SmallSEOTools

PLAGIARISM SCAN REPORT

Words	542	Date	June 21,2021
Characters	3682	Excluded URL	

0%	100%	0	37
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

To gather data, the primary module implemented was a data scraper written in node.js which uses puppeteer(headless chrome) to scrape the entire HLA(Hubble Legacy Archive) with the described RA and Dec co-ordinates of the night sky. The module ensured that, out of the entire data available in the Legacy Archive, a certain format of images would be downloaded and stored appropriately. After gathering the data, it was apparent that there were many other processed images present on the Hubble main and other citizen science projects conducted with Hubble Space Telescope. To scrape these, a python script using selenium was written so as to efficiently scrape processed images which had a minimum dimensions of 2562563. With the data gathered, the main components, i.e. the models were built.
The project entails a standard set of GAN modules, broadly, a data pre-processing pipeline, a training pipeline, intermediate state logging and a testing pipeline.
• Data Pre-processing
– The data pre-processing pipeline is an important part of the data pipeline that feeds the data to the training pipeline
– The images, being too large to be loaded into the memory simultaneously, had to be loaded through a data generator
– We implement an instance of Keras Image Data Generator to convert the raw data into training data
– We create two data generators, one for generating gray scale images and the other for generating subsequent RGB image
– We create another data pre-processor that converts the given RGB image into L*a*b color space
– The images are then resized to a dimension of 6464
– We get a vector of image batches which will be the input to the training pipeline
• Training Pipeline
– The training pipeline consists of multiple phases
– Initially, we build the basic neural networks for generator and discriminator networks, i.e. a tensor graph which will function as a computation graph for forward and backward propagation
– The next step is selection of an optimizing algorithm
– After selection of optimizer, a training pipeline is implemented which includes running a session of forward and backward propagation throughout the computation graph and optimizing the trainable variables of the networks
• Intermediate state logging
– Deep Learning models use up a lot of computational resources and are time consuming to train

– Thus, it is essential to set up methods that will extract intermediate states of the model while it is still in the training loop
– These methods are often termed as ’callbacks’ and we implement different callbacks to ensure availability of training history, weights and intermediate results during documentation
– A csv logger logs the metrics at the end of each epoch into a csv file for model evaluation
– The model weights are saved if the loss metric improves than the earlier epoch. We implement a mechanism for early stopping where the training loop exits if the model appears to have converged
• Testing pipeline
– Model testing is performed on a testing dataset that is different than the original dataset
– Testing pipeline is implemented by using the saved weights of the trained model and the images are processed through the entire dataset
– We evaluate the images qualitatively as well as quantitatively using visual inspection and distance metrics

Sources	Similarity