New ML Approaches for Nowcasting of Global Auroral Particle Precipitation

Jack Ziegler and Ryan McGranaghan

Atmospheric & Space Technology Research Associates

May 19, 2021

Presented at "Applications of Statistical Methods and Machine Learning in the Space Sciences"

Work sponsored by DARPA HEARTBEAT Program

(Heliosphere to Earth Atmosphere Rendering Through Building Excellent Artificial-Intelligence Training)

Outline

- Past work:
  - Energy Flux
  - ML, Ovation
- ML model improvements
  - Dropout, more layers, custom loss function
- Auroral Boundary/Region Models
  - Regions, model separation, multi-task models
- Number Flux Modeling
  - Total and Channel based
- Conv2d_Transpose Modeling
- New Database
  - Selection of Time scales
- New Ideas

Overview

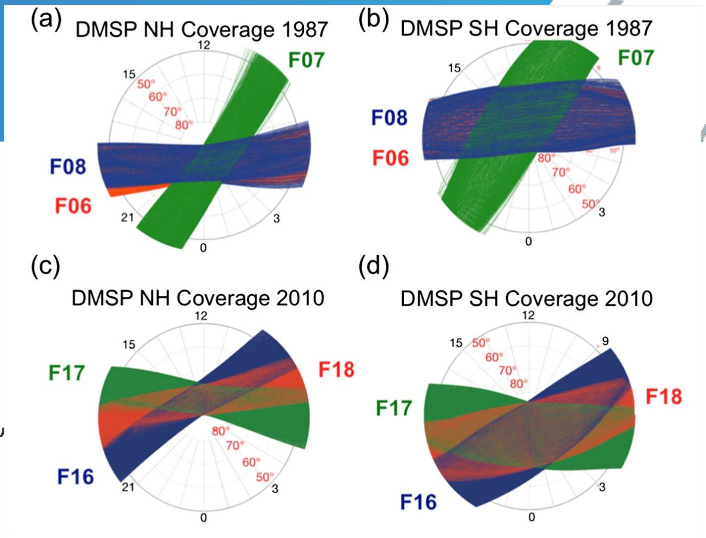
- Total Electron Energy(Number) Flux

  Full problem:

  Precipitation(time, Mlat, Mlocaltime)

  = *function*[mlat, mlocaltime, Magnetic_indices(time),

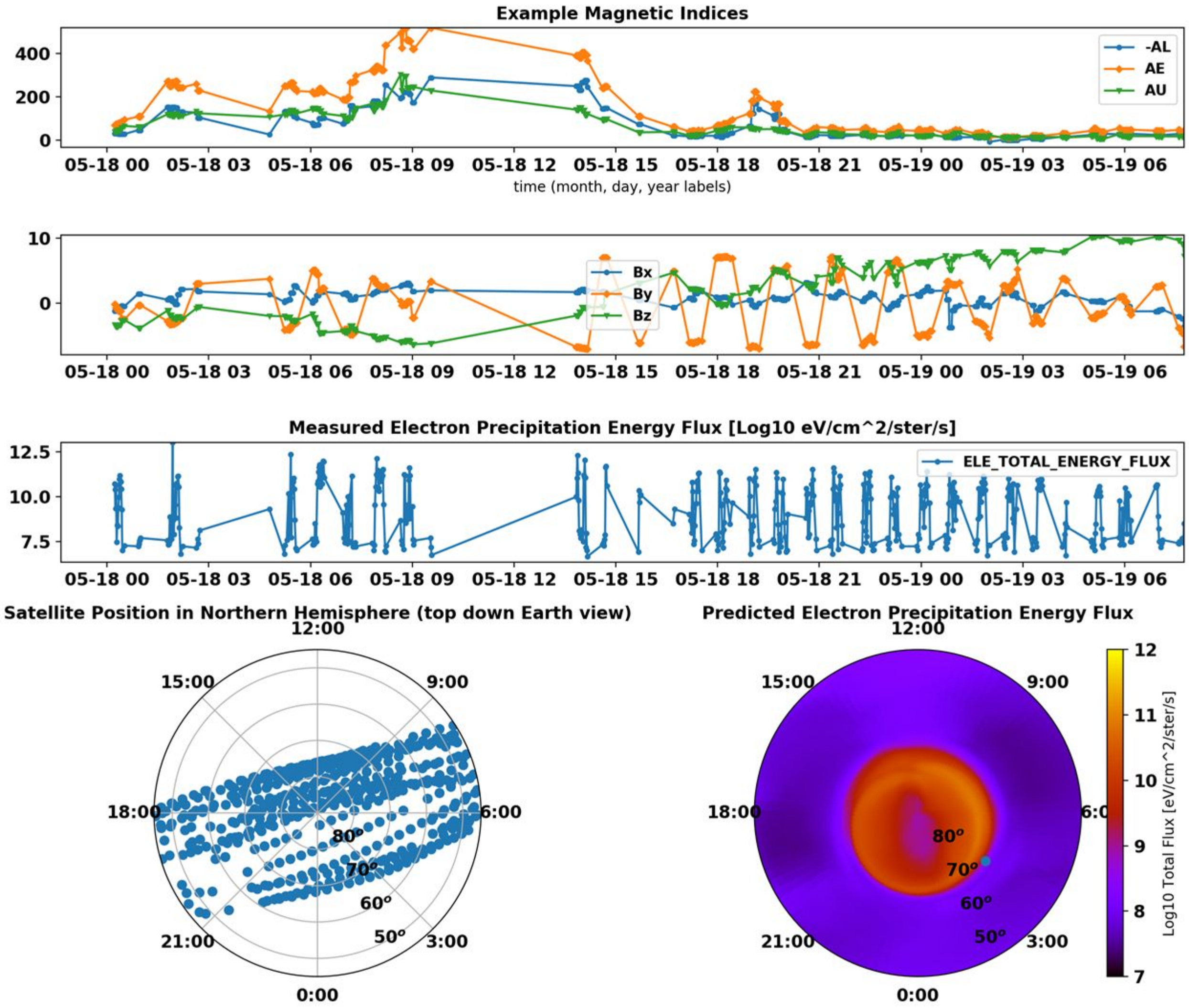
  solar_wind(time)]



Satellites from "multiple generations of sensors" from 1987 to 2018. 150 total "satellite years"

- DMSP Measurements
  - ML Modeling Challenges
  - Regression instead of time-series

- Inputs and outputs
  - Inputs: M_Lat, M_Local_time, date and Magnetic indice history
    - (5 min cadence)
  - Outputs: Total Energy flux at current time as function of MLat and MLT
    - (using 1 min cadence (sub-sampled, not averaged), 1 second cadence available)


Example Magnetic Indices
-AL
AE
AU
time (month, day, year labels)
Bx
By
Bz
Measured Electron Precipitation Energy Flux [Log10 eV/cm^2/ster/s]
ELE_TOTAL_ENERGY_FLUX
Satellite Position in Northern Hemisphere (top down Earth view)
Predicted Electron Precipitation Energy Flux
12:00
15:00
9:00
18:00
6:00
21:00
3:00
0:00
80°
70°
60°
50°
Log10 Total Flux [eV/cm^2/ster/s]

Past Work

PrecipNet

- Solves 2D time series problem as a regression problem with the target transformed to a log10 scale

For HEARTBEAT System: We use only the input features that the HB system provides.

- These are our 'existing HB low energy precipitation models' - i.e., the ones that will be integrated.

HEARTBEAT'S 33 Inputs:

'SC_AACGM_LAT','SC_AACGM_LTIME', 'ID_SC', 'sin_ut',
'cos_ut', 'sin_doy', 'cos_doy', 'sin_SC_AACGM_LTIME', 'cos_SC_AACGM_LTIME',

'F107', 'AE', 'AL', 'AU', 'SymH',
'F107_6hr', 'AE_6hr', 'AL_6hr', 'AU_6hr', 'SymH_6hr',
'F107_5hr', 'AE_5hr', 'AL_5hr', 'AU_5hr', 'SymH_5hr',
'F107_3hr', 'AE_3hr', 'AL_3hr', 'AU_3hr', 'SymH_3hr',
'F107_1hr', 'AE_1hr', 'AL_1hr', 'AU_1hr', 'SymH_1hr'

ML Model Improvements

New Model:

- Twice as many and larger layers overall
- Massive increase in batch size to 32,768
- Dropout layer
- GPU usage
- Layer size testing ->
    - 2x larger than input, 50% dropout, smaller layer, larger layer, and then smaller layers

Custom loss function for tail

8 Hidden layers:

256 node dense

50% dropout layer

64 node dense

32 node dense

256 node dense

1024 node dense

256 node dense

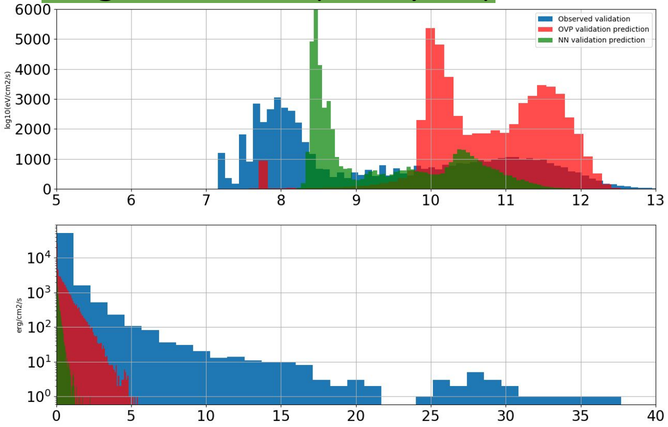
32 node dense

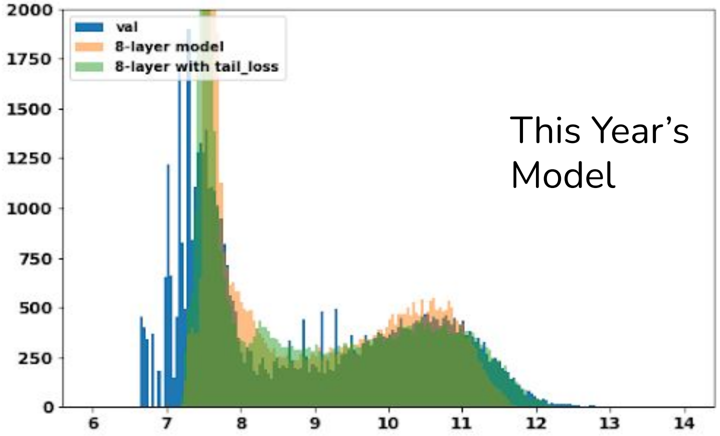
4 node dense

Output layer:

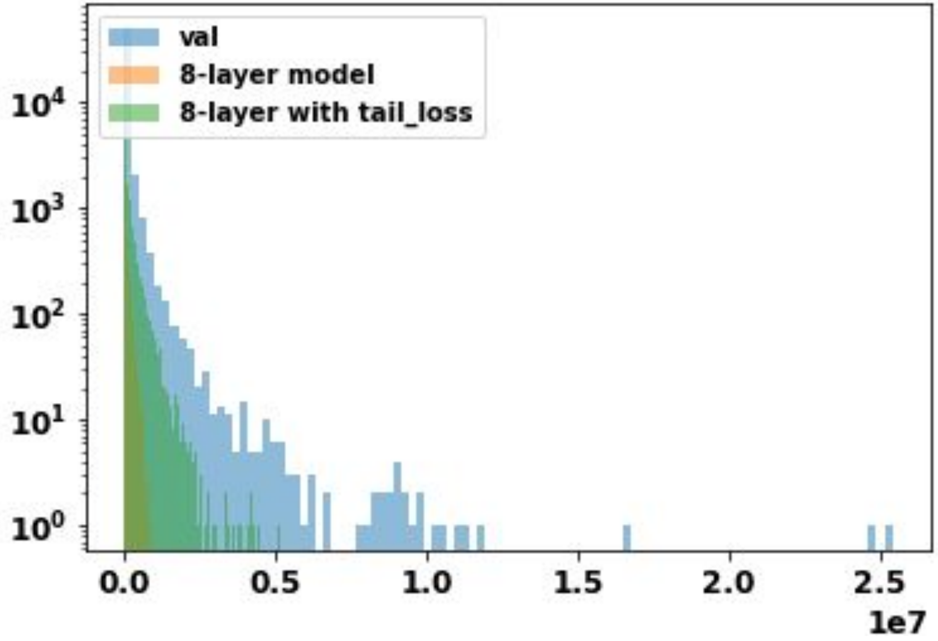
1 node dense

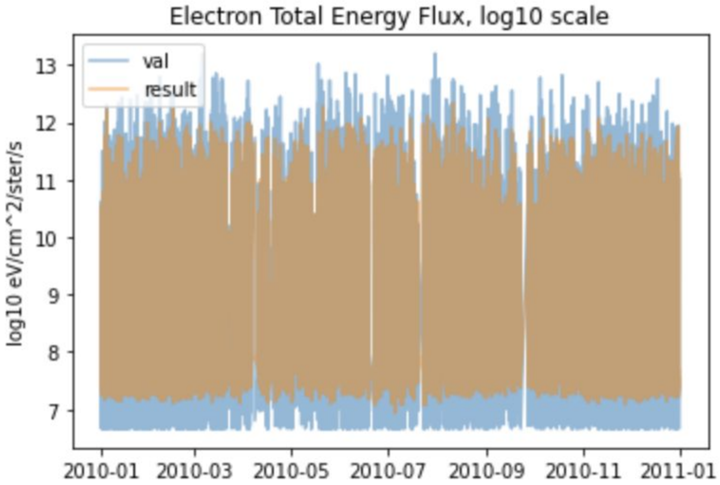
Original Model (Last year)

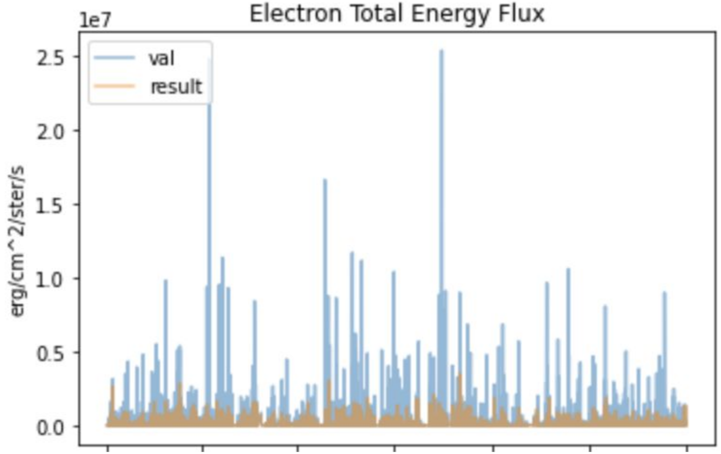




This Year's Model







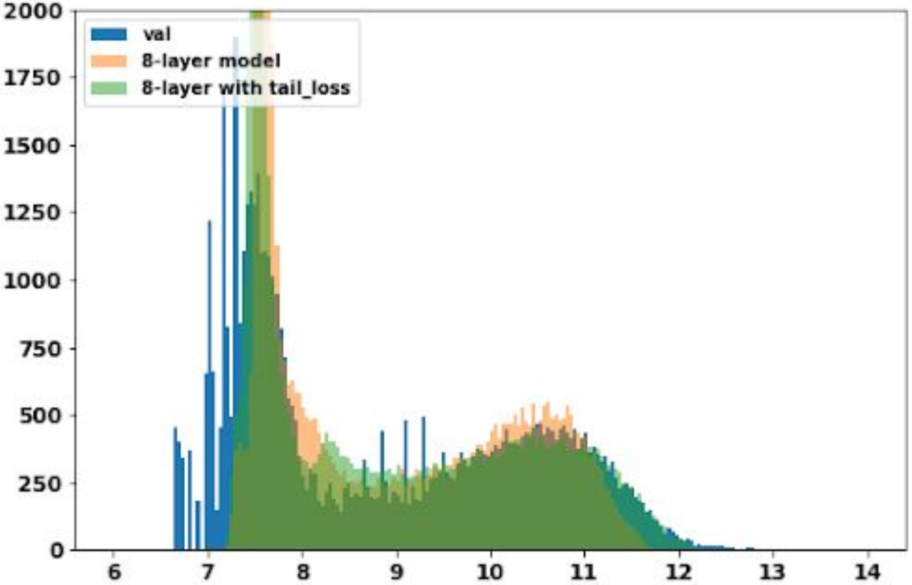
Tail loss function

$$\left[\frac{1}{N}\sum_{i=1}^{N}|obs - pred|\right]^{2} * \left[1 + \sum_{j}^{M}\gamma_j\right]$$

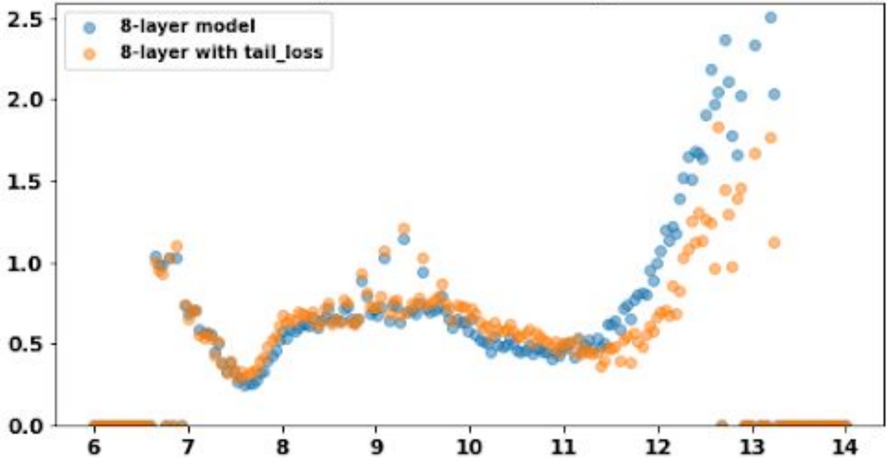
Where $\gamma_j(obs, pred) = w \quad \text{if obs} > \text{tail limit} \quad \& \quad \text{pred} < \text{tail limit}$

$\gamma_j(obs, pred) = 0 \quad \text{otherwise}$

Log10 Scale



val
8-layer model
8-layer with tail_loss
2000
1750
1500
1250
1000
750
500
250
0
6
7
8
9
10
11
12
13
14

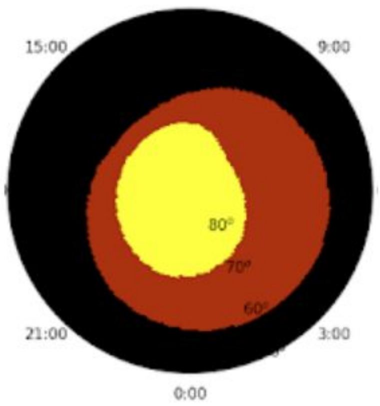


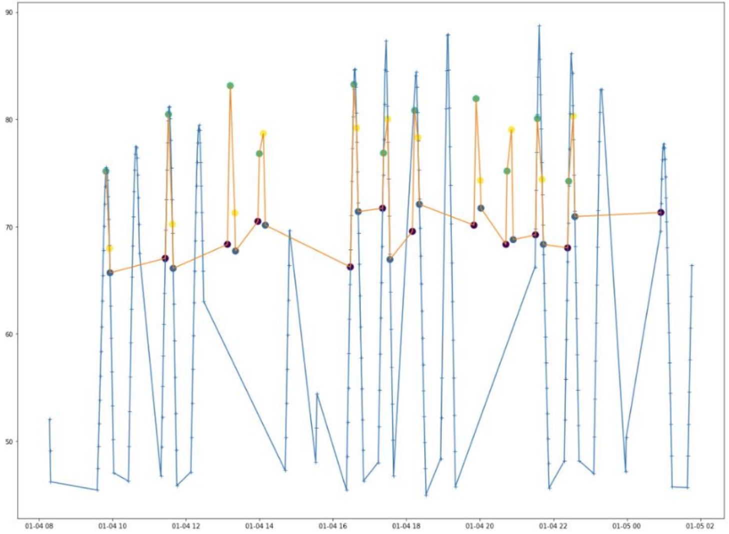
Average Validation Error over target Bins
8-layer model
8-layer with tail_loss
2.5
2.0
1.5
1.0
0.5
0.0
6
7
8
9
10
11
12
13
14

Auroral Boundaries

- 3 class classifier
  - (90% accurate over 45 to 90 M_lat)
- Extended to a three model method
  - (one model for each region)
- Also a dual classifier and regression model
  - 6 outputs,(3 classes and 3 regions)
    - Only the predicted region is used for final result
    - Combined loss function

Region	Train samples	Test samples	Test MSE
Equatorial	925,990	26,821	0.45
Auroral	496,715	14,856	0.55
Polar	415,578	13,533	0.84



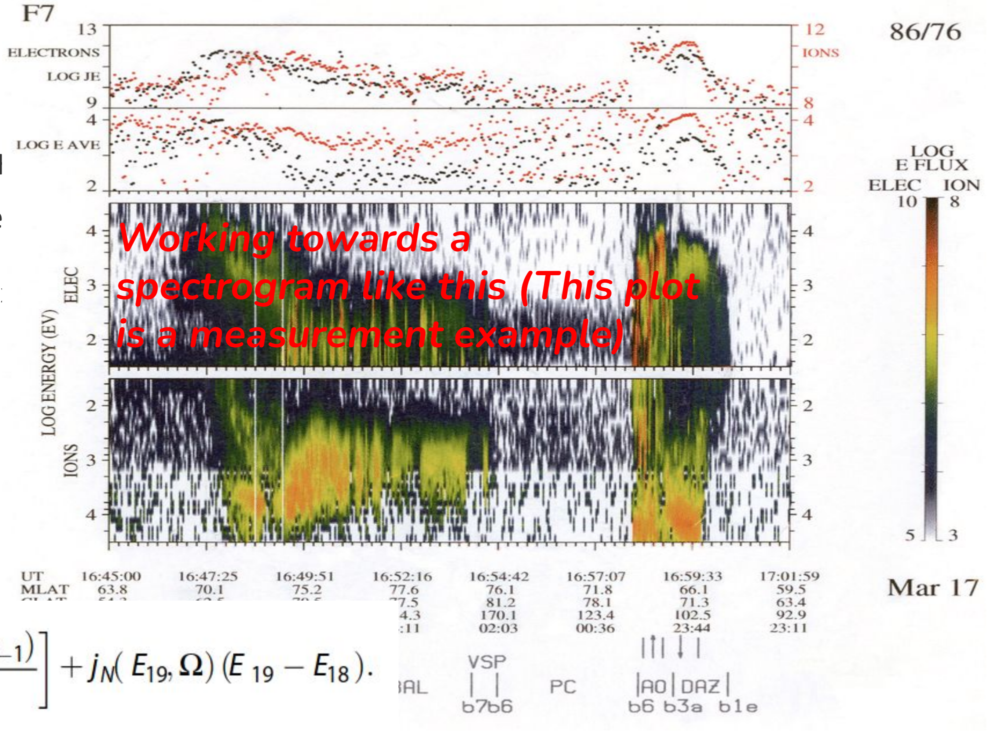


Number Flux Modeling

- Calculated from “electron counts” from 19 total energy channels
  - Each spacecraft has a different “geometric factor” thougl
- Total Number Flux is a scalar representation of all these values
  - Calculated in a way that is very similar to total energy flu

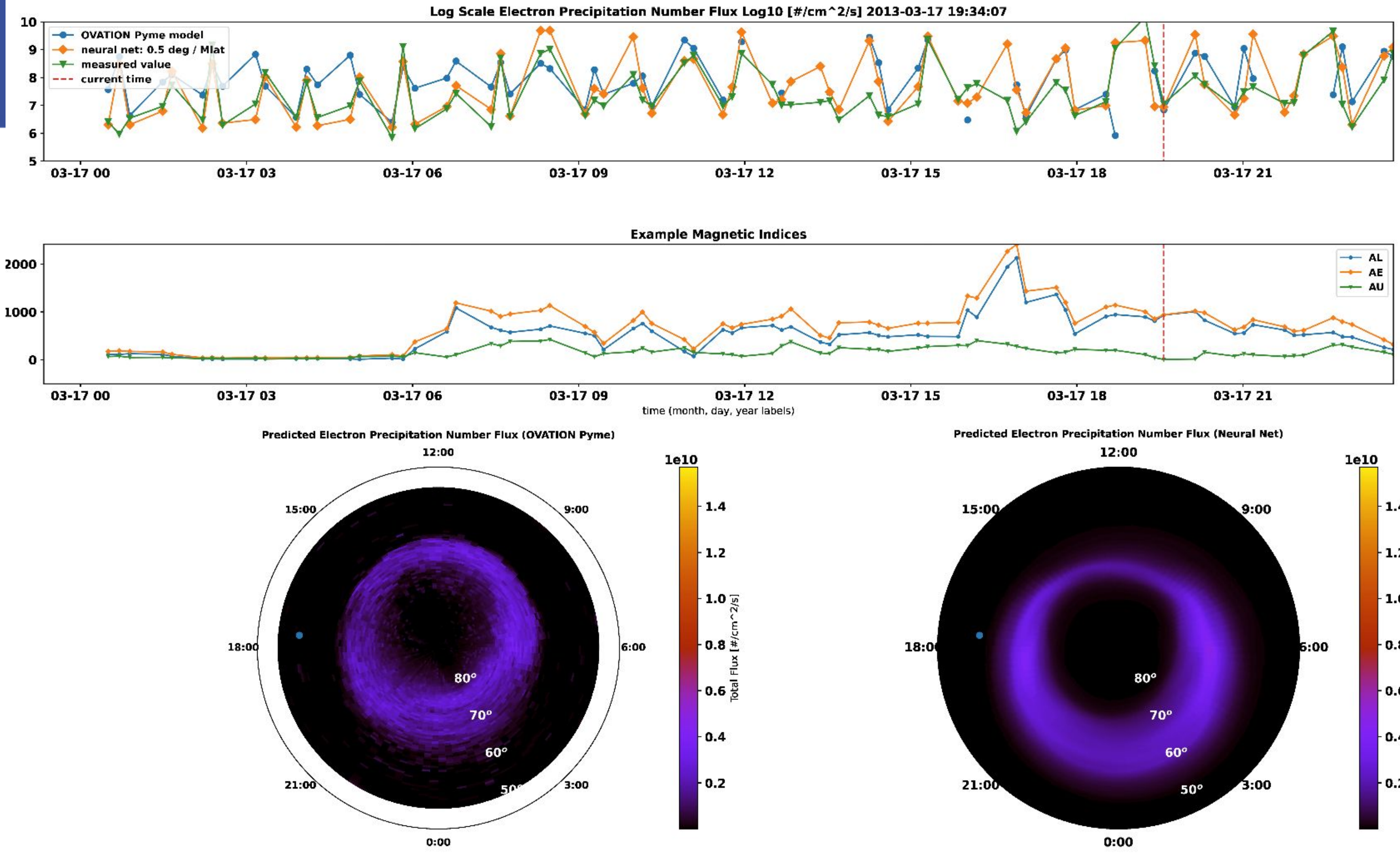
We have models for Total number flux and a 19 Channel output number flux

Uses same network structure as total energy flux

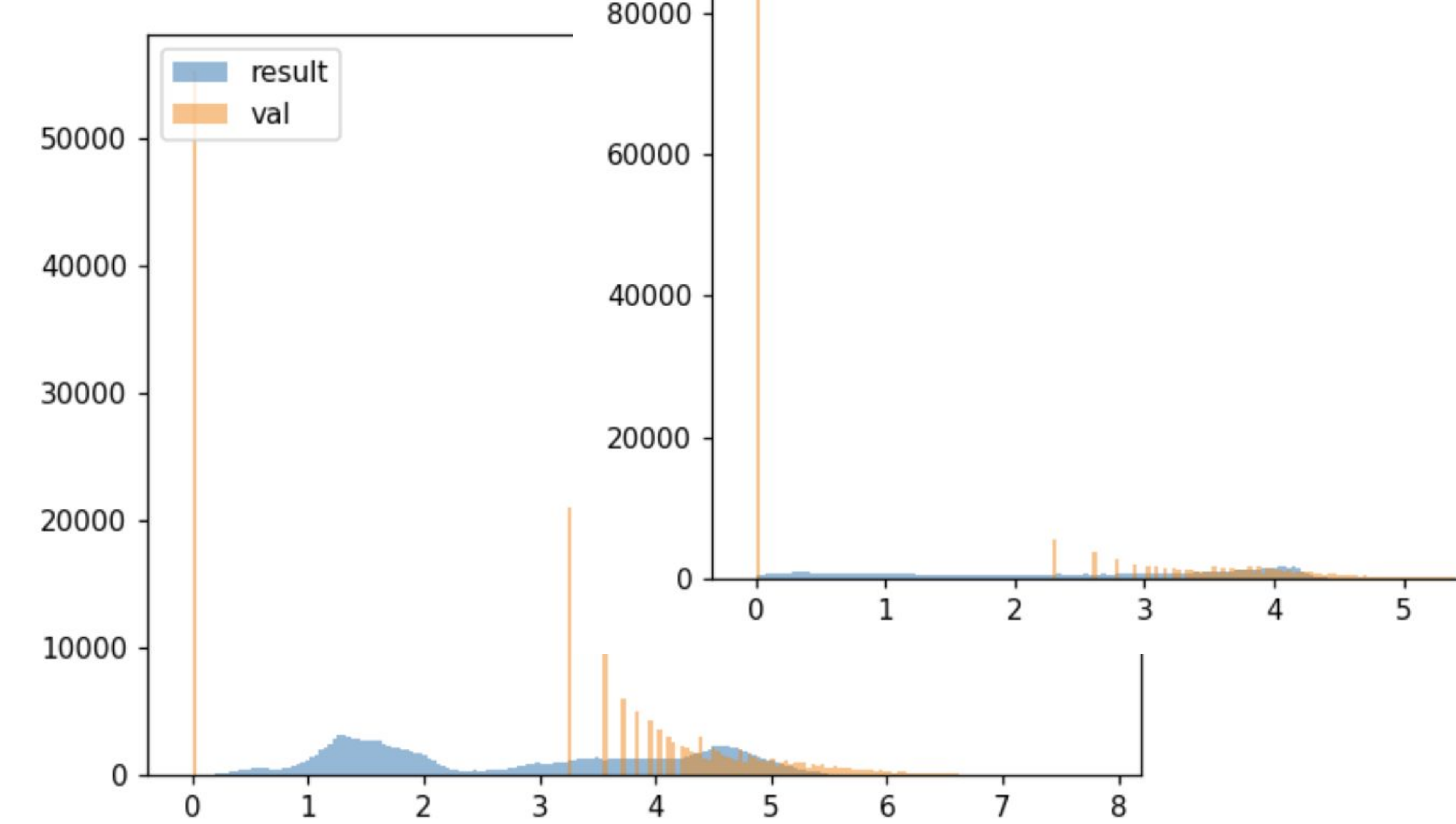


$$J_{N,\mathrm{Total}}(\Omega) = j_N(E_1,\Omega)\,(E_2 - E_1) + \left[\sum_{i=2}^{18} j_N(E_i,\Omega)\frac{(E_{i+1} - E_{i-1})}{2}\right] + j_N(E_{19},\Omega)\,(E_{19} - E_{18}).$$

units: $\dfrac{1}{cm^2 \cdot s \cdot ster}$



Log Scale Electron Precipitation Number Flux Log10 [#/cm^2/s] 2013-03-17 19:34:07
OVATION Pyme model
neural net: 0.5 deg / Mlat
measured value
current time
03-17 00
03-17 03
03-17 06
03-17 09
03-17 12
03-17 15
03-17 18
03-17 21
Example Magnetic Indices
AL
AE
AU
time (month, day, year labels)
Predicted Electron Precipitation Number Flux (OVATION Pyme)
Predicted Electron Precipitation Number Flux (Neural Net)
12:00
15:00
9:00
18:00
6:00
21:00
3:00
0:00
80°
70°
60°
1e10
Total Flux [#/cm^2/s]

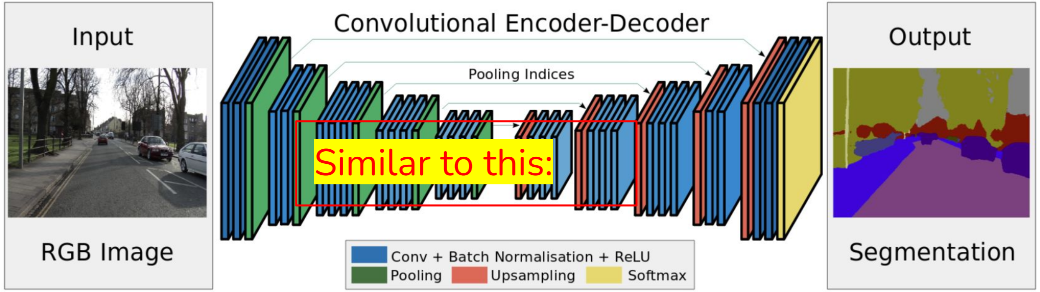

result
val
50000
40000
30000
20000
10000
0
0
1
2
3
4
5
6
7
8
80000
60000
40000
20000
0
0
1
2
3
4
5

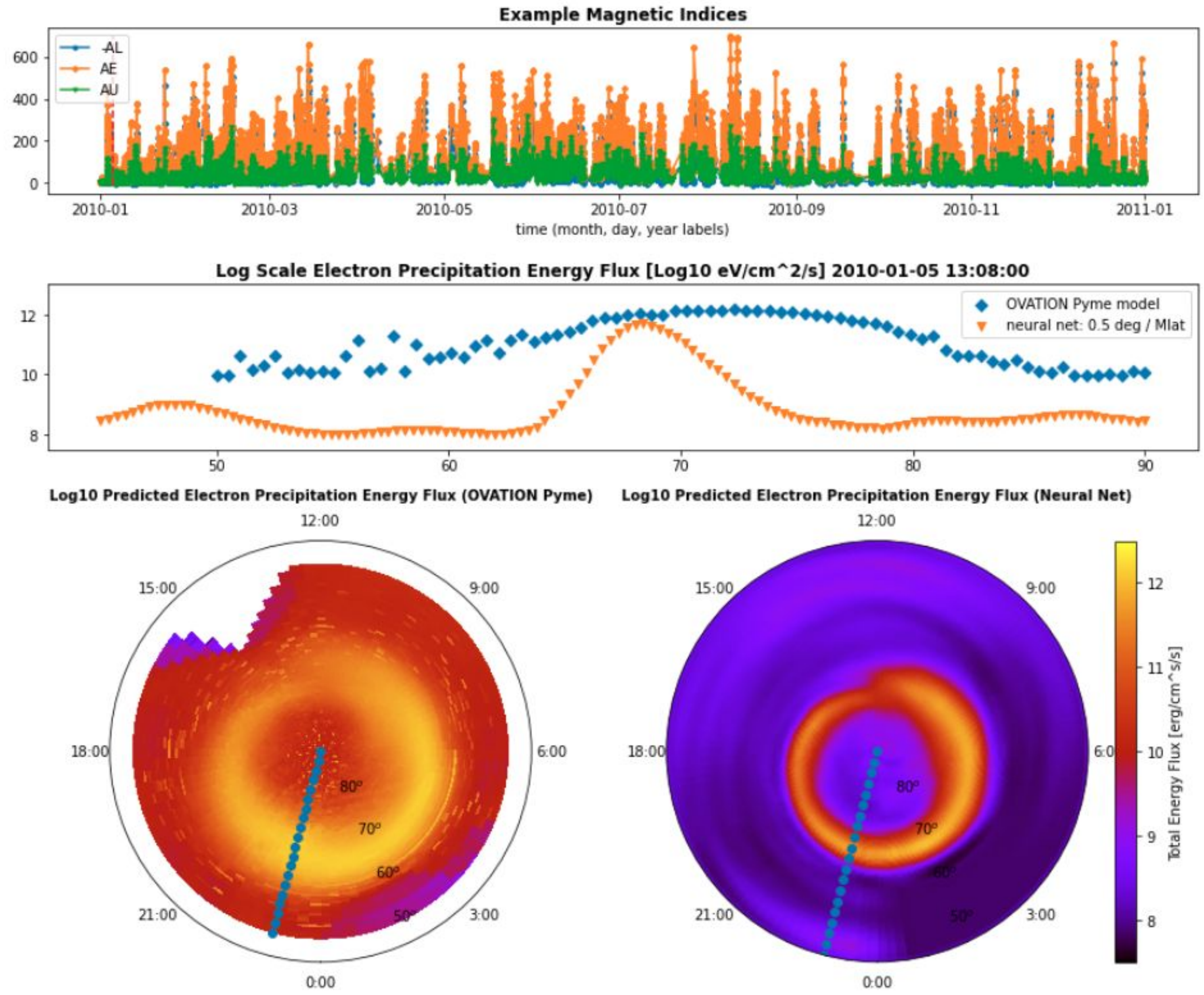
Conv2D_transpose Formulation

Inputs: 'ID_SC', 'sin_ut','cos_ut', 'sin_doy', 'cos_doy',
'F107', 'AE', 'AL', 'AU', 'SymH',
'F107_6hr', 'AE_6hr', 'AL_6hr', 'AU_6hr', 'SymH_6hr',
'F107_5hr', 'AE_5hr', 'AL_5hr', 'AU_5hr', 'SymH_5hr',
'F107_3hr', 'AE_3hr', 'AL_3hr', 'AU_3hr', 'SymH_3hr',
'F107_1hr', 'AE_1hr', 'AL_1hr', 'AU_1hr', 'SymH_1hr'

```
model1 = Dense(256, activation='relu')(input1)
model1 = model1 = Dropout(0.5)(model1)
model1 = Dense(int(64),   activation='relu')(model1)
model1 = Dense(int(32),   activation='relu')(model1)
model1 = Dense(int(256),   activation='relu')(model1)
model1 = tf.keras.layers.Reshape((16, 16, 1))(model1)
# 16x16 to 16 32x32 feature map
model1 = tf.keras.layers.Conv2DTranspose(4, (9,9),
                strides=(2,2), padding='same')(model1)
model1 = tf.keras.layers.Conv2DTranspose(4, (5,5),
                strides=(4,4), padding='same')(model1)
# 4 128x128 feature map to 1 128x128
model1 = model1 = Dropout(0.5)(model1)
model1 = PeriodicPadding2D(3)(model1)
model1 = tf.keras.layers.Conv2D(1, kernel_size=(7,7),
                              padding='valid')(model1)
```

```
Layer (type)                 Output Shape
=================================================================
input_1 (InputLayer)         [(None, 145)]

dense (Dense)                (None, 256)

dropout (Dropout)            (None, 256)

dense_1 (Dense)              (None, 64)

dense_2 (Dense)              (None, 32)

dense_3 (Dense)              (None, 256)

reshape (Reshape)            (None, 16, 16, 1)

conv2d_transpose (Conv2DTran (None, 32, 32, 4)

conv2d_transpose_1 (Conv2DTr (None, 128, 128, 4)

dropout_1 (Dropout)          (None, 128, 128, 4)

periodic_padding2d (Periodic (None, 134, 134, 4)

conv2d (Conv2D)              (None, 128, 128, 1)
=================================================================
Total params: 65,281
```





2010-01-05 23:27:00

```python
class DataGenerator_train(keras.utils.Sequence):
    'Generates data for Keras'
```

MLat

0	0	0	0	0
0	0	0	0	0
0	0	(Ytrue-Ypred)^2	0	0
0	0	0	0	0
0	0	0	0	0

MLocalTime

Left-right Periodic Padding
(not yet implemented)

Loss function for each time step and target value

Note: 2048 are combined for each batch

```python
def custom_mse(y_true, y_pred):


    mse = K.sum( K.cast(K.greater(y_true, 0),'float32')*(
        K.square(y_true-y_pred)) )/params['batch_size']
    return mse
```

```python
model1 = Dense(256, activation='relu')(input1)
model1 = model1 = Dropout(0.5)(model1)
model1 = Dense(int(64),   activation='relu')(model1)
model1 = Dense(int(32),   activation='relu')(model1)
model1 = Dense(int(256),   activation='relu')(model1)
model1 = tf.keras.layers.Reshape((16, 16, 1))(model1)
# 16x16 to 16 32x32 feature map
model1 = tf.keras.layers.Conv2DTranspose(4, (9,9),
                       strides=(2,2), padding='same')(model1)
model1 = tf.keras.layers.Conv2DTranspose(4, (5,5),
                       strides=(4,4), padding='same')(model1)
# 4 128x128 feature map to 1 128x128
model1 = model1 = Dropout(0.5)(model1)
model1 = PeriodicPadding2D(3)(model1)
model1 = tf.keras.layers.Conv2D(1, kernel_size=(7,7),
                                     padding='valid')(model1)
```

Time dependent inputs are only changing every 5 min (from NASAOMNI)

One min available (but not as accurate)

Idea: use all traced data (collected at 1 sec cadence) within a 5 minute window for each 5 min cadence input

Also use multiple SC_IDs at once for satellites

```python
history = model.fit(training_generator,
                    validation_data=validation_generator,
                    batch_size=2048,epochs=400,#verbose=2,
                   callbacks=[tf.keras.callbacks.EarlyStopping(moni
              patience=50)], use_multiprocessing=True,
                     workers=6)#
```

- Uses tf.keras Data_generator class which creates batches as needed
    - Save all 160 GB data in data files and load dynamically, each epoch still can use all data at once

```python
class DataGenerator_train(keras.utils.Sequence):
    'Generates data for Keras'
```

New Idea,
For each 2D target use all Spacecraft at that time step as well as data from nearby times.
This assumes time scale changes are >> 1 second

MLat

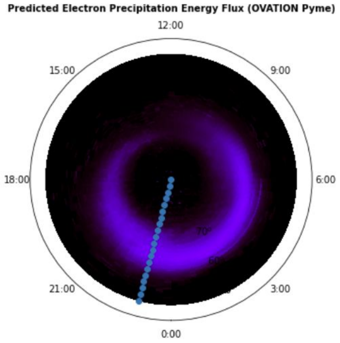
0	0	0	0	0	0	0	0	0	0
0	F16 Spacecraft	0	Y ti+1	0	0	0	0	0	0
0	0	Y ti	0	0	0	0	Y ti+1	0	0
0	Y ti-1	0	0	0	0		0	Y ti	Y ti-1
0	0	0	0	0	0	0	F17 Spacecraft	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

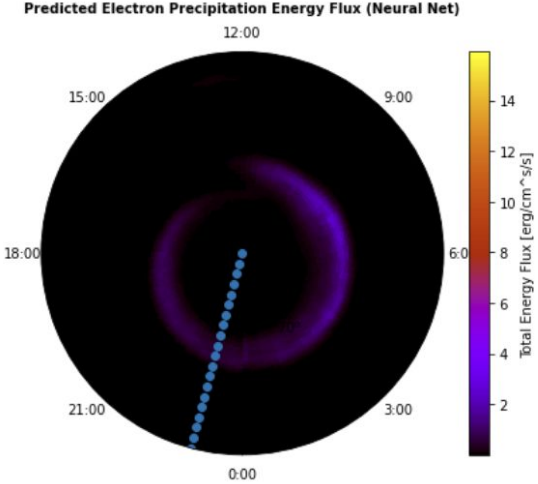
MLocalTime

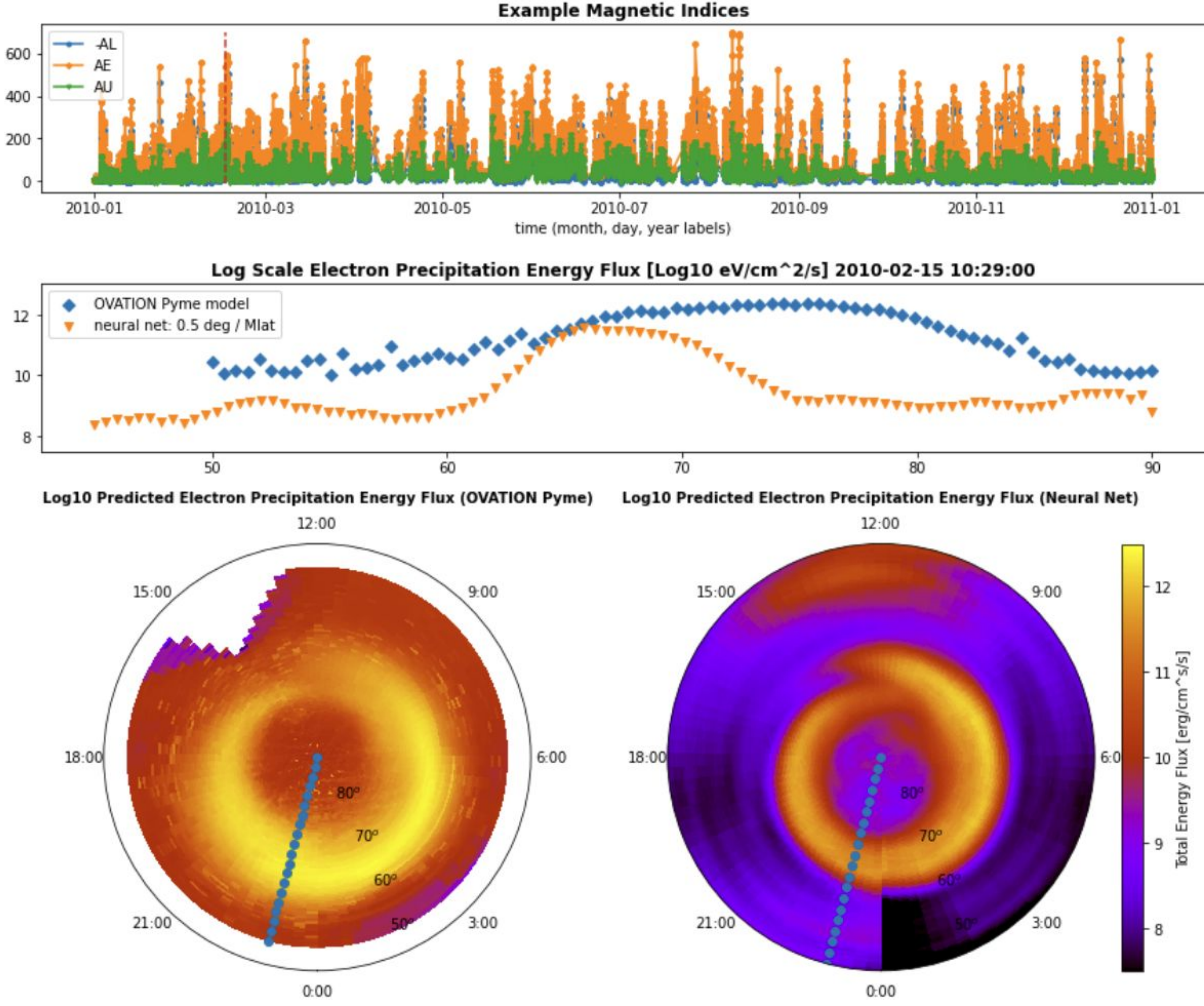
Loss function for each "time step" and 2D target value

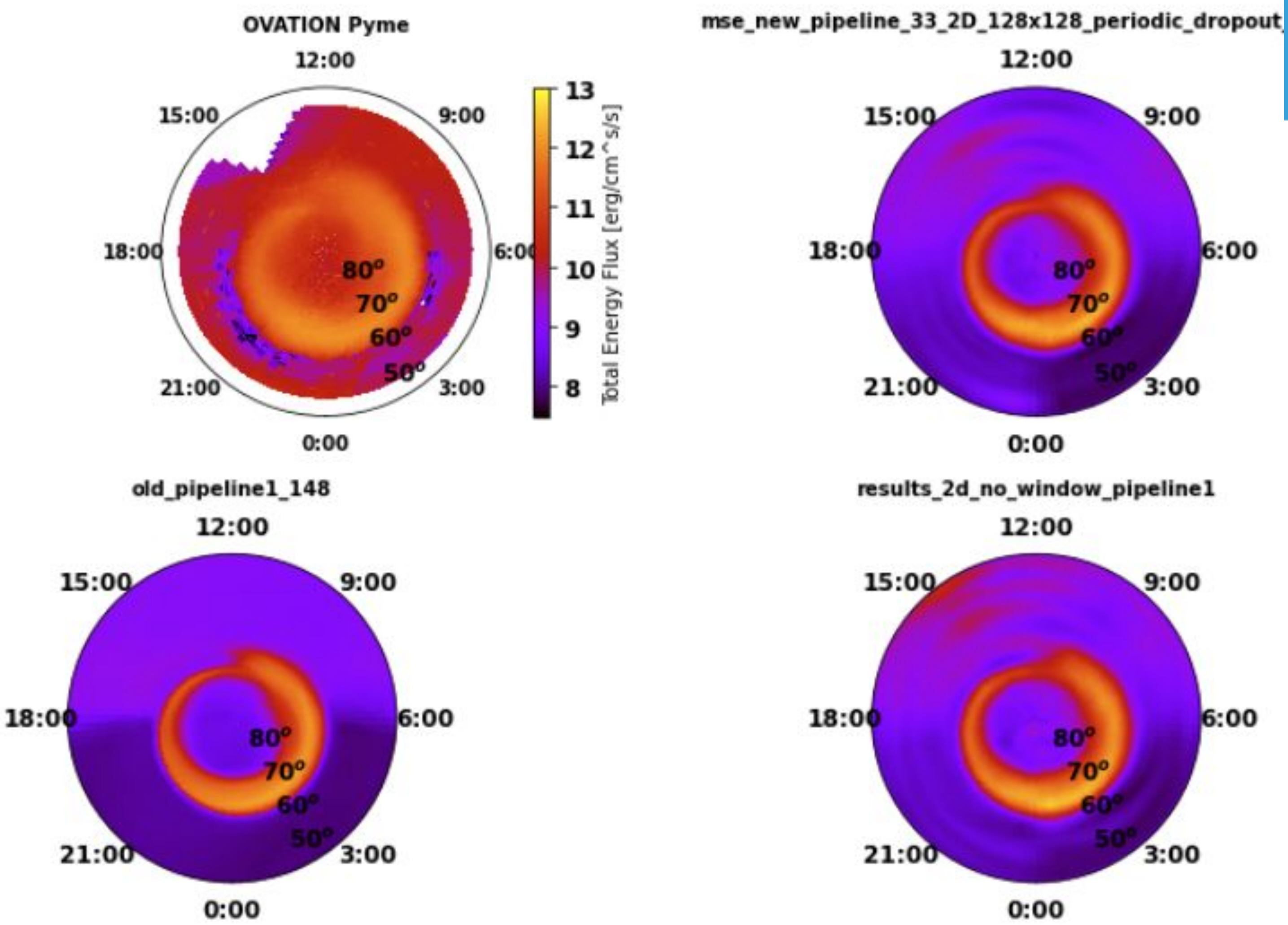
Note: 1024 are still combined for each batch

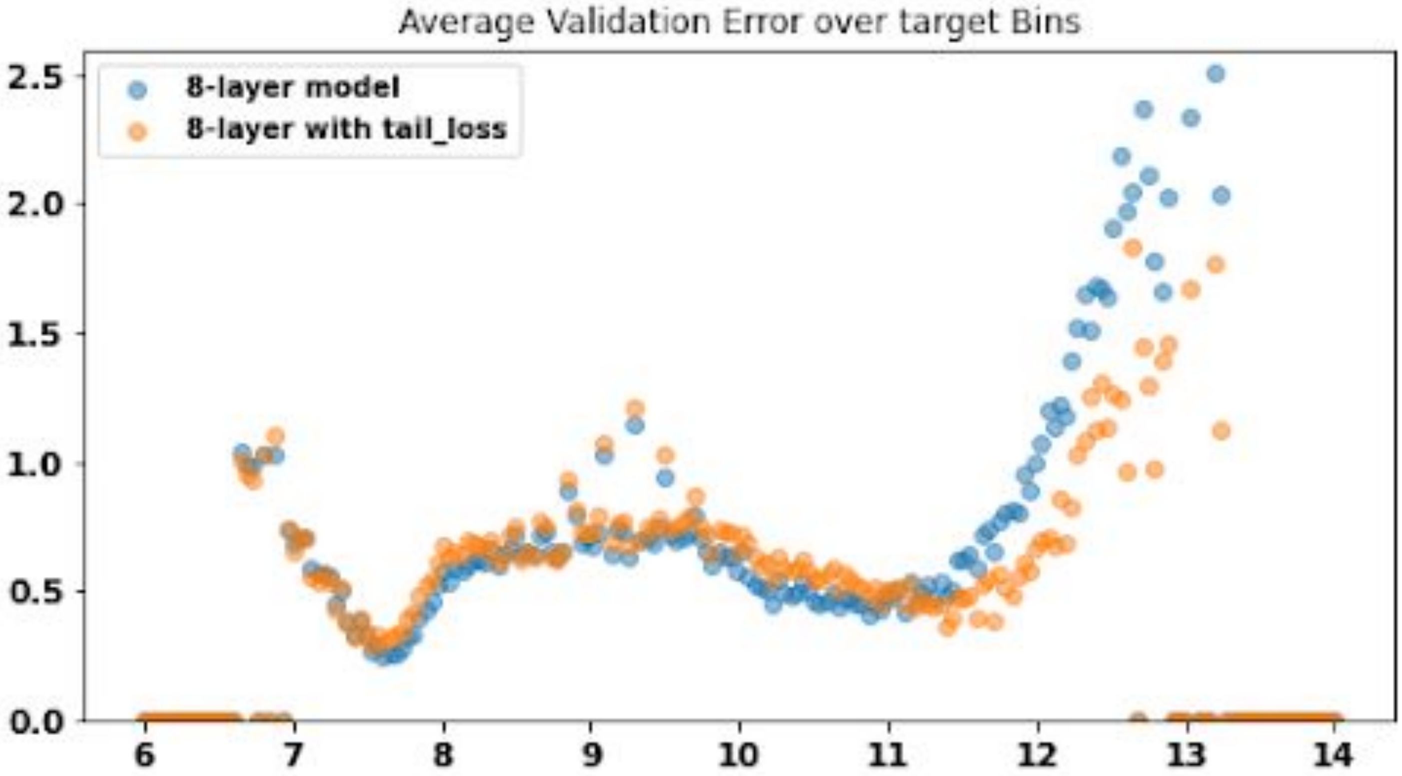
145 Inputs (with solar wind)








OVATION Pyme
mse_new_pipeline_33_2D_128x128_periodic_dropout
old_pipeline1_148
results_2d_no_window_pipeline1
12:00
15:00
9:00
18:00
6:00
21:00
3:00
0:00
80°
70°
60°
50°
Total Energy Flux [erg/cm^s/s]
13
12
11
10
9
8



Average Validation Error over target Bins
8-layer model
8-layer with tail_loss
2.5
2.0
1.5
1.0
0.5
0.0
6
7
8
9
10
11
12
13
14

https://github.com/rmcgranaghan/HEARTBEAT

New Database

- Full DMSP dataset is > 100 GB, cadence is 1 second
    - Currently using every 60th point (1 minute data)
- Still using indices at 5 minute cadence

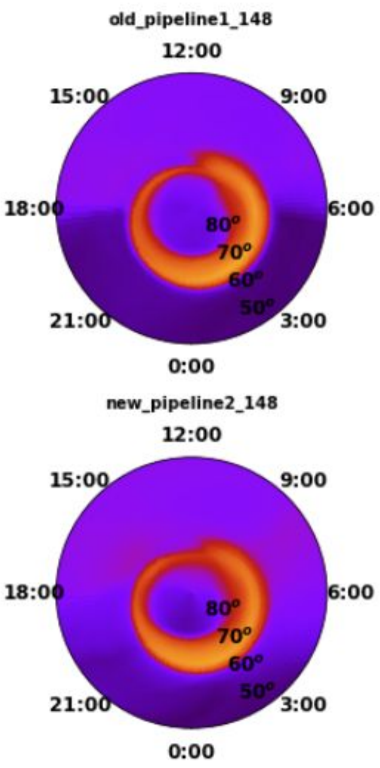
- All DMSP data is saved to CDF files locally
- Indices and solar wind data is access online and moving averages over time a formed

- Can very rapidly create ranges of 1 second data to test for validation (a few days

  - Great for testing single storms

Goal: Create new custom datasets from 1 second data (currently using 1 minute data)

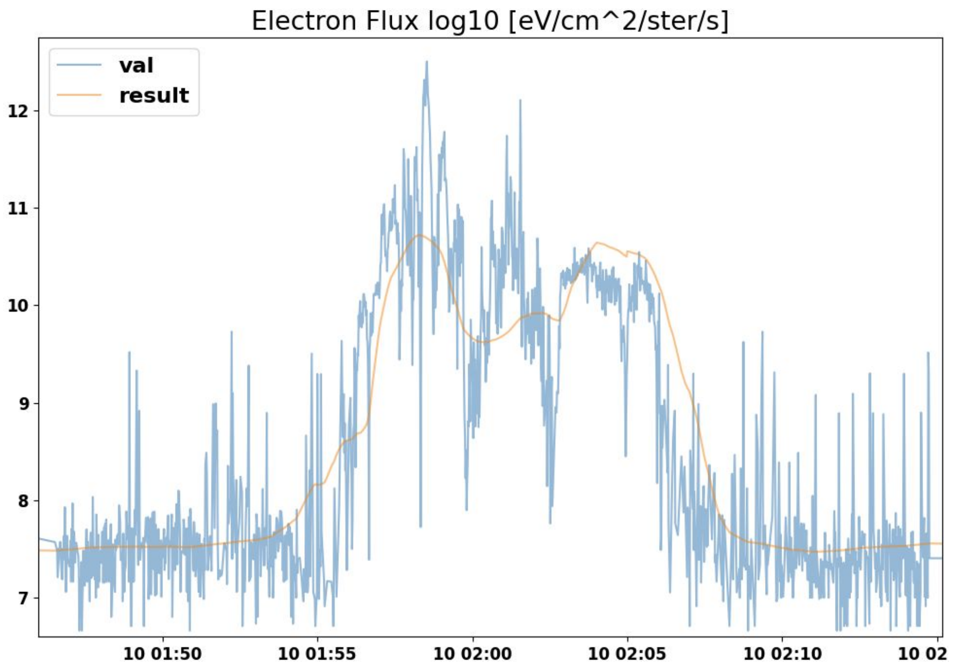
- Our mesoscale interest needs 15 second cadence to get appropriate spatial resolution
- Whole database create is slow,
    - ideal to parallelize and store NASAOmni time histories locally to run faster



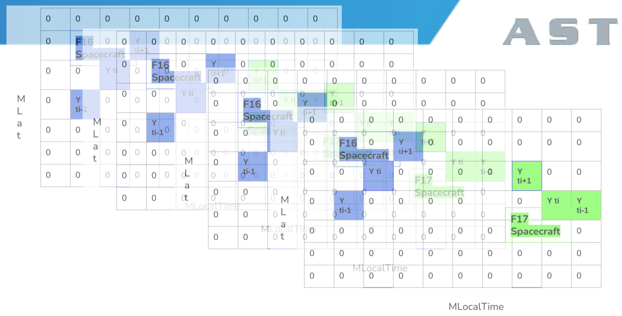
1 second data comparisons

Current issues

- 60x more data than 1 min cadence
- First tries of training on a subset using dense neural network (PrecipNet) averages out the variations
- 1 second data appears very random, hard to separate out “mesoscale” and “smaller scales”
  - Maybe use an approach similar to turbulence modeling??



Future work and Ideas



- Simultaneous Spacecraft 2d model
  - Adding in multiple timestep predictions from one input time
    - Would work like predicting multiple movie frames
- Channel-based Energy Flux (similar to number flux)

$$J_{TOT} = j(E_1)(E_2 - E_1) + \sum_{i=2}^{15} j(E_i)\frac{E_{i+1} - E_{i-1}}{2} + j(16)(E_{16} - E_{15})$$

the integral energy flux in units of keV/cm^2 s sr defined as

$$JE_{TOT} = E_1 j(E_1)(E_2 - E_1) + \sum_{i=2}^{15} E_i j(E_i)\frac{E_{i+1} - E_{i-1}}{2} + E_{16} j(E_{16})(E_{16} - E_{15})$$

- Improving mesoscale identification
  - Simple solution: running at higher resolution
  - Extended solution: use 1-second resolution data (right now using 60-second sub-sampled data samples)
- Tail Loss function exploration
- Phase II ideas
  - Multi-task learning (auroral region + energy and number flux)
- The challenges:
  - Computational burden of developing the full-map model? Of using one-second data?

https://github.com/rmcgranaghan/HEARTBEAT