NN-SAR: A Neural Network Approach for Spatial AutoRegression

Pranita Dewan, Raghu Ganti, Mudhakar Srivatsa
IBM T.J Watson Research Center
Email: {dewanp, rganti, msrivats}@us.ibm.com

Sebastian Stein
University of Southampton
Email: ss2@ecs.soton.ac.uk

***Abstract*—Geographic phenomena such as weather, pollution, pollen are increasingly monitored through deployments of wide-area networked sensors. However, the coverage of these sensors is limited to key densely populated regions. A standard approach to inferring missing spatial and temporal values is to use regression. In this paper, we present a new approach, NN-SAR, to inferring spatiotemporal values from existing deployed sensors. We model this inference problem as that of learning a spatial representation of the underlying phenomena from the existing data and use deep learning based *auto-encoder* approach. Classical auto-encoders learn on image or singular time series data without taking "spatial" similarities into account. We present a novel mechanism for encoding the spatially distributed sensor readings as "images" and apply the auto-encoder with convolutional layers to learn an efficient representation of the data, which can then be used to infer missing sensor data. Preliminary results indicate that the performance of our approach is far superior to the state-of-the-art Spatial Auto-Regressive (SAR) models by 20% on average.**

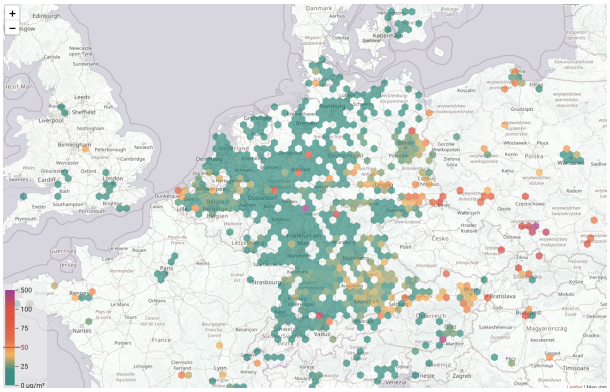


Fig. 1: OK Labs PM sensor map

I. Introduction

Spatiotemporal phenomena such as weather, pollen, air pollution levels are increasingly measured through large scale deployments (e.g., weather stations, fine particulate matter sensors). Most of these deployments are targeted to densely populated regions, where it is easy for sensors to be installed, configured, and wired to the Internet. However, nearby regions that are unmonitored by these installations lack measurements. Unlike other localized and indoor phenomena, wide area geographic phenomena can be estimated in regions lacking measurements using statistical and mathematical models [1]–[4]. In this paper, we propose a novel neural network based modeling approach for estimating spatiotemporal phenomena. We show that our approach performs better than standard spatial auto-regresssive models when enough data is provided for the neural network approach.

A standard and state of the art technique for estimating missing values in spatiotemporal measurements is called Spatial AutoRegression (SAR) [5], [6]. A SAR model predicts a continuous outcome variable (e.g., weather conditions, incidence of disease, pollution levels) based on input variables (e.g., cloud cover, number of vehicles). One can think of SAR as linear regression applied in the spatial domain, by allowing for the outcomes to be affected by outcomes, covariates, and errors in nearby areas. We provide a further explanation of SAR models in later parts of the paper. However, a challenge with SAR models is that they are only as good as the choice of the input variables. In theory, if all the variables are chosen accurately, SAR models can achieve optimal prediction accuracy. To address this drawback of SAR models, we decided to explore a data driven approach to modeling spatiotemporal phenomena. We observe that as more and more data is collected, through long term and denser deployments, applying a data driven approach for modeling, such as neural networks, can provide us significantly better results than traditional modeling approaches, such as SAR. Neural networks have become quite popular in the last five years [7]–[9], their applications range from image classification, video analysis, language translation, to medical imaging [10]–[15]. There are two broad ways in which neural networks can be used, one is to learn models for classification tasks and the other is to learn models for prediction tasks. Our interest is in prediction that captures spatiotemporal characteristics. Specifically, we believe that the choice of a Convolutional AutoEncoder (CAE) is appropriate in this scenario. To illustrate this, let us consider a satellite view (or bird's eye view) of the phenomena, the observed measurements are points on a snapshot of the satellite view. If we were to grid the entire image (using a typical spatial grid), then some of the cells have observations and some of them do not. The goal of the neural network then is to be able to learn a compact representation of the spatiotemporal phenomenon so that it can be used to predict any missing spatial and temporal values. Hence, the choice of a CAE, where convolutions capture the spatial dependencies and the auto-encoder extracts generic features and learns the key input variables. The main advantage of such data driven model

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

learning is that the input variables do not need to be explicitly modeled, with such dependencies being learned through the observations themselves. In this paper, we show that such an approach is superior to the traditional modeling approaches using real sensor data collected over a large spatial region (a country) and time range (about a month). The sensors monitor particulate matter, $PM_{2.5}$ and PM_{10} [16], which are standards used to monitor air pollution levels. The approach that we developed can be easily transferable to model other spatiotemporal phenomena as well, as the input variables are not explicitly captured. Our technique, provided enough data, provides on average 20% higher accuracy than the traditional SAR approaches.

II. Problem and Dataset Description

Particulate Matter (PM) refers to microscopic solid or liquid matter suspended in the atmosphere of Earth and is typically classified as a pollutant. Depending on their size, the particles either fall into the PM_{10} category, which includes coarser particles that are generally 10 micrometers or less or the $PM_{2.5}$ category, which includes fine inhalable particles with diameter less than 2.5 micrometers. These pollutants can have adverse effects on the climate as well as human heath, including respiratory diseases and heart issues, given their microscopic size.

Due to these concerns, it is extremely important to be able to always monitor the level of these pollutants in the atmosphere and take the necessary steps to reduce them when they start exceeding the acceptable thresholds. There are large scale efforts to address these problems in the form of deploying sensors that can calibrate the atmospheric particulate matter content, but since this is a continuous ongoing process, the coverage is of course restricted to areas that have a functioning sensor. Since atmospheric particulate matter at a particular location is spatially correlated with that at the neighboring locations, we can use models that capture spatial interactions, such as Spatial Autoregression (SAR) or Convolutional Networks (CNNs) to predict and approximate PM quantities at the locations that do not have sensors to measure these pollutants. While there is an abundance of past work in this area using SAR or other statistical models for prediction [17]–[19], more recently neural network based approaches have also been proposed [20]–[23]. A key difference of our work in this paper compared to other neural network based approaches, is that none of them use convolutional layers to model spatial dependencies.

The dataset that we used in this paper comes from the Open Knowledge Labs [24]. The data is collected from volunteer deployed sensors that can collect air quality data by tracking various metrics such as fine dust matter, temperature and humidity. Figure 1 shows the spatial spread of one day's worth of data from the PM_{10} sensors in the country of Germany. While the dataset consists of data from all over the world, the maximum number of sensors and hence most of the collected data is from European countries. In this paper, we will focus on training and scoring our models with both PM_{10} and $PM_{2.5}$ quantities, in desired spatial locations, given a partial spatial measurement of these metrics, using SAR (section III-A) as well as a Convolutional Auto-encoder (CAE) network approach (section III-B) as mentioned in section I.

III. Approach

In this section, we describe in detail the techniques that we used to solve the problem introduced in the previous section. We first provide an overview of spatial autoregression and its application to the current problem. We then describe the deep neural network approach, including feature extraction and representation for data features to be used with this approach, followed by a description of the architecture of the deep learning pipeline itself.

A. Spatial Autoregression

Spatial Autoregressive (SAR) models have been traditionally used to model spatial dependencies among data observations in various domains, including income, crime rates, population, housing prices, etc. The SAR model is represented as follows:

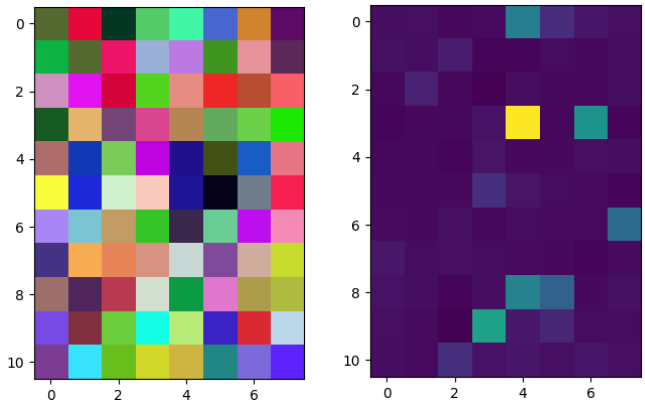
$$y = \rho W y + X\beta + \epsilon \tag{1}$$

where y is an n by 1 vector representing the dependent variable and ρ is a scalar coefficient that represents the strength of spatial dependence, with W representing an n by n spatial weight matrix which quantifies the connections between spatial regions. W is usually defined prior to the regression analysis and is often dependent on the application. X is an n by k matrix representing the observed explanatory variables and β is a k by 1 vector representing the regression parameters for the model. ϵ is an error term that follows a multivariate normal distribution, with zero mean and a constant scalar diagonal variance-covariance matrix $\sigma^2 In$. The regression parameters can be estimated using techniques such as maximum likelihood estimation.

In our application of SAR to the current problem, we first create a spatial grid of the area for which we aim to train a SAR model. The grid is obtained using the technique of geohashing [25]. Data for each cell in the grid is obtained using either the available PM sensor values or through an expanding radius grid neighborhood based bilinear interpolation when we have missing values. We also take into account multiple historical values of y when performing the regression analysis. We experimented with different history lag values and present the results in our evaluation. We generally restrict W to assign weightage to only the immediate neighbor values, since increasing the neighborhood radius did not seem to be particularly helpful for predicting PM values as seen in our evaluation. We trained and applied SAR towards predicting both $PM_{2.5}$ and PM_{10} using data from Germany. The results of this exercise are described in section IV.

B. Convolutional Auto-Encoder

A primary drawback of the SAR model described in the previous section is that the spatial weight matrix, W, has to be defined by domain experts and the task can become increasingly difficult as the size and the spread of the available



(a) Data Image (first 3 channels) (b) Label Image

Fig. 2: Data and Label Images used for the Convolutional Autoencoder

data increases. A SAR model can also be represented as a power series expansion of the form, under the condition that $||\rho W|| < 1$:

$$y = \sum_{i=0}^{\infty} \rho^i W^i (\beta X + \epsilon) \tag{2}$$

The above equation can be interpreted as learning y through the summation of increasing powers of W. In other words, we learn the effect of spatial interactions at increasingly coarser spatial neighborhood sizes. This learning can be simulated as a series of convolutional filters of varying sizes. This equivalence motivated us to use convolutional auto-encoder networks as a technique to implicitly learn the value of W from the available data rather than manually defining it, as is the case with a SAR based model. We first provide an overview of the data transformation and feature extraction steps that are necessary to represent the data in a form that can be consumed by a convolutional auto-encoder. We then describe the high level architecture of the network and the variants that we explored.

1) Feature Extraction and Representation: Since we intend to use a convolutional auto-encoder network to model and predict PM spatial dependencies, a necessary prerequisite for this approach is to be able to represent the PM data in the form of images. We create two images for each example instance to be used with the convolutional auto-encoder network, (i) an input data image and (ii) a target label image. To encode PM information in the form of images, we use geohashing to create a rectangular grid of all locations within the geographical area under consideration. This helps us to preserve spatial locality while encoding the desired information in the form of an image. Every pixel in an image represents the PM information in a particular geohash. The precision of the geohash (area represented by a single geohash) and the geographic area to be modeled determines the size of data and label images. While the height and width of both, the data and the label image is assured to be the same, the number of channels in them differ. A label image encodes the expected PM value in a particular location to the pixel that represents its geohashed value. Since this is a single expected value at a given time instance, a single channel is sufficient to represent this information. So, typically a label image will contain only a single channel. The data image represents the variables that an expected value in a label image depends on. In our current application, the number of channels in a data image is generally equal to the number of historical values that we use to predict the next value. This means that while the data image encodes PM values in the same way as the label image, it stores more than one value per location or geohash. Specifically, it encodes k values if we deem that the values at the last k time steps are important factors in predicting the next value. Hence, this gives us an image with k channels, one for each of the time steps. Figure 2 shows an example of data and label images that encode PM values using the technique described above. The dimensions of the data image are 11x8x5, whereas the dimensions of the label image are 11x8x1. For representation purposes, the data image shows only the first three of the five channels.

2) Deep Learning Pipeline: The architecture of the deep learning pipeline is shown in figure 3. The high level architecture consists of an encoder, which includes a number of interleaved convolutional and max pool layers, with ReLU (Rectifier Linear Unit) activations to obtain a compressed encoding of the original data image, and a decoder which includes the same number of convolutional layers as that of the encoder, in addition to deconvolutional layers. The decoder translates the compressed representation to the desired label image, which contains the target PM values. In addition, the network also includes skip connections from the encoder feature maps to the decoder feature maps to provide additional context that might have been lost in the encoding process, to the reconstruction process. This architecture is inspired by the U-Net architecture that is described in [26]

The convolutional auto-encoder was trained using the stochastic gradient descent implementation from TensorFlow [27], using a MSE (Mean Squared Error) loss function. We experimented with different numbers of convolutional layers and found 3 convolutional layers to provide the best performance. In addition, we also augmented our original image dataset to improve the performance further. Since the source and the target of our deep learning pipeline are images, we can apply traditional image augmentation techniques to increase the size of our dataset and make our network more robust and performant. We used a number of techniques to augment the original dataset, including mirroring, flipping and rotation. Since the images were not too large to begin with, we did not use any cropping. The convolutional auto-encoder models that we trained and evaluated with consist of variants with different numbers of convolutional layers as well as with and without data augmentation. We provide a detailed comparison of all these model variants in our evaluation in section IV

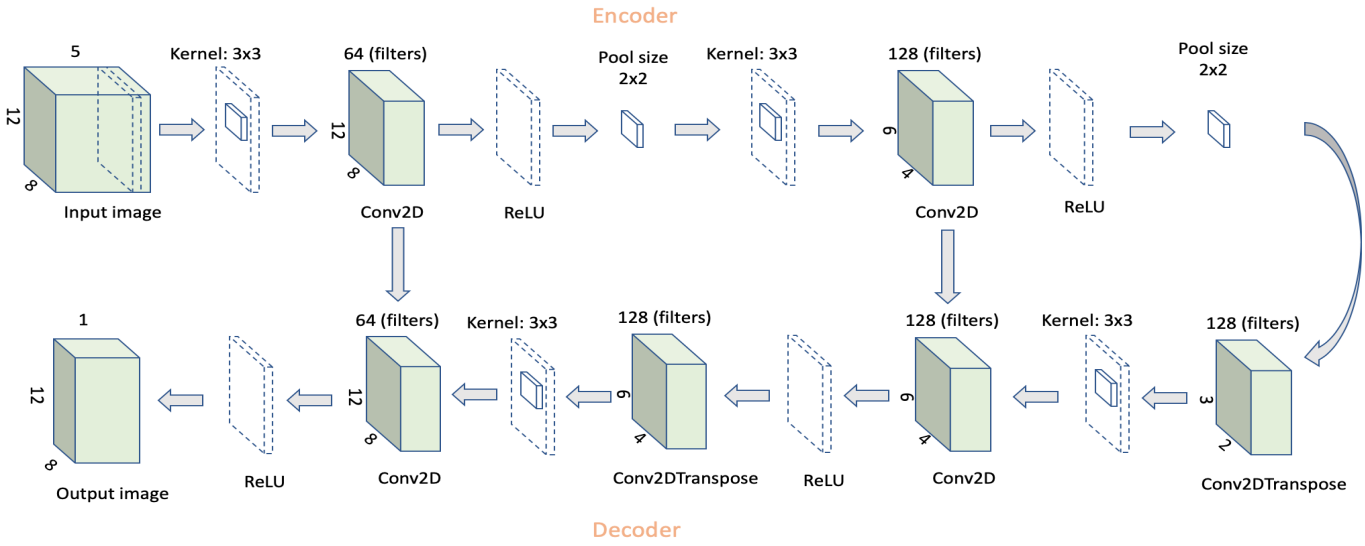


Fig. 3: Deep Convolutional Auto-encoder Pipeline Architecture

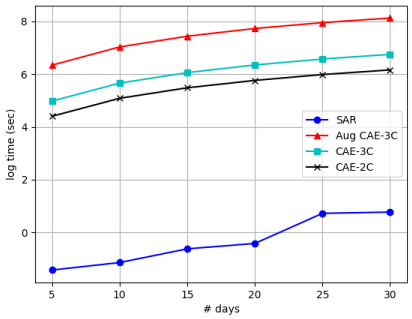


Fig. 4: Training Time

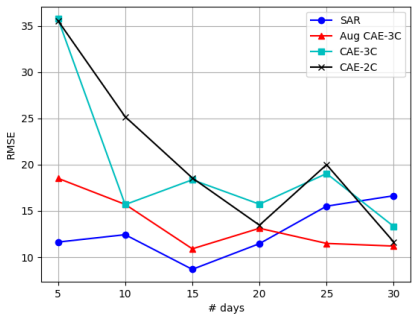


Fig. 5: PM_{10} Prediction Error

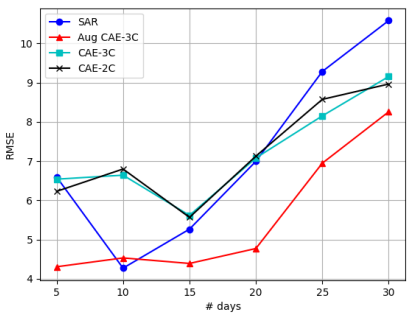


Fig. 6: $PM_{2.5}$ Prediction Error

IV. Evaluation

In this section, we describe the experiments that we conducted to evaluate the performance of various approaches described in the previous section. As mentioned previously, we used particulate matter (PM_{10} and $PM_{2.5}$) data from Open Knowledge Labs. Our SAR model used a Java based implementation and all SAR experiments were run on a machine with 16 GB RAM, 8-cores (2.5 GHz each). All deep learning models were implemented in Python with TensorFlow and experiments were run on a machine with 64 GB RAM, 2-cores and 2 NVIDIA Tesla K40 GPUs.

We primarily used PM data from Germany for the month of September, 2018, for the evaluation presented in this paper. Specifically, we picked all sensor locations that are within a radius of 200 km from Frankfurt, Germany. We created a geohash grid for our SAR model and obtained images from the geohash grid for the deep learning based models, with a $\sim$50 km granularity for each geohash in the grid. Sensors were available for 80% of the grid locations. The missing data (20% of the grid locations) was obtained through interpolation, as mentioned in section III-A. We preprocessed the data obtained from all sensors to align them and windowed and averaged the data into 15 minute intervals, which gives us 96 samples (or images) per day. By evaluating on dense spatial (almost all of Germany) and temporal (a month) data, we can ensure that our models can capture a wide range of PM values. All the models were evaluated with a random 80-20 train-test split.

For most of the experiments, the history size for the data features was fixed to be 5, meaning we considered data from past five time steps (windows) to predict PM values for the next time step for each of the locations in the grid. The neighborhood size for the SAR based model was typically fixed to be 1, meaning PM quantities at a particular location are only influenced by the PM values at its immediately adjoining locations in the grid. This means that for every location in the grid, historical values of its eight immediate neighbors, including the historical values in the current location are used as features to train all of our models. We also conducted experiments to measure the effects of other history sizes when training our models and effects of a larger neighborhood when training SAR models.

In our results, we refer to the convolutional auto-encoder models as 'CAE', with a model consisting of three convolutional layers represented as 'CAE-3C', while a model

consisting of two convolutional layers is represented as 'CAE-2C'. We also include the three layer convolutional auto-encoder model that was trained with the augmented dataset and this is represented by 'Aug CAE-3C'. The size of the augmented dataset is 4x the size of the original un-augmented dataset. All CAE models were trained for 120 epochs, with a learning rate of 10^{-4}.

We first measure the time required to train each of our models, for data points from different number of days, ranging from 5 to 30 days. The results are shown in figure 4, with x-axis representing the size of training data and y-axis representing the $\ln$(training time). The SAR model has the fastest training time, followed by the CAE-2C and CAE-3C, with Aug CAE-3C taking the longest time to train. Deep neural network techniques usually take longer times to train when compared to linear regression techniques such as SAR, given that the number of variables through all the layers that they model is much larger. CAE-3C takes longer than CAE-2C given the additional number of convolutional layers. Aug CAE-3C has the longest training time since the size of the training dataset that it operates with is about four times that of the other models.

To compare model quality, we measure prediction error for each of our trained models. Figures 5 and 6 show the prediction error for all four models for predicting PM_{10} and $PM_{2.5}$ respectively. We use RMSE (Root Mean Squared Error) as the error metric. As seen from figure 5, for predicting PM_{10} values, the SAR model performs best when the dataset sizes are relatively smaller. At some crossover point, as the training data size increases, the CAE models generally start to perform better than SAR. Amongst the three CAE variants, Aug CAE-3C shows the best performance. This is in line with the data hungry nature of deep learning models. Since Aug CAE-3C has the largest training dataset of all the models, the network parameters are tuned more finely and hence show better prediction performance. The crossover point for Aug CAE-3C is around 22 days worth of data versus the crossover point for the other two CAE based models is around 27 days worth of data. While the SAR model shows about 37% better performance than the best CAE model for smaller data sizes (5 days worth of data), for larger data sizes (30 days worth of data), Aug CAE-3C shows an improvement of about 32% over the SAR model.

Figure 6 shows a similar comparison for predicting $PM_{2.5}$ values. Aug CAE-3C model is able to do a much better job in predicting $PM_{2.5}$ values across all data sizes, when compared to all other models, even though the error decreases or remains constant before increasing with larger data sizes. One reason for this could be that the variation in $PM_{2.5}$ values is typically higher than that for PM_{10} values. This is observed in figure 7. The larger dataset size in case of Aug CAE-3C helps to make it more performant than the other models in face of data variability. CAE-2C and CAE-3C show very similar performance across all data sizes and while they are slightly worse than SAR for smaller data sizes, they start showing a small prediction improvement over SAR for larger data sizes. This is again likely due to high data variability and insufficient training data and hence the number of convolutional layers do not seem to make a marked difference. Aug CAE-3C shows between 21 and 34% improvement when compared to SAR.

Next, we aim to look at a high level data distribution of the test dataset for both PM_{10} and $PM_{2.5}$ to get a more granular view of how well the different models capture the real distribution. We used 30 days worth of PM_{10} and $PM_{2.5}$ data for this experiment. Figure 7(a) shows the real distribution, distribution of the predicted values using SAR model and the distribution of the predicted values using the CAE2 model (without augmentation) of PM_{10} values. The values in all the distributions are clipped to 100 for visualization purposes (since while the distribution tail could be long, there are very few values in those buckets). The distribution of predicted values for both SAR and CAE look similar to the real values, on a high level. Both the models sometimes predict negative values even though real PM values can never be negative. A similar distribution for $PM_{2.5}$ values is shown in figure 7(b). Predictions from both the models seem to have subtle differences when compared to the real values. For smaller values, the SAR distribution seems closer to the real values, the CAE distribution seems closer to the real distribution for somewhat higher values but tends to also overestimate larger PM values, as seen by the smoother tail end when compared to the real values. Both SAR and CAE models show some variability and spillover in subsequent bins when compared to the real values. Finally, we experiment with different history sizes to be used when training our models and different neighborhood sizes in the case of SAR. It should be noted that we used convolutional filters of a fixed 3x3 size when training our CAE based models and did not experiment with this parameter for experiments in this paper, given the fairly small image sizes. Figure 8 shows the results of these experiments. Figure 8(a) shows the effect of features of varying history size (number of time steps). We tried history sizes of 1, 2, 5, 8 and 10. We chose to use 10 days worth of $PM_{2.5}$ data for this experiment and compare the prediction performance by measuring RMSE for SAR and CAE-3C, since it is generally the best performing deep learning model amongst all the variants that we tried. It is interesting to note that increasing the history size generally seems to benefit the SAR model whereas with the CAE based model, the performance improves with increasing history size but only until a certain history size. Increasing the history size beyond a particular point tends to hurt the model performance. Increasing history size in case of the CAE model implies an increase in the feature dimensionality i.e. the number of channels in the data image. This leads to an increase the amount of hidden variables the deep learning model has to learn and might explain the increasing error when the amount of training data is unable to keep up with the number of variables. Next, we measure the effect of varying the neighborhood size to be considered when training the SAR model. We measure the prediction error for both $PM_{2.5}$ and PM_{10} with neighbor sizes of 1 through 5. A neighbor size of n implies that we consider a total of

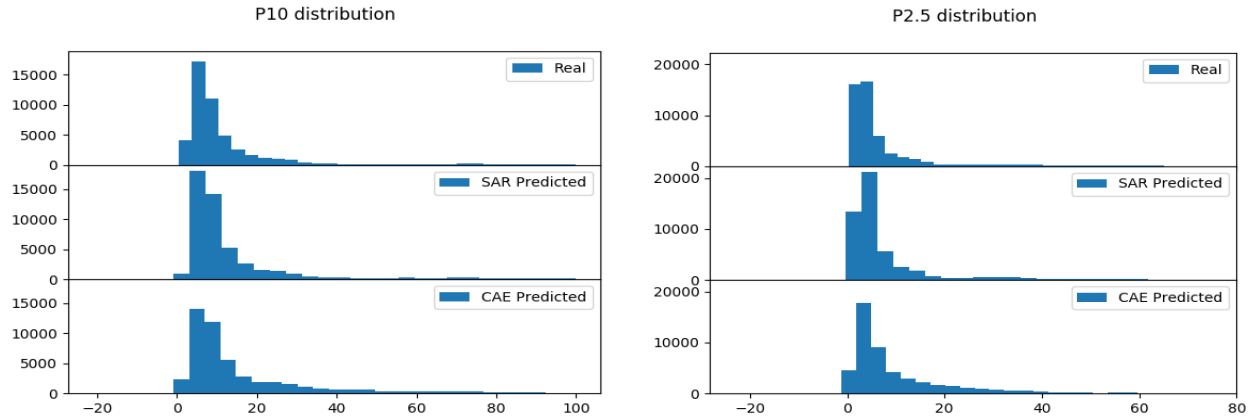
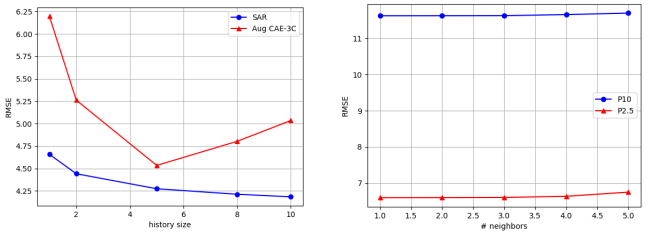


Fig. 7: PM_{10} and $PM_{2.5}$ histograms - Real, Predicted with SAR model and Predicted with Convolutional Autoencoder model



(a) $PM_{2.5}$ RMSE with different history sizes

(b) SAR prediction error with different neighbor sizes

Fig. 8: Effect of different history sizes in training SAR and CAE and effect of different neighbor sizes in training SAR

$(2*n+1)^2$ location features for the current location, including itself. Varying the neighborhood size does not seem to have any significant effect for predicting either $PM_{2.5}$ and PM_{10} and in fact, seems to deteriorate the performance when the neighborhood size increases beyond a certain value, at least in the case of the current application. It is conceivable that phenomena that have a larger spatial spread could benefit more by leveraging features from a larger neighborhood size.

V. Discussion and Future Work

In this paper, we introduced the problem of modeling and predicting particulate matter data and compared various approaches towards solving it, including the traditionally used Spatial Auto-regression model as well as a deep convolutional auto-encoder approach by converting the data into image features and labels. We compared prediction performance as well as training times across all the models, including SAR and multiple variants of the convolutional auto-encoder model. Based on the results, it is apparent that larger data sizes favor the deep learning based approaches and might be a good investment in terms of training time and resources when a large amount of historical data is available. In addition, CAE models can learn parameters such as the spatial weight matrix, W, that would otherwise need to be carefully defined by domain experts. SAR models train much faster and are more suitable when the training dataset is relatively small. With a limited amount of data, increasing the history size of the features used for training benefit a SAR approach, whereas increasing the history size beyond a certain value is not suited to deep learning models, when the training dataset is not too large. As part of our future work, we would like to experiment more with network parameters for the deep learning approaches. Since the size (height and width dimensions) of our images was relatively small, we could not try out larger sized convolutional filters or add more layers. By either increasing the spatial area that we consider or increasing the precision of our geohashing technique to create the grid, we can obtain a larger sized image to try out these tweaks and observe the effects they might have on prediction performance. While we did not observe any benefit in increasing neighborhood size for SAR models in our current experiments, it might also be interesting to measure neighborhood size effects when we have a larger spatial grid. Some of the tweaks for the deep learning techniques mentioned above would also require larger training dataset sizes to show any significant benefits. While we did see a performance penalty with an increase in history size for the auto-encoder approach, increasing the training dataset size could also offset this effect since more training examples would help in better learning the additional network parameters.

Acknowledgment

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] M. Brauer, G. Hoek, P. van Vliet, K. Meliefste, P. Fischer, U. Gehring, J. Heinrich, J. Cyrys, T. Bellander, M. Lewne *et al.*, "Estimating long-term average particulate air pollution concentrations: application of traffic indicators and geographic information systems," *Epidemiology*, 2003.

[2] L. Anselin, J. Cohen, D. Cook, W. Gorr, and G. Tita, "Spatial analyses of crime," *Criminal justice*, vol. 4, 2000.

[3] J. K. Ord and A. Getis, "Local spatial autocorrelation statistics: distributional issues and an application," *Geographical analysis*, vol. 27, 1995.

[4] C. Elbers, J. O. Lanjouw, and P. Lanjouw, "Micro–level estimation of poverty and inequality," *Econometrica*, vol. 71, 2003.

[5] L. Anselin, *Spatial econometrics: methods and models*. Springer Science & Business Media, 2013, vol. 4.

[6] H. H. Kelejian and I. R. Prucha, "A generalized spatial two-stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances," *The Journal of Real Estate Finance and Economics*, vol. 17, 1998.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, 2006.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, 2006.

[9] N. Le Roux and Y. Bengio, "Representational power of restricted boltzmann machines and deep belief networks," *Neural computation*, vol. 20, 2008.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[11] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in neural information processing systems*, 2013.

[12] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, 2015.

[13] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[14] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015.

[15] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, 2017.

[16] "EPA PM Basics," https://www.epa.gov/pm-pollution/particulate-matter-pm-basics.

[17] P. S. Kanaroglou, M. D. Adams, P. F. De Luca, D. Corr, and N. Sohel, "Estimation of sulfur dioxide air pollution concentrations with a spatial autoregressive model," *Atmospheric Environment*, vol. 79, 2013.

[18] M. Cameletti, F. Lindgren, D. Simpson, and H. Rue, "Spatio-temporal modeling of particulate matter concentration through the spde approach," *AStA Advances in Statistical Analysis*, vol. 97, 2013.

[19] R. Shad, M. S. Mesgari, A. Shad *et al.*, "Predicting air pollution using fuzzy genetic linear membership kriging in gis," *Computers, environment and urban systems*, vol. 33, 2009.

[20] P. Gupta and S. A. Christopher, "Particulate matter air quality assessment using integrated surface, satellite, and meteorological products: 2. a neural network approach," *Journal of Geophysical Research: Atmospheres*, vol. 114, 2009.

[21] X. Feng, Q. Li, Y. Zhu, J. Hou, L. Jin, and J. Wang, "Artificial neural networks forecasting of pm2. 5 pollution using air mass trajectory based geographic model and wavelet transformation," *Atmospheric Environment*, vol. 107, 2015.

[22] B. T. Ong, K. Sugiura, and K. Zettsu, "Dynamically pre-trained deep recurrent neural networks using environmental monitoring data for predicting pm2. 5," *Neural Computing and Applications*, vol. 27, 2016.

[23] X. Li, L. Peng, Y. Hu, J. Shao, and T. Chi, "Deep learning architecture for air quality predictions," *Environmental Science and Pollution Research*, vol. 23, 2016.

[24] "OK Labs Fine Dust Sensors," https://luftdaten.info/en/home-en/.

[25] "G Niemeyer. Geohash," https://en.wikipedia.org/wiki/Geohash.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015.

[27] "TensorFlow," https://www.tensorflow.org/.