Argonne Summer 2020

Rick Nueve: Update Week 4 (Data Loader and WeatherNet)

Overview

- For ten weeks, I (Rick Nueve) am an intern at Argonne National Lab under the SAGE project.
- **MISSION STATEMENT**: My primary tasks are to design a Deep Learning model that uses images and FLIR images from a node, have the model be able to run on a node, and also to write a tutorial explaining to students how to make their programs be able to run on the nodes.

TimeLine

Done

TODO

Week: 1-3	Week: 4	Week: 5	Week: 6-7	Week: 8-9	Week: 10
• Learn about SAGE • Training • Make dataset	Make data loader and initial WeatherNet model.	Revise model to improve performance.	Make model plugin/edge friendly and write tutorial.	Work on paper and package model for deployment/git hub.	Finish poster and present.

What was done between 11-17th of June?

1. Made ImgSeqGenerator
2. Version 1 of WeatherNet was made
3. So does it work?

Tensorflow's non-existent documentation for the data loader

1. ImgSeq-Generator

My feeble mind

1. ImgSeqGenerator

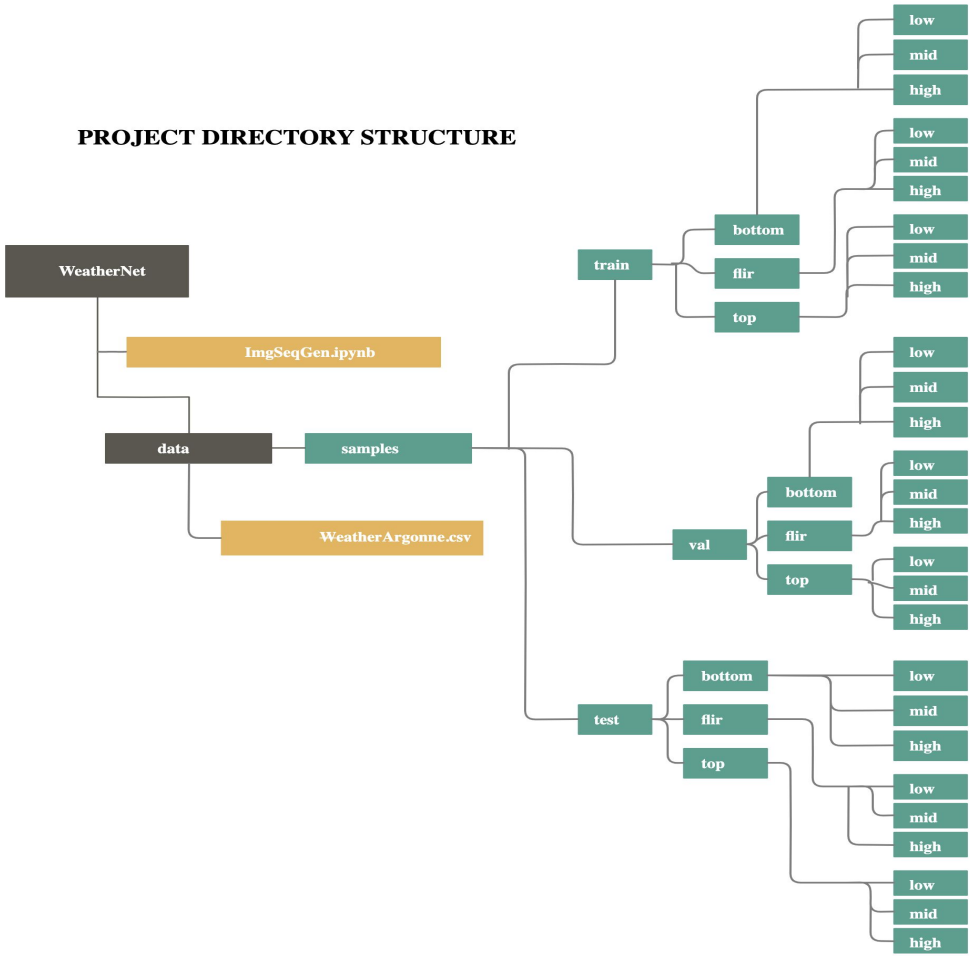
- A data loader in Tensorflow was built to load in data in batches to memory to prevent memory from running out.
- The data loader is in the file ImgSeqGen.ipynb.
- Examples of the data loader are demonstrated in ImgSeqGen.ipynb.

1. ImgSeqGenerator

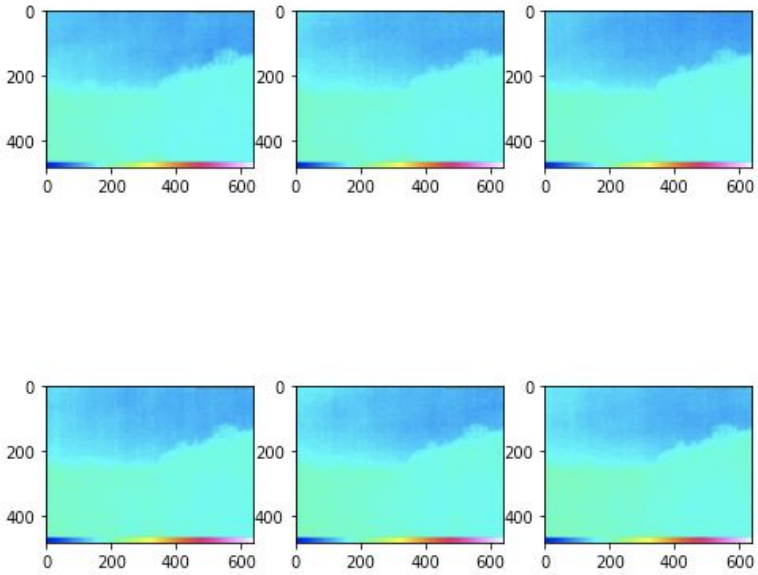
```
train_gen = ImageDataGenerator(rescale = 1./255)

n_frames = 6
n_batches = 1

train_generator=generate_mult_seq(generator=train_gen,
                                  dir1= train_flir_path,
                                  dir2= train_bottom_path,
                                  dir3= train_top_path,
                                  n_frames = n_frames,
                                  n_batches = n_batches,
                                  img_height=480,
                                  img_width=640)
```

```
X,y = train_generator.__next__()
X_flir, X_bottom ,X_top = X[0], X[1], X[2]
y_flir, y_bottom, y_top = y[0], y[1], y[2]
```

1. ImgSeq-Generator



1. ImgSeqGenerator



1. ImgSeqGenerator

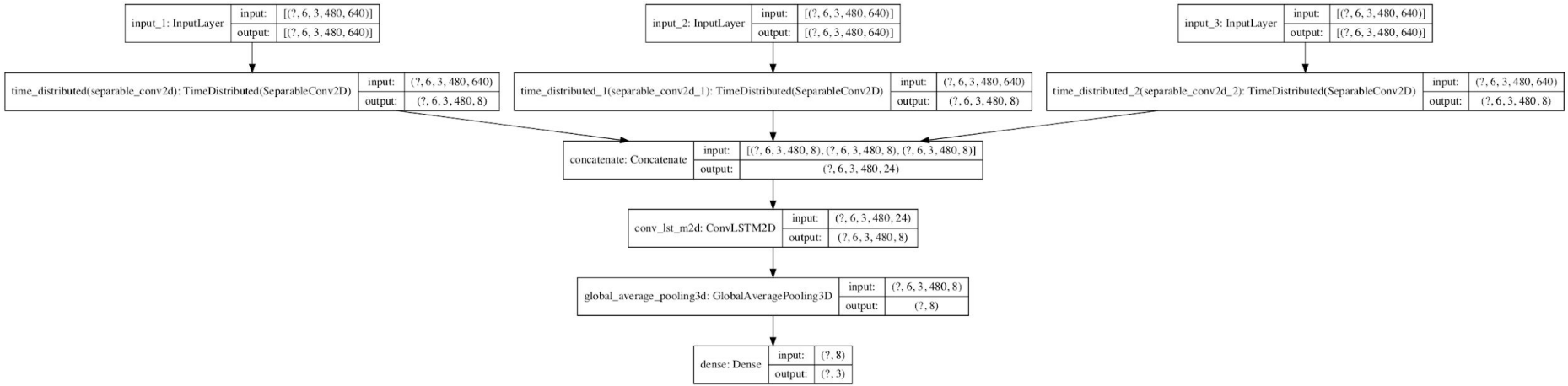
```
batch 0
Memory percent usage 69.8
(4, 6, 3, 480, 640) (4, 6, 3, 480, 640) (4, 6, 3, 480, 640)
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
batch 1
Memory percent usage 71.4
(4, 6, 3, 480, 640) (4, 6, 3, 480, 640) (4, 6, 3, 480, 640)
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
batch 2
Memory percent usage 71.0
(4, 6, 3, 480, 640) (4, 6, 3, 480, 640) (4, 6, 3, 480, 640)
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
batch 3
Memory percent usage 72.5
```

2. WeatherNet

1. Inputs: Sequence of six images from the previous hour from each of the three cameras: flir, top, and bottom facing.
2. Output: Inferes next hours average solar radiation (categorically: low, medium, or high based on three quantiles from historical data)

2. WeatherNet

- Has 68,000 parameters currently.



3. So does it work?

- Currently, I do not know.
- The data generator is not cooperating with the model, so I am going ahead and designing a whole generator class so it can be fully compatible with the model
- I will also need to set up a chameleon account so that I can get gpu access to start grid-searching different architectures for optimal performance.

Paper update

- The abstract was added (have received some feedback, will be updating shortly).
- Due to the complexity of atmospheric systems, the task of nowcasting (short term weather forecasting) is a challenging problem. However, with the rise of Deep Learning, scientists are exploring ways to use Neural Networks to perform nowcasting. In this paper, the authors attempt to use Deep Learning to predict future solar radiation values uniquely using a remote camera systems on the ground in a single location. To our best knowledge through our literature review, this task has not been performed or documented where nowcasting is attempted with only using cameras placed at a single location. Due to the uniqueness of this task, we are proposing to call this problem pocket-casting. The remote camera system consists of three cameras: a top facing camera, a leveled facing camera, and a FLIR (forward-looking infrared) camera. These three camera's photos from the current hour are fed into a ConvLSTM variation, which then predicts categorically (low, medium, or high) the amount of solar radiation for the following hour. To perform the nowcasting on the remote camera system, Edge Computing is employed. The inference is performed on the the Edge, and the predicted values of the future solar radiation are sent back to a server. To perform the Edge Computing, SAGE, a Cyber-infrastructure for Edge computing, is used. Leveraging Sage Cyber-infrastructure allows the code to be run at the edge enabling a dramatic reduction in needed bandwidth. By employing SAGE, only the predicted solar radiation values, which are only $1.92 e^{-7}$ GB, are sent back to the server. Otherwise, each day's worth of photos, which is 0.6912 GB, would have to be sent back to the server for inference. Overall, the model was able to predict solar radiation values of the following hour with "# TBD #" percent accuracy.

Questions from the audience?

How to contact me

I can be reached at enueve@anl.gov or at enriquenueve9@gmail.com.

Also, check out my portfolio at: https://realestrick-c137.github.io/EnriqueNueve.github.io/index.html# .