10)                92170

=================================================================
Total params: 167,818
Trainable params: 167,818
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
131/131 [==============================] - 15s 108ms/step - loss: 1.0271 - accuracy: 0.6827 - val_loss: 0.7203
- val_accuracy: 0.8297
Epoch 2/10
131/131 [==============================] - 14s 107ms/step - loss: 0.2645 - accuracy: 0.9394 - val_loss: 0.5793
- val_accuracy: 0.8944
Epoch 3/10
131/131 [==============================] - 14s 106ms/step - loss: 0.1477 - accuracy: 0.9784 - val_loss: 0.6401
- val_accuracy: 0.9037
Epoch 4/10
131/131 [==============================] - 14s 106ms/step - loss: 0.1186 - accuracy: 0.9837 - val_loss: 0.6228
- val_accuracy: 0.9102
Epoch 5/10
131/131 [==============================] - 14s 107ms/step - loss: 0.0937 - accuracy: 0.9902 - val_loss: 0.4775
- val_accuracy: 0.9030
```

Epoch 6/10
131/131 [==============================] - 14s 107ms/step - loss: 0.0737 - accuracy: 0.9933 - val_loss: 0.6802 - val_accuracy: 0.9023
Epoch 7/10
131/131 [==============================] - 14s 108ms/step - loss: 0.0663 - accuracy: 0.9931 - val_loss: 0.6662 - val_accuracy: 0.8764
Epoch 8/10
131/131 [==============================] - 14s 107ms/step - loss: 0.0545 - accuracy: 0.9940 - val_loss: 0.5558 - val_accuracy: 0.8894
Epoch 9/10
131/131 [==============================] - 14s 107ms/step - loss: 0.0455 - accuracy: 0.9952 - val_loss: 0.9097 - val_accuracy: 0.8922
Epoch 10/10
131/131 [==============================] - 14s 106ms/step - loss: 0.0425 - accuracy: 0.9952 - val_loss: 0.6574 - val_accuracy: 0.9116

Testdata Evaluation

44/44 [==============================] - 3s 79ms/step - loss: 0.7852 - accuracy: 0.8822
Loss with l2: 0.785243570804596
Accuracy with l2: 0.8821839094161987





Ans: The most important observation after adding regularizations to the model in part 2.1 was that l2 regularizer seemed to generalize better than l1 with the same amount of lambda (0.001 in our case). This was probably because the image pixels tend to be strongly correlated and l2 generally works better in this case.
The magnitude of regularization lambda when increased resulted in a decrease in test accuracy because the model was underfit due to the presence of 50% dropout. When decreased, the model could not generalize well and resulted in a lower test accuracy.

Part3 - ** only for 574 students **.

Fine tune the well pre-trained model, Resnet 50, with different freeze layers. First, load pre-trained resnet 50 from library. Second, Fine-tune the model to fit our project, 10-classes. Third, freeze different layers, plot different training process with different frozen layers (at least three different layers).

Found 4176 images belonging to 10 classes.
Found 1392 images belonging to 10 classes.
Found 1392 images belonging to 10 classes.

Downloading data from
https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 0s 0us/step
94781440/94765736 [==============================] - 0s 0us/step
Model: "model"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
tf.__operators__.getitem (SlicingOpLambda)	(None, 224, 224, 3)	0
tf.nn.bias_add (TFOpLambda)	(None, 224, 224, 3)	0
resnet50 (Functional)	(None, 7, 7, 2048)	23587712
flatten_3 (Flatten)	(None, 100352)	0
dense_3 (Dense)	(None, 10)	1003530

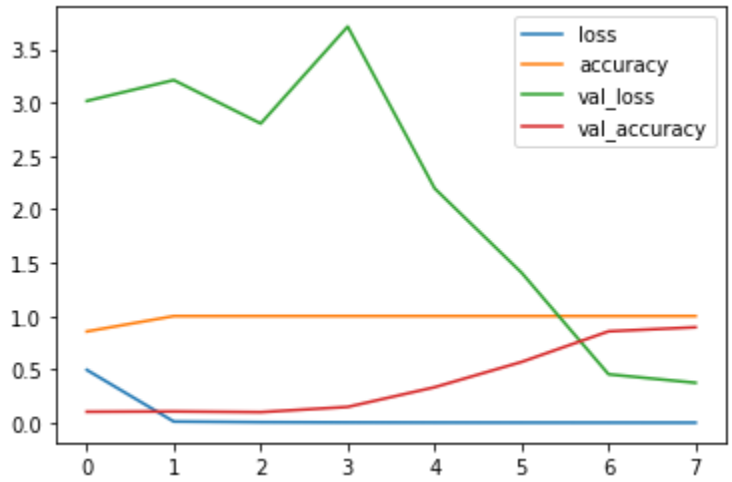
Total params: 24,591,242
Trainable params: 24,538,122
Non-trainable params: 53,120

Epoch 1/8
131/131 [==============================] - 59s 395ms/step - loss: 0.4952 - accuracy: 0.8558 - val_loss: 3.0158 - val_accuracy: 0.1034
Epoch 2/8
131/131 [==============================] - 49s 377ms/step - loss: 0.0117 - accuracy: 0.9998 - val_loss: 3.2113 - val_accuracy: 0.1063

Epoch 3/8
131/131 [==============================] - 50s 378ms/step - loss: 0.0056 - accuracy: 1.0000 - val_loss: 2.8044 - val_accuracy: 0.0999
Epoch 4/8
131/131 [==============================] - 49s 377ms/step - loss: 0.0034 - accuracy: 1.0000 - val_loss: 3.7129 - val_accuracy: 0.1487
Epoch 5/8
131/131 [==============================] - 50s 378ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 2.1961 - val_accuracy: 0.3333
Epoch 6/8
131/131 [==============================] - 50s 379ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 1.4060 - val_accuracy: 0.5704
Epoch 7/8
131/131 [==============================] - 51s 387ms/step - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.4543 - val_accuracy: 0.8570
Epoch 8/8
131/131 [==============================] - 50s 381ms/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.3748 - val_accuracy: 0.8951

Testdata Evaluation

44/44 [==============================] - 6s 131ms/step - loss: 0.4163 - accuracy: 0.8628



Ans: The best hyperparameters were chosen after these observations :-

1. The learning was chosen such that it was low enough for this process to be called fine tuning :p . And was high enough so that it converged faster with less number of epochs. (8 in our case)

2. Image size of 64x64 seemed too small for resnet because of the large number of convolution blocks even though 3 of them were frozen. So, the full image 224x224x3 seemed to provide better performance.