Auto Encoder Assignment: Report

Mohit Kumar Soni

SC15B103

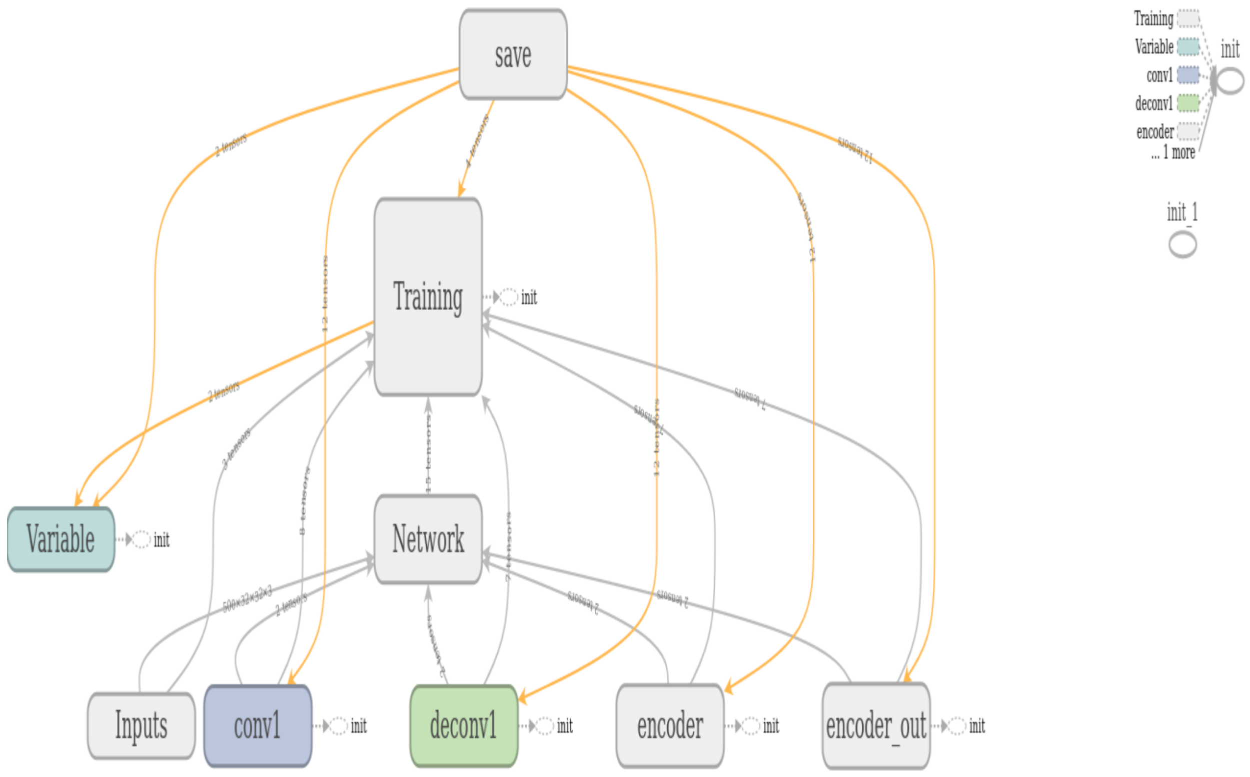
Data set used: - Cifar-10

Theory:

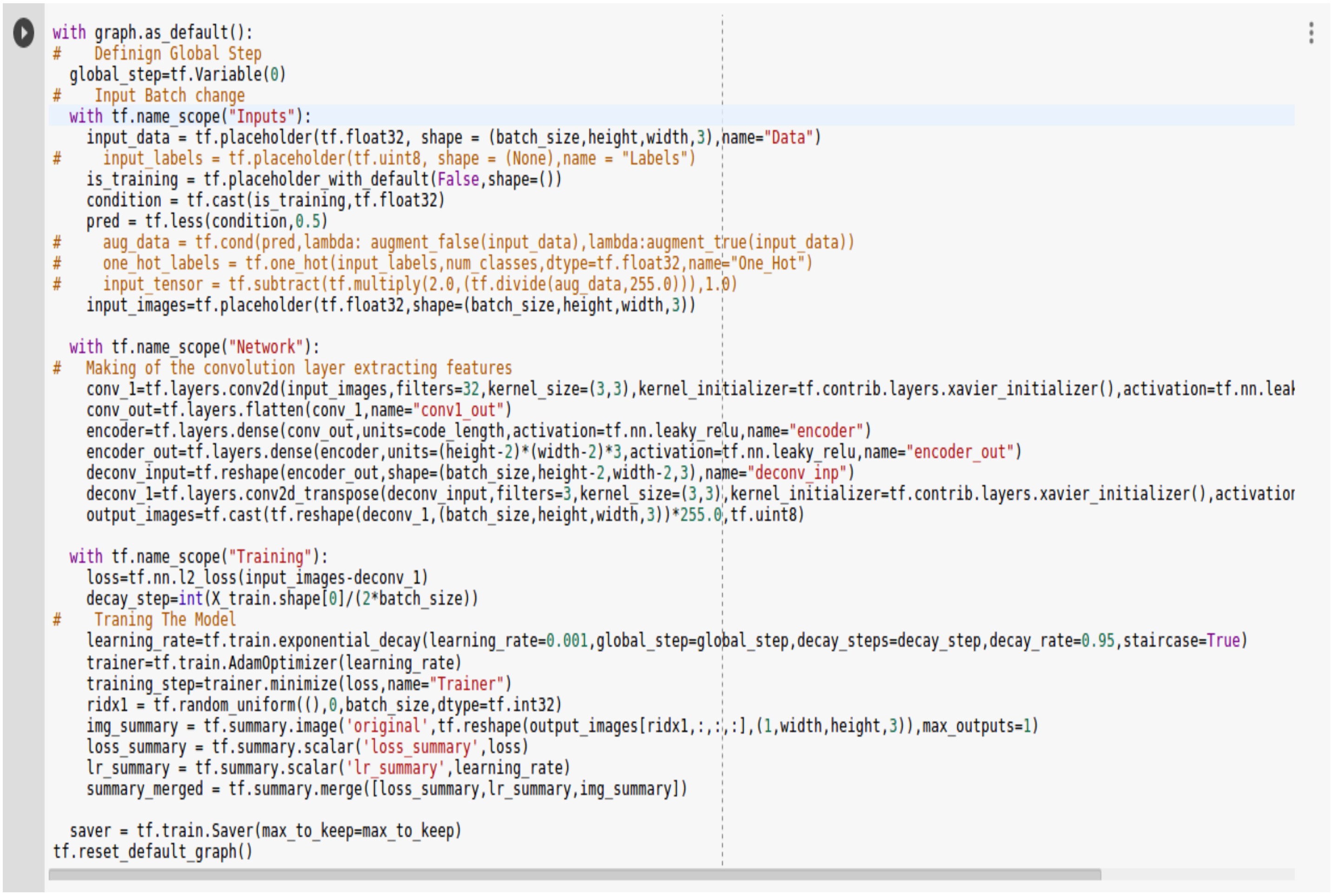
Autoencoders are used for various purposes like dimensionality reduction, denoising image, text removal and spare coding. Auto encoders are generative models. For training on Cifar-10 the following model has been made:

Model Specification:

- Number of input images = 10000
- Batch size for training = 500 images / batch
- No of nodes in hidden layer = 256*3
- Training Epochs = 500
- Learning Rate = Adaptive exponential Decay with initial value 0.01



Coded Graph:

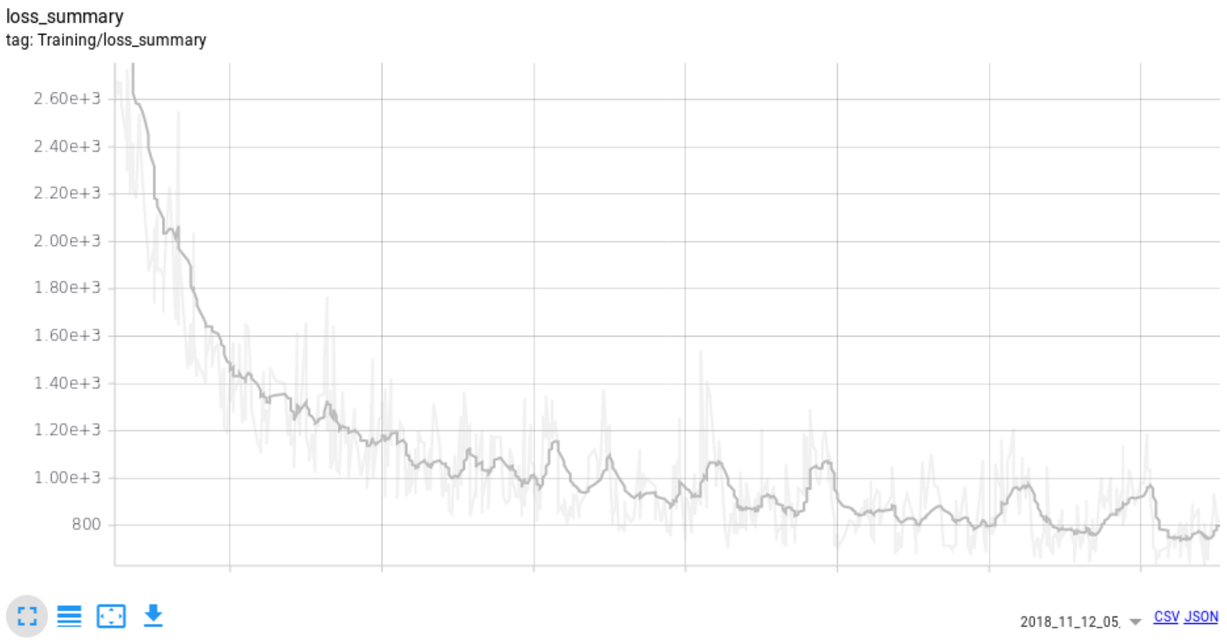


```
with graph.as_default():
#    Definign Global Step
  global_step=tf.Variable(0)
#    Input Batch change
  with tf.name_scope("Inputs"):
    input_data = tf.placeholder(tf.float32, shape = (batch_size,height,width,3),name="Data")
#     input_labels = tf.placeholder(tf.uint8, shape = (None),name = "Labels")
    is_training = tf.placeholder_with_default(False,shape=())
    condition = tf.cast(is_training,tf.float32)
    pred = tf.less(condition,0.5)
#     aug_data = tf.cond(pred,lambda: augment_false(input_data),lambda:augment_true(input_data))
#     one_hot_labels = tf.one_hot(input_labels,num_classes,dtype=tf.float32,name="One_Hot")
#     input_tensor = tf.subtract(tf.multiply(2.0,(tf.divide(aug_data,255.0))),1.0)
    input_images=tf.placeholder(tf.float32,shape=(batch_size,height,width,3))

  with tf.name_scope("Network"):
#   Making of the convolution layer extracting features
    conv_1=tf.layers.conv2d(input_images,filters=32,kernel_size=(3,3),kernel_initializer=tf.contrib.layers.xavier_initializer(),activation=tf.nn.leal
    conv_out=tf.layers.flatten(conv_1,name="conv1_out")
    encoder=tf.layers.dense(conv_out,units=code_length,activation=tf.nn.leaky_relu,name="encoder")
    encoder_out=tf.layers.dense(encoder,units=(height-2)*(width-2)*3,activation=tf.nn.leaky_relu,name="encoder_out")
    deconv_input=tf.reshape(encoder_out,shape=(batch_size,height-2,width-2,3),name="deconv_inp")
    deconv_1=tf.layers.conv2d_transpose(deconv_input,filters=3,kernel_size=(3,3),kernel_initializer=tf.contrib.layers.xavier_initializer(),activatior
    output_images=tf.cast(tf.reshape(deconv_1,(batch_size,height,width,3))*255.0,tf.uint8)

  with tf.name_scope("Training"):
    loss=tf.nn.l2_loss(input_images-deconv_1)
    decay_step=int(X_train.shape[0]/(2*batch_size))
#    Traning The Model
    learning_rate=tf.train.exponential_decay(learning_rate=0.001,global_step=global_step,decay_steps=decay_step,decay_rate=0.95,staircase=True)
    trainer=tf.train.AdamOptimizer(learning_rate)
    training_step=trainer.minimize(loss,name="Trainer")
    ridx1 = tf.random_uniform((),0,batch_size,dtype=tf.int32)
    img_summary = tf.summary.image('original',tf.reshape(output_images[ridx1,:,:,:],(1,width,height,3)),max_outputs=1)
    loss_summary = tf.summary.scalar('loss_summary',loss)
    lr_summary = tf.summary.scalar('lr_summary',learning_rate)
    summary_merged = tf.summary.merge([loss_summary,lr_summary,img_summary])

  saver = tf.train.Saver(max_to_keep=max_to_keep)
tf.reset_default_graph()
```

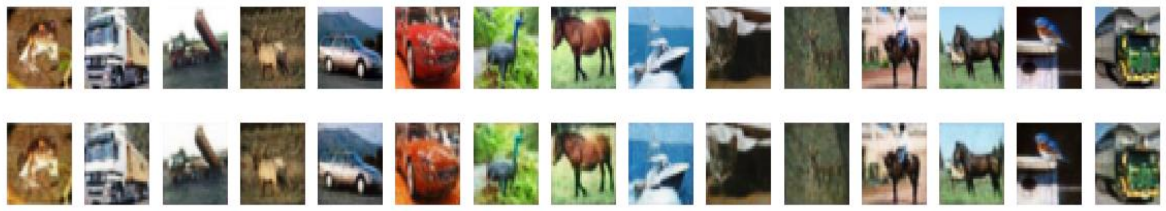
Here First input scope has been defined in which batches has been passed. In the network graph first input has been convolved to get top level features. These inputs have been passed through the fully connected encoder layer which is having 256*3 hidden layer neurons. After that decoded output from reverse of fully connected layer has been formed and the encoder output have been followed by deconvolution is giving the desired output. This is how single layer autoencoder has been made.

The optimizer used was Adam optimizer and the activation function used was leaky relu followed by sigmoid activation in last layer.

The training Graph of L2 loss has been found as:



The output from the training for 500 epochs has been shown below



The top row is the original image and the bottom row is the reconstructed image.

Inferences and Improvements:
We found that as we were increasing the number of hidden nodes in the hidden layer the loss was reducing more which means the previous numbers was not enough to represent the whole dataset.

In place of all sigmoid activation the introduction of relu has improved the performance much.

If we use dense autoencoder we can improve the performance even more.