# Project Milestones [xrelab]

## Image Classification and Comparison of Large Image datasets using Batch Normalization Vs. Group Normalization

### Project Configuration

* Requirements.txt for the project is @ <https://github.com/sananand007/Xrelab/tree/master/final-project>
* OS – Windows 10, 64bit Environment used – Anaconda Prompt
* Python-3.6.6 with Tensorflow 1.11 🡪 Tensorflow 1.11 with the latest Tensorboard will only work with python 3.6.6
* Remote Environment used – [www.paperspace.com](http://www.paperspace.com) 🡪 This is to run Full Training Data, which cannot be run using just the CPU
* While Using the CPU version in paperspace.com , you can only use the sse2 version and not the avx2 as there are issue with it, find out the sse2 version of tensorboard that supports your case for CPU .

**1st week milestone**

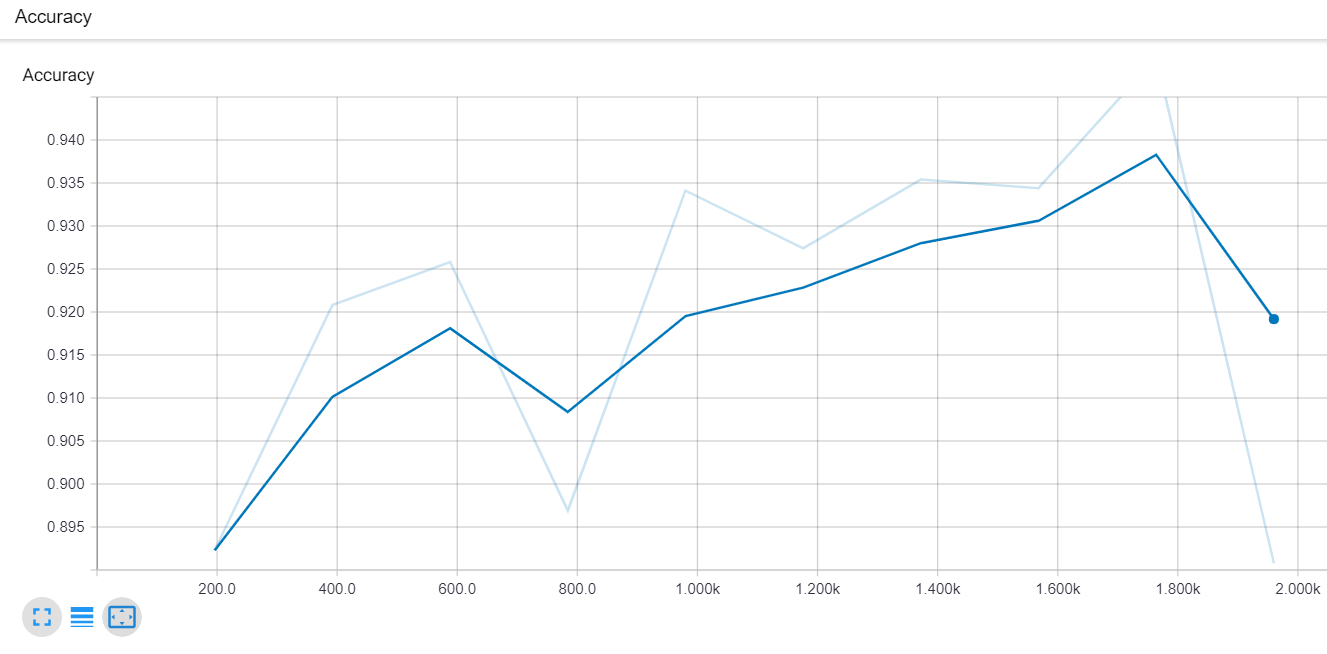
* Use and Analyze the DeepSat (SAT-6) Airborne Dataset 🡪 Done
  + Analyze the dataset, load the dataset and create training and test sets respectively
  + Whole dataset consists of Training set of 324000 images , each being 28x28 , and 4 channels
  + Whole dataset consists of Test set of 81000 images , each being 28x28 , and 4 channels
  + Plot some images to check the categories
  + Make sure the labels are One-hot encoded
* Build a Simple CNN Pipeline using tensorflow and tensorboard to Train 🡪 Done
  + I use a CNN pipeline as below to train the simple 2-Layer model

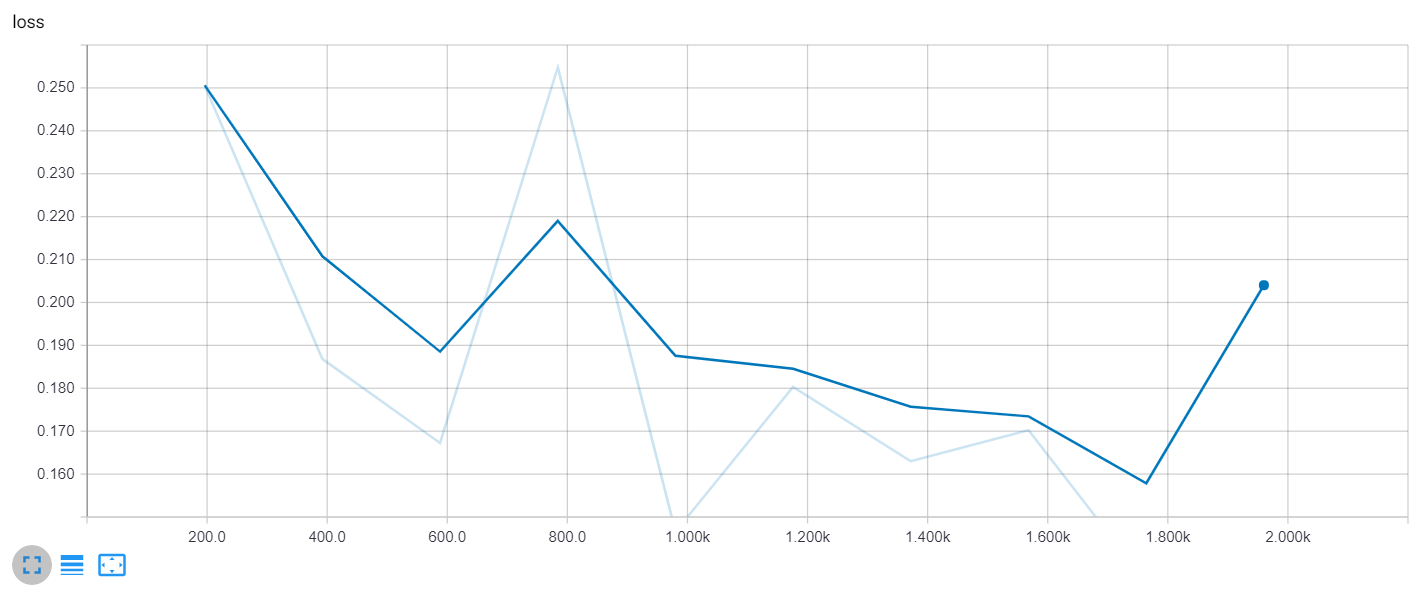
**Conv1 🡪 Max-pool 🡪 Conv2 🡪 Max-pool 🡪 flatten 🡪 dense 1🡪 dropout 🡪 dense2 🡪 Logits**

* + Get the Loss, Accuracy, Precision and Recall metrics
  + Log Summaries using tensorflow logging api so that we can visualize using tensorboard
* Log Summaries and Check Using Tensorboard for Performance of the Model 🡪 Done
  + Current Model Plots are as below after some hyperparameter tuning , Plots are made using Tensorboard currently, In the future I plan to use matplotlib
  + I see the Maximum accuracy for the model reached upto nearly 94% based on runs

Parameters used for the Model

* learning\_rate=1e-04
* decay\_steps=10000
* decay\_rate=0.96
* staircase=True
* optimizer = Gradient Optimizer
* No Batch Normalization used
* training batch size = 512
* Currently only use 1/3rd of the images for trainings, I plan to use the entire set as I add on top of the model for later
* Steps = 2000 steps, with each 200 steps after which we calculate the Loss and evaluate the model on the test set, hence total of 10 Iterations of 200 steps each





For complete Jupyter Notebook

* [Github](https://github.com/sananand007/Xrelab/tree/master/final-project)
* [Dataset](https://www.kaggle.com/crawford/deepsat-sat6)

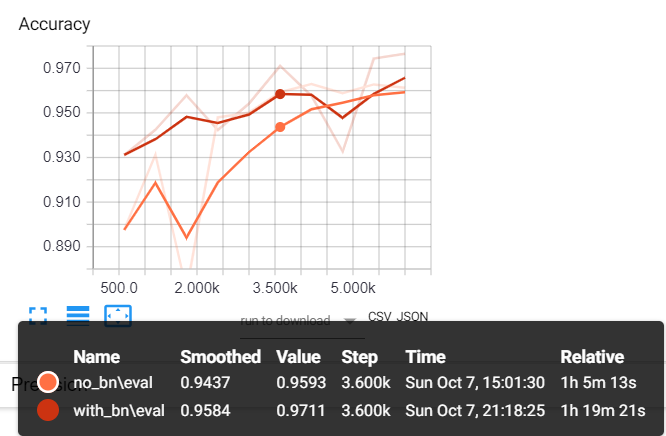
**2nd week milestone**

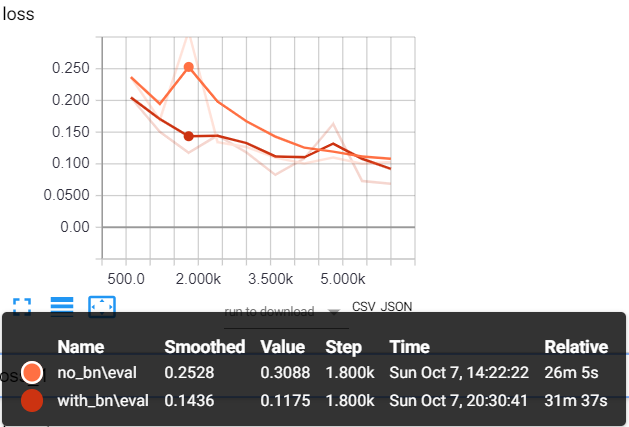
* Use Batch Normalization on the dataset, and compare while using Group Normalization based on recent research
* Start comparing performance with other large datasets using the same Deep learning model
  + Without Batch Norm Vs. With Batch Norm Vs. With Group Norm

### Analysis of dataset with Vs. w/o Batch-Norm, Plots are represented using Tensorboard and plots are during the evaluation step

### Batch Normalization only used in the last layer currently of the Model

* Amount of training set data used = 324000 images = 100%
* Amount of test set data used = 81000 images = 100%
* 6000 training Steps performed and below figure shows comparison with batch\_norm Vs. without batch\_norm
* Accuracy is considerable better with\_bn and reaches a Max of ~0.97, the trend shows the Accuracy is also moving upwards with\_bn and coming to a flat slope without\_bn
* Loss is considerably converging faster with batch normalization, at 1800 global-step we can see that with batch norm, we are at a loss that is almost half of compared to without Batch normalization
* Latest Tensorflow code @ <https://github.com/sananand007/Xrelab/blob/master/final-project/Paperspace/Image%20Classification%20of%20Large%20Image%20Datasets%20Techniques%20Using%20Novel%20Methods.ipynb>





Summary of curves with Batch Normalization

