SmallSEOTools

PLAGIARISM SCAN REPORT

Words	965	Date	June 21,2021
Characters	6732	Excluded URL	

0%	100%	0	51
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

7.1 TYPES OF TESTING
We test the modules of our entire setup to ensure that the modules are performing as they're intended to. We perform a kind of unit testing to evaluate the output of every function in the code.
7.2 TEST CASES AND TEST RESULTS
7.2.1 Data Gathering and Pre-processing
• Test Case 1: Image Scraping test
– Test Case Description:
The module function should scan through the legacy archive only in the described co-ordinates and scrape images that are RGB and of dimensions 256256
– Status: Pass
• Image Colorization:
– RGB Color Space:
* Test Case 2: Pre-processing System: Grayscale Images (Input distribution)
· Test Case Description:
The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions n 64641 where n is the batch size. The output image vector should contain black & white images
· Status: Pass
* Test Case 3: Pre-processing System: Color Images (Target distribution)
· Test Case Description:
The target distribution should contain color counter-parts of the respective input data. The function should output a numpy array
with dimensions n64643 where n is the batch size.
· Status: Pass
– L*a*b Color Space:
* Test Case 2: Pre-processing System: RGB to L*a*b conversion
· Test Case Description:
The pre-processing function takes in a stream of raw data and converts the images into a numpy array with dimensions n 64643 where n is the batch size. The output is an image tensor in L*a*b color space. The L channel is provided for the grayscale input
· Status: Pass
* Test Case 3: Pre-processing System: L*a*b to RGB conversion
· Test Case Description:
The function takes in its input as an image tensor in L*a*b colorspace.

The function should output a numpy array with dimensions
n64643 where n is the batch size and the image
output is converted back to the RGB space
· Status: Pass
• Image Super-resolution:
– Test Case 4: Pre-processing System: Low Resolution Images (Input distribution)
* Test Case Description:
The pre-processing function takes in the raw data and produces low
resolution 64643 images. The output should be a numpy vector
of n dimensions representing the mini-batch size
* Status: Pass
– Test Case 5: Pre-processing System: High Resolution Images (Target
distribution)
* Test Case Description:
The target distribution contains the labelled data that is a high resolution
image of dimensions 2562563. The function produces a
numpy vector of size n where n is the size of mini-batch
* Status: Pass
7.2.2 Model Implementation
• Image Colorization Model
– Test Case 1: Generator network
* Test Case Description:
The generator network is a tensor graph implemented in tensorflow.
The computation graph should have an input layer of dimensions
n64641 which is propagated through till the final layer. The
final layer output should be a tensor of dimensions n64643
where n is the mini-batch size
* Status: Pass
– Test Case 2: Discriminator network
* Test Case Description:
The discriminator network is a tensor graph implemented in tensorflow.
The computation graph should have an input layer of dimensions
n64643 which is propagated through till the final
layer. The final layer output should be a tensor of dimensions
n30301 where n is the mini-batch size
* Status: Pass
• Image Super-resolution Model
– Test Case 1: Generator network
* Test Case Description:
The generator network is a tensor graph. The computation graph
should have an input layer of dimensions n64643. The final
layer output should be a tensor of dimensions n2562563
where n is the mini-batch size
* Status: Pass
– Test Case 2: Discriminator network
* Test Case Description:
The discriminator network is a tensor graph. The computation graph
should have an input layer of dimensions n2562563. The
final layer output should be a tensor of dimensions n1 where n is
the mini-batch size
* Status: Pass
7.2.3 Training Loop
• Test Case 1: Forward propagation
– Test Case Description:
The forward propagation function should run the computation graph over
all the training samples and calculate losses of the generator and discriminator.
The loss should be calculated according to the GAN objective
function and final output should be the cost of the entire training epoch.
– Status: Pass
• Test Case 2: Backward propagation
– Test Case Description:
The backward propagation should calculate gradients of the objective
function based on the cost calculated during the forward propagation.
The gradients should be used to update the trainable variables of the network
using gradient descent with momentum and weighted average sum,

i.e. Adam.
– Status: Pass
• Test Case 3: Callbacks and intermediate results logging
– Test Case Description:
While training the model, the callback functionality should keep track of the intermediate training results. It should perform the following:
A csv file containing history of the entire training session
Save the model weights on a best performance basis by keeping track of the earlier losses
Keep track of the convergence and stop the model training if the model converges early
– Status: Pass
7.2.4 Evaluation system
• Model performance evaluation
– Test Case 1: Model history plotting
* Test Case Description:
The function should plot the model losses throughout the training loop and present a graph of the said plot
* Status: Pass
– Test Case 2: Output plotting
* Test Case Description:
The function should take in the predictions by the model and save the images with their respective ground truth images for evaluation
* Status: Pass
– Test Case 3: Model evaluation (L1 norm)
* Test Case Description:
The function should take in two numpy vectors consisting of predicted values and ground truths respectively. The function output should be a single number consisting of the L1 norm (Manhattan distance) between the target and predictions
* Status: Pass

Sources	Similarity