Iowa Initiative for Artificial Intelligence
Final Report

Project title:	Deep Digital Twins: A Framework to Make Cities More Sustainable and Resilient	
Principal Investigator:	Joe Gomes and Gregory Carmichael, Department of Chemical & Biochemical Engineering	
Prepared by (IIAI):	Avinash Mudireddy	
Other investigators:		
Date:	07/11/2022	
Were specific aims fulfilled:	Aim 1 is fulfilled Aim 2 – No data	
Readiness for extramural proposal?		
If yes … Planned submission date		2023
Funding agency		NSF, NASA
Grant mechanism		
If no … Why not? What went wrong?		

<u>Brief summary of accomplished results:</u>

<u>Research report:</u>
Aims (provided by PI):

The goal is to integrate modern spatial-temporal deep learning into a prototype digital twin for urban applications and demonstrate the power of such a tool in answering complex and pressing problems related to sustainability and resilience. The project specific aims are:

1. Construct a deep learning model for the prediction of NO2 concentration fields (y) from emissions, mobility, land use, and meteorological observational data (X);
2. Integration of the trained deep learning model into the prototype digital twin modeling system to test and evaluate the digital twin during COVID-19 shutdown and recovery periods.

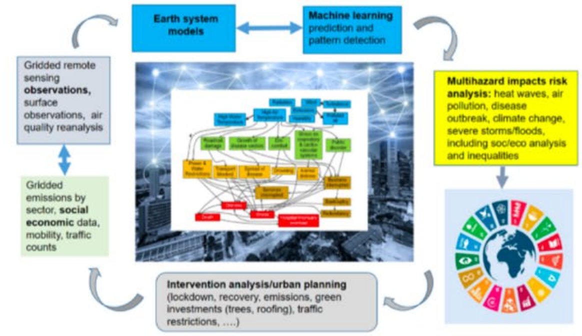


Figure 1. Digital Twins framework to conduct research focused on making cities more sustainable and resilient. Details presented in the text.

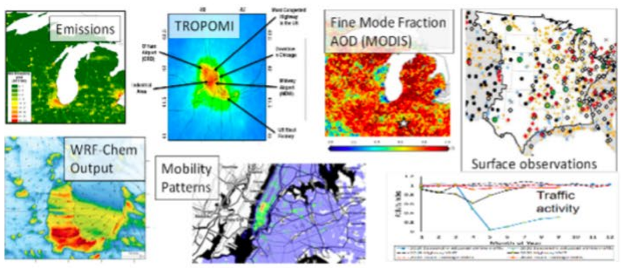


Figure 2: Examples of inputs relevant to machine learning, and currently assembled for use.

Data for Aims:

The basic elements of the digital twin that we plan to develop consist of combining in a single analysis framework: i) physical based models of air quality, weather and climate; ii) modern spatial/temporal machine learning techniques; iii) observational data from multiple sources, including new satellite capabilities that can detect air pollutants (e.g., TROPOMI), routine air quality monitors, and low cost sensors; iv) detailed physical representation of the urban environment including roadways, schools, building types and heights, population distribution, mobility data, traffic patterns, and socio-economic information; v) emission estimates of air pollutants and greenhouse gases by source sector driven by various policy interventions; vi) an intervention, strategy and planning module that modifies the urban environment in specified ways (e.g., green roofs, lockdowns, traffic electrification) and estimates costs associated with the interventions; and vii) a multi-hazard risk analysis module where exposure to air pollutants, heat waves, floods, etc. for various specified interventions can be benchmarked against a baseline, and impacts (such as air pollution exposure and resultant morbidity and mortality), can be assessed using a variety of metrics including those related to social equity.

AI/ML Approach:

We proposed to use the Generative adversarial network (GAN) to solve this prediction problem.

<u>The data for the model includes:</u>

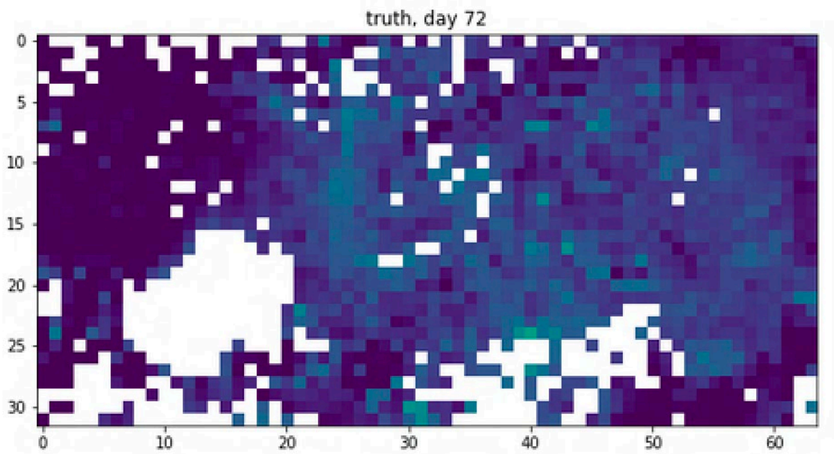
Modern-Era Retrospective analysis for Research and Applications, Version 2 (MERRA-2) data which includes the inputs for the model.

1) ASM data:
   a) t2m_data: 2-m air temperature [K]
   b) ps_data: surface pressure [Pa]
   c) u_data, v_data: 10-m wind speed [m/s]
2) GAS data:
   a) aod_data: aerosol optical depth []
3) AER data:
   a) rh_data: relative humidity []
   b) pm25_data: PM2.5 concentration [uh m-3]

4) FLX data:
   a) prec_data: total precipitation (bias corrected) [kg m-2 s-1]
   b) pblh_data: planetary boundary layer height [m]

The TROPOspheric Monitoring Instrument (TROPOMI) satellite data for NO2 gas concentration as the output label. The map/area of coverage for MERRA2 and TROPOMI are different as they measure the atmosphere at a different scale. However, using Lat and Long information, we stitched the maps to fit the input size to the output size.
The input is scaled using power transformation followed by a min_max_scalar. The outputs are scaled using a min_max_scalar.

However, it is to be noted that the output NO2 map contains a lot of missing values. Example:



The missing values are represented by white pixels.

<u>Model design:</u>
The GAN model consists of a generator and a discriminator network.
In our GAN model, we designed the generator to be a UNET model which takes in the stacked MERRA image map as input and outputs the map similar to NO2 tropomi data. The total generator loss is then calculated using

The total generator loss = Ganloss + lambda * l1_loss
Ganloss = -1* reduced mean(log likelihood(1- discriminator generated output))
l1_loss = mean squared error (Actual NO2 output map, Predicted generator map)

[Note, only non-empty pixels are compared]

The discriminator model is a Convolutional Neural Network (CNN) that has two parts while training:
- The real output part which takes in the stacked MERRA image map along with the real NO2 map
- The generated or fake part which takes in the stacked MERRA image map along with generator predicted NO2 map

The discriminator loss is then calculated as:
The total discriminator loss = -1* reduced mean([-1* log likelihood(discriminator real output) + The total generator loss])

During the training, the algorithm is designed in a way that generator tries to mimic the original NO2 prediction whereas the discriminator learns to differentiate between real and generated/fake output.

Experimental methods, validation approach:
As a part of the experimental setup,
The inputs and outputs are cut into (32,64) pixel images to improve the sample size and prediction. The train: test split is 85:15.

Our generator Model is an unet model. It contains

```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_3 (InputLayer)           [(None, 32, 64, 9)]  0           []

 gaussian_noise (GaussianNoise)  (None, 32, 64, 9)   0           ['input_3[0][0]']

 sequential (Sequential)        (None, 16, 32, 32)   2592        ['gaussian_noise[0][0]']

 sequential_1 (Sequential)      (None, 8, 16, 64)    18688       ['sequential[0][0]']

 sequential_2 (Sequential)      (None, 4, 8, 128)    74240       ['sequential_1[0][0]']

 sequential_3 (Sequential)      (None, 2, 4, 256)    295936      ['sequential_2[0][0]']

 sequential_4 (Sequential)      (None, 1, 2, 512)    1181696     ['sequential_3[0][0]']

 sequential_5 (Sequential)      (None, 2, 4, 512)    2361344     ['sequential_4[0][0]']

 concatenate_1 (Concatenate)    multiple             0           ['sequential_5[0][0]',
                                                                  'sequential_3[0][0]',
                                                                  'sequential_6[0][0]',
                                                                  'sequential_2[0][0]',
                                                                  'sequential_7[0][0]',
                                                                  'sequential_1[0][0]',
                                                                  'sequential_8[0][0]',
                                                                  'sequential[0][0]']

 sequential_6 (Sequential)      (None, 4, 8, 256)    1770496     ['concatenate_1[0][0]']

 sequential_7 (Sequential)      (None, 8, 16, 128)   442880      ['concatenate_1[1][0]']

 sequential_8 (Sequential)      (None, 16, 32, 64)   110848      ['concatenate_1[2][0]']

 conv2d_transpose_5 (Conv2DTran  (None, 32, 64, 1)   1537        ['concatenate_1[3][0]']
 spose)

==================================================================================================
Total params: 6,260,257
Trainable params: 6,256,417
Non-trainable params: 3,840
```

The discriminator CNN model is:

```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 32, 64, 9)]  0           []

 input_2 (InputLayer)           [(None, 32, 64, 1)]  0           []

 concatenate (Concatenate)      (None, 32, 64, 10)   0           ['input_1[0][0]',
                                                                  'input_2[0][0]']

 conv2d (Conv2D)                (None, 16, 32, 64)   10304       ['concatenate[0][0]']

 leaky_re_lu (LeakyReLU)        (None, 16, 32, 64)   0           ['conv2d[0][0]']

 conv2d_1 (Conv2D)              (None, 8, 16, 128)   131200      ['leaky_re_lu[0][0]']

 leaky_re_lu_1 (LeakyReLU)      (None, 8, 16, 128)   0           ['conv2d_1[0][0]']

 conv2d_2 (Conv2D)              (None, 4, 8, 256)    524544      ['leaky_re_lu_1[0][0]']

 leaky_re_lu_2 (LeakyReLU)      (None, 4, 8, 256)    0           ['conv2d_2[0][0]']

 zero_padding2d (ZeroPadding2D)  (None, 6, 10, 256)  0           ['leaky_re_lu_2[0][0]']

 conv2d_3 (Conv2D)              (None, 3, 7, 512)    2097152     ['zero_padding2d[0][0]']

 batch_normalization (BatchNorm  (None, 3, 7, 512)   2048        ['conv2d_3[0][0]']
 alization)

 leaky_re_lu_3 (LeakyReLU)      (None, 3, 7, 512)    0           ['batch_normalization[0][0]']

 zero_padding2d_1 (ZeroPadding2  (None, 5, 9, 512)   0           ['leaky_re_lu_3[0][0]']
 D)

 conv2d_4 (Conv2D)              (None, 5, 9, 1)      8193        ['zero_padding2d_1[0][0]']

 activation (Activation)        (None, 5, 9, 1)      0           ['conv2d_4[0][0]']

==================================================================================================
Total params: 2,773,441
Trainable params: 1,024
Non-trainable params: 2,772,417
```

We used Adam optimizer (lr=0.0001) with a gradient clipvalue of 0.5. The model is trained for 1000 epochs with 512 as batch size.

Results:

The results for the model are as follows.

During training:

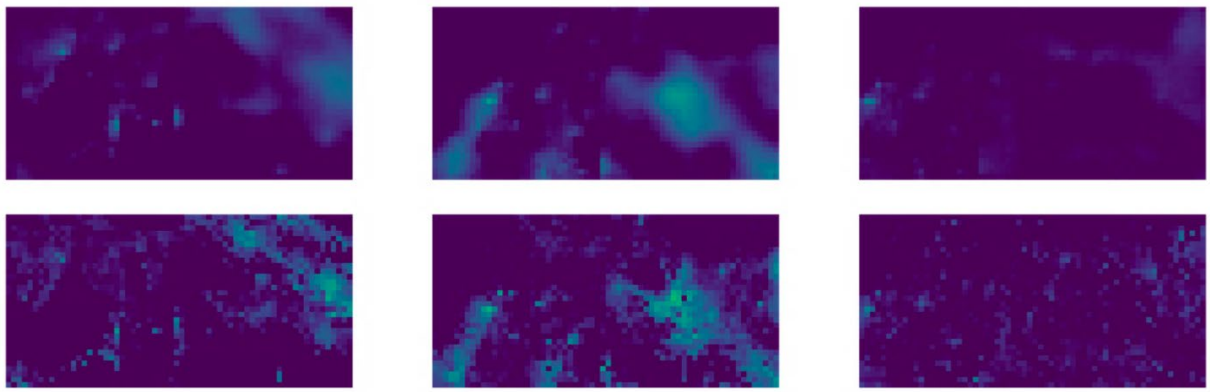


Figure (a)

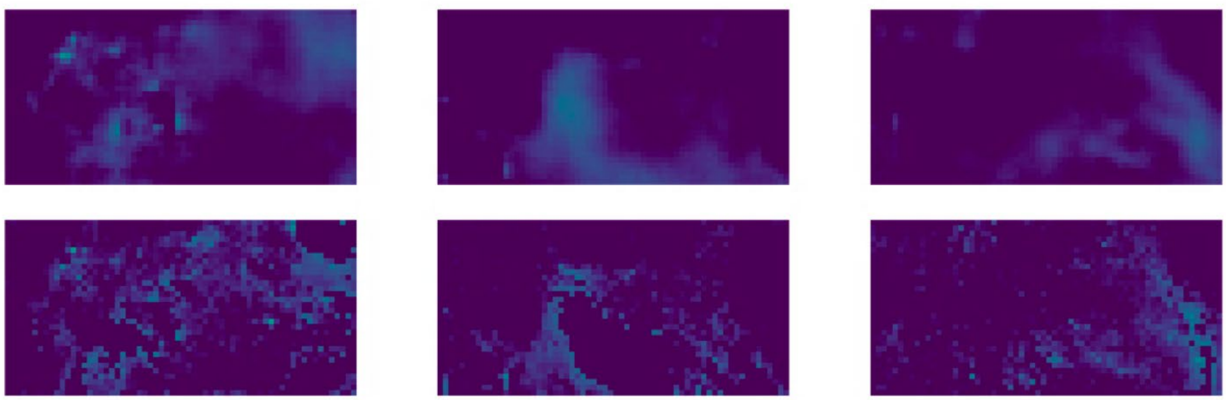
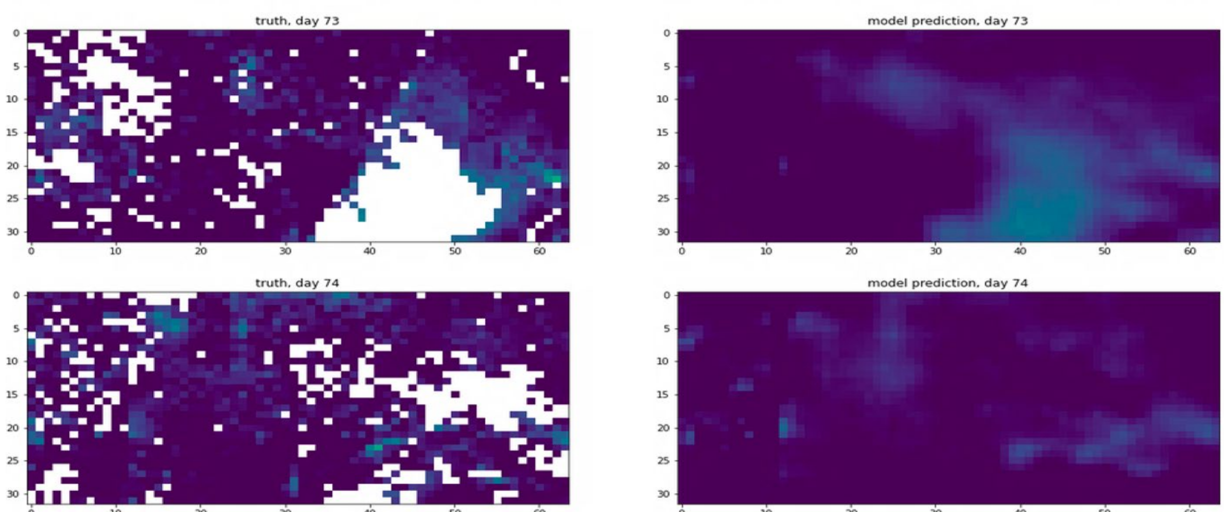


Figure (b)

In the above figure panels (a) and (b), the bottom rows represent the actual NO2 maps and the top rows represent the generator model generated NO2 maps.

During testing:



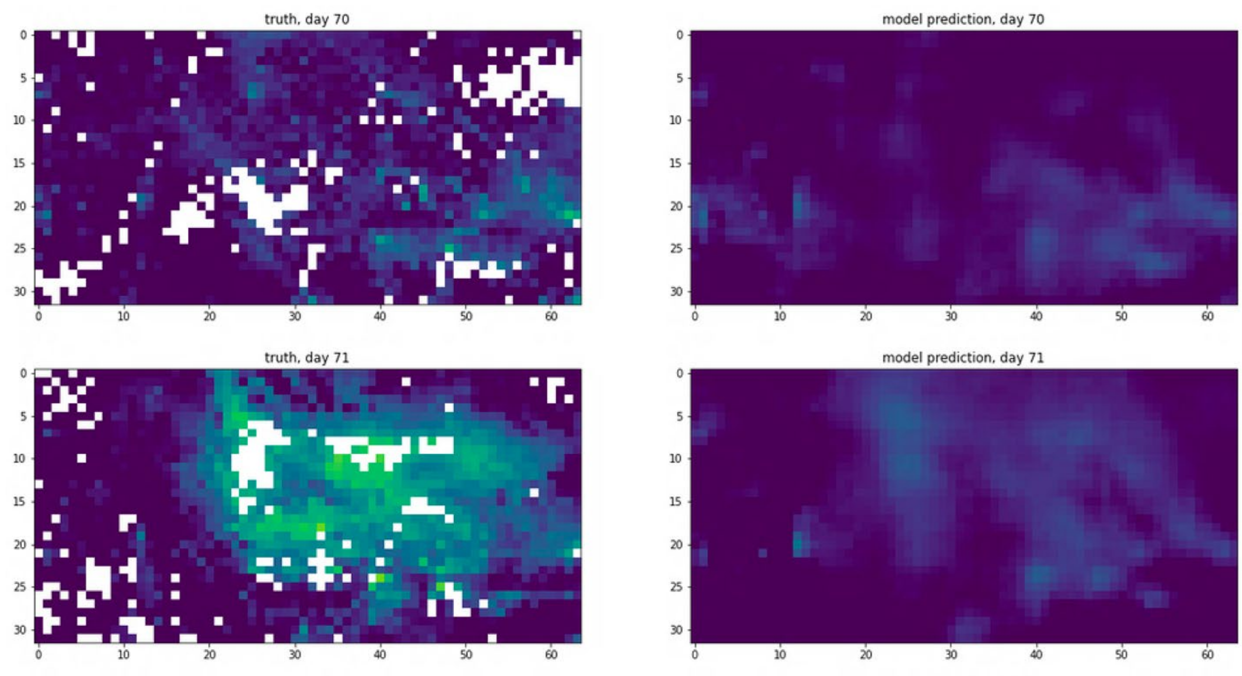


Figure (c)

In Figure c, the left side represents the actual NO2 maps and the right side represent the generator model generated NO2 maps.

The adjusted R2 score = 0.2755930025904001

* adjusted R2 score is the r2 score of actual and predicted maps where there are non-zero pixels.

Ideas/aims for future extramural project:

The development of a digital twin focused on watershed issues is a main element of a current NSF Engineering Research Center proposal (5yrs/$5M/yr). This preproposal is currently under review at NSF and we are awaiting news regarding the opportunity to submit a full proposal. The results from the IIAI seed grant will be used in new grant proposals focused on building more comprehensive digital twins and their application. One proposal will be developed for the NSF Leading Engineering for America's Prosperity, Health, and Infrastructure (LEAP HI) program (5-yr; $1-2M total; LoI July 15, full proposals Sept. every year). We also expect to use the digital twin as a key element in a proposal to NASA’s Atmospheric Composition: Modeling and Analysis Program.