Readme

# Setup

## Libraries needed

The commands are run in an anaconda environment.

TensorFlow: pip install tensorflow\_gpu

NumPy: pip install numpy

SciPy: pip install scipy==1.1.0

## Preparing the Dataset

: number of input frames

: number of output frames

We need to create text files that will contain the path to the frames.

In total, we need to create number of files.

Let the total number of frames in the dataset be n.

For example:

The first text file will contain names of frames from 1 to . The second text file will contain paths of frames from 2 to . The following code can be run in the dataset folder to create these text files:



# Major Files

1. Voxel\_flow\_train.py: The function creates an object of type Dataset for each text file and sends it to function train().

* **train()**: This is the function that trains the model and saves all the results. If training on a different dataset one can just change the , , , and parameters. We make an object of and initialize the optimizer. The function reads the text files and stores them in different variables. The training then starts, and we load the images in batches for each iteration. The batch of images is passed to the variable, after which we calculate the losses and update the parameters. The loss is shown as output after every ten iterations, and the sample output, model, and results.csv is saved after every 100 iterations.

1. Voxel\_flow\_model.py:
   * **\_build\_model()**: This the function which initializes all the model parameters and contains the logic. The input is passed through a CNN and then through bilinear interpolation. We return the result as predictions.
   * **loss()**: The function is used to choose the loss function after importing it from . The and needs to be sent as parameters.
2. Dataset.py: Set the DATA\_PATH\_BASE variable to the dataset folder path.
3. Loss\_utils.py: It contains all the loss functions. The user can import any of these in and pass the predictions and targets as parameters.
4. Image\_utils.py: It contains the functions to load and save images.

# Changes to be made for different parameters

--batch\_size: The number of samples in each batch.

--initial\_learning\_rate: Initial learning rate.

--num\_in: Number of input frames.

--num\_out: Number of output frames.

--max\_steps: Number of batches to run.

--train\_image\_dir: Directory where to output images.

--train\_dir: Directory where to write event logs and checkpoint.

1. Batch\_size: Change the batch\_size in when the grid is being initialized.



1. Changing number of inputs(num\_in) or outputs(num\_out):
   1. Dataset: The text files containing the paths of images needs to be changed.
   2. Function definitions: The number of parameters in the function definitions of the train/test needs to be changed. The number of parameters is equal to . For e.g. if num\_in=4 and num\_out=2, function definition for train would be:



1. Changing the number of iterations or learning rate: No changes needed.

**CAUTION**: The destination for and should be changed for every setup or else the new results will overwrite the previous ones.

# Running the script

Via Terminal:



Via Jupyter Notebook:

