# Quantifying Uncertainty in Source Term Estimation with Tensorflow Probability

Alessandro Fanfarillo
*Research Applications Laboratory*
*National Center for Atmospheric Research*
Boulder, CO, USA
elfanfa@ucar.edu

***Abstract*—Fast and accurate location and quantification of a dangerous chemical, biological or radiological release plays a significant role in evaluating emergency situations and their consequences. Thanks to the advent of Deep Learning frameworks (e.g. Tensorflow) and new specialized hardware (e.g. Tensor Cores), the excellent fitting ability of Artificial Neural Networks (ANN) has been used by several researchers to model atmospheric dispersion. Despite the high accuracy and fast prediction, regular ANNs do not provide any information about the uncertainty of the prediction. Such uncertainty can be the result of a combination of measurement noise and model architecture. In an urgent decision making situation, the ability to provide fast prediction along with a quantification of the uncertainty is of paramount importance. In this work, a Probabilistic Deep Learning model for source term estimation is presented, using the Tensorflow Probability framework.**

***Index Terms*—Probabilistic Deep Learning, Source Term Estimation, Urgent Decision Making, Bayes, Machine Learning, Edge Computing**

## I. Introduction

In catastrophic scenarios, such as wildfires, hurricanes, tsunamis, earthquakes, floods, and chemical/radioactive attacks, rapid prediction capabilities are essential to limiting the loss of human lives and economic damages. Traditional High Performance Computing (HPC) techniques have a long history of successfully simulating disasters, but they can hardly be used to face unexpected emergencies. HPC machines are expected to be less reliable in the future [16] and this limits the ability to guarantee time constraints. Furthermore, most of the data preprocessing and postprocessing workflows are currently not designed to run efficiently and thus they represent a bottleneck to making urgent and critical decisions. Although cloud computing represents a valid solution to mitigate the lack of reliability of traditional HPC platforms, a stable and reliable internet connection may be difficult to obtain in a catastrophic situation.

Thanks to the presence of modern specialized hardware designed for AI tasks (e.g. Intel Movidius Myriad 2), System-on-Module (SOM) (e.g. Nvidia Jetson), and System-on-Chip (SOC) architectures (e.g. Raspberry Pi), it is now possible to perform a fair amount of computation in a reasonable amount of time at *the edge* of the network [17], close to where the data is gathered. Putting computing at the proximity of data sources has several benefits compared to the traditional cloud-based computing paradigm: lower and more reliable latency and response time [5], [10], [22], network bandwidth and energy savings, better reliability, privacy and security.

The combination of the new hardware with AI-based algorithms has the potential to produce accurate (although approximate) results useful for urgent decision making, using a fraction of the energy required by a traditional HPC machine, within predictable time constraints and without needing an internet connection.

In recent years, Artificial Neural Networks (ANN) have been successfully used for atmospheric dispersion modeling and real-time estimation problems [3], [11]–[13], [18]. As shown by Wang et al. [19], well trained ANNs can provide fast and high accuracy predictions, even in the presence of complex topology.

An ANN-based source term estimation model running on a small device, such as a Raspberry Pi, can be very useful in war scenarios to quickly detect the source of chemical, bacteriological, and radioactive releases on the field, where power and internet connectivity are barely present. The same algorithm can also be used to monitor large gas pipelines and quickly detect and identify ruptures that might endanger people and nearby structures.

Although powerful, classic ANNs do not provide any information about the uncertainty of a prediction. Risk-Cost-benefit analysis in emergencies often requires the use of probabilistic approaches in order to quantify the uncertainty coming from forecasts about the intensity of a hurricane, the spreading of a wildfire or the impact of a flood. [6], [8], [20], [21].

The goal of this work is to provide a proof-of-concept for a neural network-based source term estimation model able to provide a prediction along with uncertainty. For this purpose, Probabilistic Deep Learning and the Tensorflow Probability library have been used.

## II. Probabilistic Deep Learning

Regular ANNs commonly used in Deep Learning can be seen as sets of units, called neurons, organized in layers, usually connected one after the other. This layer-wise architecture of the neural network is also called a *feed-forward* network. The connections between neurons are not equally important, and *weights* are used to represent the strength of the connection between neurons belonging to two different layers. The weights of the ANN are *learned* from the data

978-1-7281-5995-9/19/$31.00 ©2019 IEEE
DOI 10.1109/UrgentHPC49580.2019.00006

using a two-step approach: forward propagation and backward propagation.

After the training phase, the weights in the ANN have a fixed and deterministic value; meaning that if we feed the network with the same input two times we will get the same output both times. When ANN are applied to physical problems involving measurements, the presence of uncertainty (e.g. malfunctioning sensors, measurements errors, model limits) may significantly impact the model prediction.

Probabilistic Deep Learning, also called Bayesian Deep Learning, tries to include the concept of uncertainty in ANN by seeing the output of the ANN (and the weights) as random variables. Assuming that a normal variable is representing the output of the ANN, the immediately previous layer of the ANN will be in charge of parameterizing the mean (and possibly the variance) of the normal random variable. Because the output of the ANN is now a distribution, the loss function to be used can be a negative log likelihood, since the training phase is now seen as finding the parameters of the distribution that explain the data. In [2] the authors provide a detailed description of how the backward propagation can be done on a Bayesian Neural Network. After the training phase of a Bayesian Neural Network, in order to get a prediction, the output of the network should be "sampled" and the mean of the sampling will represent the prediction.

Having a distribution as an output of a neural network gives us the opportunity to run all sorts of statistical methods used to quantify the uncertainty of the prediction.

### A. Aleatoric and Epistemic Uncertainty

Uncertainty can be classified into two categories: Aleatoric and Epistemic.

Aleatoric uncertainty is known also as *statistical uncertainty* and it usually represents the "noise" in the data. If we are using a sensor for our measurements, it will introduce noise in the data given by the imperfections of its structure. Every aspect of the measurement that introduces some sort of random component is aleatoric uncertainty. Aleatoric uncertainty can be reduced by improving the quality and reliability of the measurements (e.g. better sensors), as well as using more data.

Epistemic uncertainty is known also as *systematic uncertainty* and it represents how good the model is in representing the phenomenon that we want to predict. A too simplistic model cannot represent a complex phenomenon, not even with an infinite amount of noise-free data. Too little or missing data used during training, ignoring an important feature, or the presence of underrepresented situations are all potential causes of high epistemic uncertainty. Assuming to have a model powerful enough to represent a certain phenomenon, the epistemic uncertainty can be reduced by providing more complete data. Epistemic uncertainty is really important for safety-critical applications, because epistemic uncertainty is required to understand examples which are different from training data. In simpler words, when a model provides a prediction along with an estimate of the epistemic uncertainty it is telling us how reliable that particular prediction is.

The variance of random variables expresses the uncertainty related to the estimation of the mean value. Because Bayesian Neural Networks have random variables as output layer, to get a numerical output of mean and variance, the output distribution should be "sampled" several times.

For the Bayesian Neural Networks mentioned in Sec. II, the aleatoric uncertainty can be quantified by adding one more neuron to the layer in charge of parametrizing the output distribution, to compute mean and variance of the normal distribution. Higher variance in the output represents higher aleatoric uncertainty, which can be mitigated by reducing the noise in the dataset.

To quantify the epistemic uncertainty, the weights of the neural network should be transformed into random variables as well. If the model is appropriate to represent a certain phenomenon and the dataset is representative, the variance associated with the weights used for the prediction will be small because small variations in the weights will not impact the output. On the other hand, if an instance given as input to the neural networks has never been or has rarely been seen during the training, the variance associated with the weights will be higher.

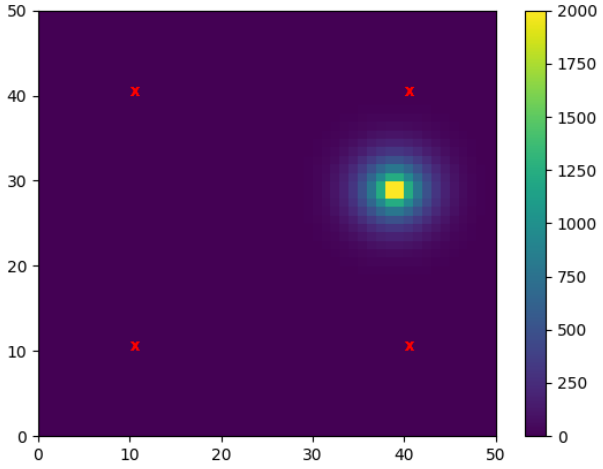
## III. Methodology

In order to test the performance of a Bayesian Neural Network for the source estimation problem, a simple 2D heat equation simulation has been used for the preliminary evaluation.

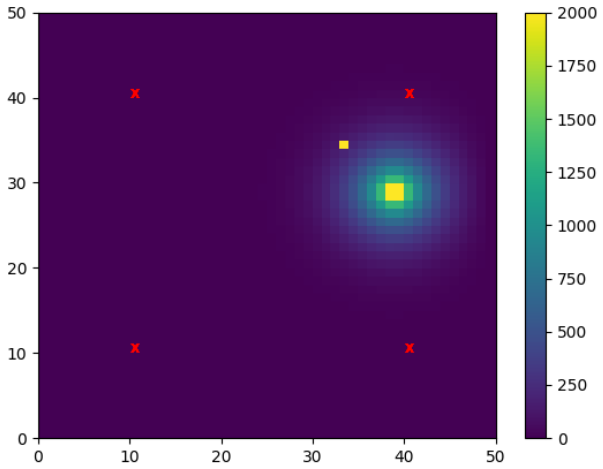
The simulation environment consists of a square plate of thermally conductive material with unit thermal conductivity constant. On the plate, a network of four static sensors is placed close to the four corners of the plate. In Fig. 1, the red x represents a sensor and the yellow 2x2 square with a round halo represents the heat source that is supposed to be estimated by the ANN. For the sake of simplicity, the estimation of the source consists only in the location. The concentration/heat intensity is considered constant.

Among all the possible choices of ANN architectures, we decided to adopt a well-known kind of Recurrent Neural Networks (RNN) called Long Short-Term Memory (LSTM) networks [9]. These networks have been successfully used in speech recognition tasks and, more generally, when the sequence of the inputs should be kept into consideration in order to make a good prediction [14], [15]. The inputs to the neural network are the temperature values read by the sensors at each timestep; the output consists of two numbers representing the X and Y coordinates of the source. Because the sequence of data is relevant for the prediction, the input data during training and testing should be organized as a time series, with a certain time window collecting the temperature values detected over the last n time steps. For this particular case, the time window has been fixed to 32 timesteps per sample. This means that the network will start making predictions at each timestep, starting at timestep 32, when there will be enough data available to fill the time window. In the visual output produced by the simulation, the source estimation is

represented by a 1x1 yellow point, visible in Fig 1b. Figure 1a shows the state of the simulation at timestep 30, right before having the first estimations.




(a) After 30 time steps




(b) After 64 time steps

Fig. 1: Visual output of source estimation

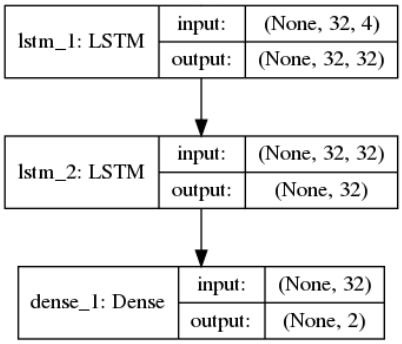
During the simulation, there is a convergence towards the source in about 300 timesteps.

The training set consists of 100 simulations, 1000 dt long, where the source is placed in a random location on the grid. The output expected by the network is a two-elements long 1D array, where the elements represent the x and y coordinate of the source. During the training phase, the output never changes.

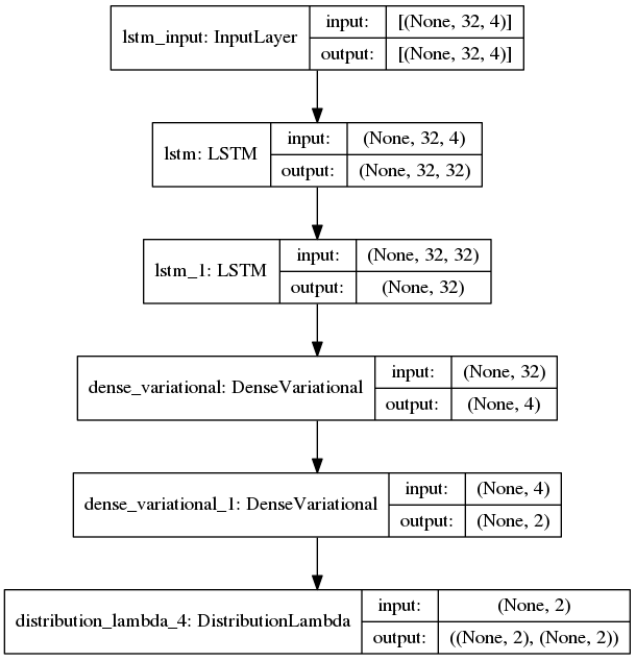
Given the synthetic nature of the scenario used to assess the performance of the bayesian neural network, only the epistemic uncertainty is taken into consideration. In fact, no noise or uncertainty is introduced by the sensors. In order to highlight the effect of the epistemic uncertainty on the final prediction, we decided to create an unbalanced training by placing the sources only on a subset of the whole surface. The predictions coming from this unbalanced training are compared with the predictions coming from a uniform training set.

### A. *Bayesian Neural Networks with Tensorflow Probability*

A simple Tensorflow/Keras [1], [4] model able to correctly identify the location of a heat source in our synthetic scenario is shown in Fig. 2a.




(a) Regular LSTM model




(b) Combination of LSTM and DenseVariational (bayesian) layers

Fig. 2: Model graphs for regular and bayesian networks

In order to quantify uncertainty in neural networks, a regular neural network can be transformed into a Bayesian neural network by using the layers provided by a probabilistic library: Tensorflow Probability [7]. The library includes a wide selection of probability distributions, tools to build deep probabilistic models (including probabilistic layers), variational

inference and Markov Chain Monte Carlo. In Fig. 2b it is shown that the output layer is no longer a tensor but a normal distribution, and the final output consists of 2 parts: the mean and standard deviation of the output distribution for the x and y coordinates.

Since we are interested in assessing the epistemic uncertainty related to the prediction, we need a way to quantify the variance of the model (not the data) given certain inputs. In order to do that, the weights of a few layers of the neural network should be seen as random variables instead of fixed values. This is shown in Fig. 2b by the two DenseVariational layers following the regular LSTM layers. The DenseVariational layers model the weights connecting the hidden units as normal random variables; during the training, the parameters of the distribution of each weight are estimated. During the inference, if the input data is somewhat familiar to the model, the variance of the weights in the DenseVariational layer will be relatively small. On the other hand, if the data has never been seen during training, or if the model is not capturing the phenomenon correctly, the input to the DenseVariational layer will be considered unusual and the variance of the weights will be relatively high.

Because the weights and output of the Bayes Neural Networks are distributions, a sampling step is required. In this work, we decided to sample from the model 100 times and compute the mean and standard deviation of the results.

### B. Test Description

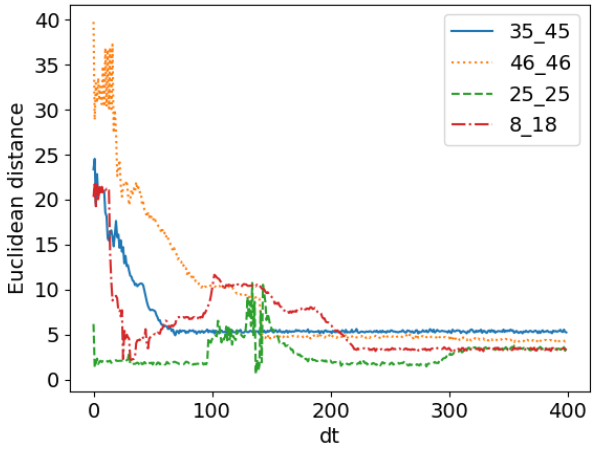
As already mentioned in Sec. II-A, assessing epistemic uncertainty is very important for safety-critical applications and urgent decision making. In fact, it gives us a measure of how confident the model is in making a certain prediction. Assuming to have an appropriate model to represent the phenomenon under study, epistemic uncertainty is higher when the input data has never, or has rarely seen during the training phase. In order to show how the Bayesian neural network depicted in Fig. 2b is able to quantify epistemic uncertainty, we trained the model using two different training sets: 1) a "uniform" training set, consisting in 100 simulations where the source is placed randomly anywhere on the plate; 2) a "Top-right corner missing" training set, consisting in 100 simulations where the source can be placed anywhere on the plate, except in the 20x20-wide top-right corner of the plate.

The model is then asked to produce a prediction when the source is placed in one of the following four locations: 35_45, 46_46, 25_25, and 8_18. The first digit represents the x coordinate of the source, the second represents the y coordinate. When the model is trained on the "top-right corner missing" dataset and is asked to produce a prediction for data coming from source 35_45 or 46_46 (both in the top-right corner of the plate), it is exposed to data never seen during training and thus an inaccurate prediction, accompanied by a high standard deviation on the model weights should be expected. On the other hand, when the model is trained on the "uniform" dataset, the predictions for any of the four testing sources should be good with relatively small standard deviation.

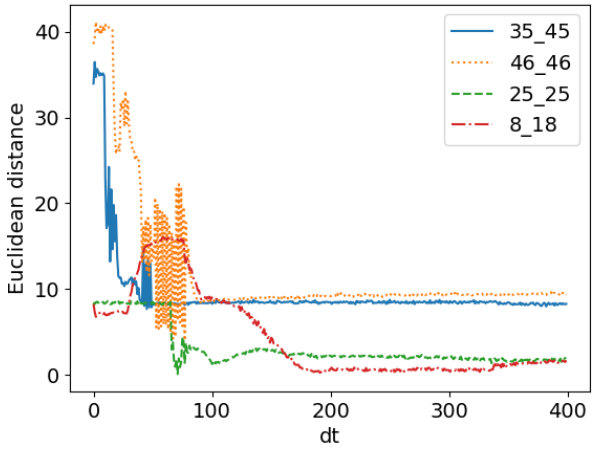
## IV. Results

All the tests described in Sec. III-B have been run on Casper, a heterogeneous system equipped with Intel Xeon Gold 6140 (Skylake processors) at 2.3 GHz and 768 GB of memory per node. The models use python-3.6.4, Tensorflow-1.14, and Tensorflow-probability-0.7.0.

In order to measure the accuracy of the prediction of the source location was, we used the Euclidean distance between the estimated source location at each time step and the real location (e.g. 46_46). In Fig. 3 it is shown that the prediction for sources placed in the top-right corner is worse than others when the training set did not include sources placed in that particular area (Fig. 3b). On the other hand, the model performs equally well when the training set is uniform across the whole area (Fig. 3a).



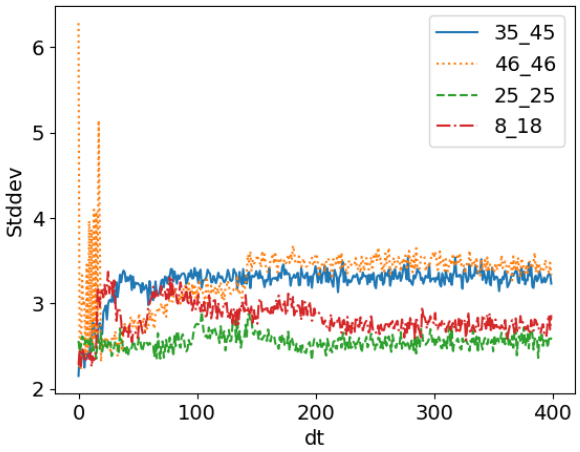

(a) Uniform Training Set



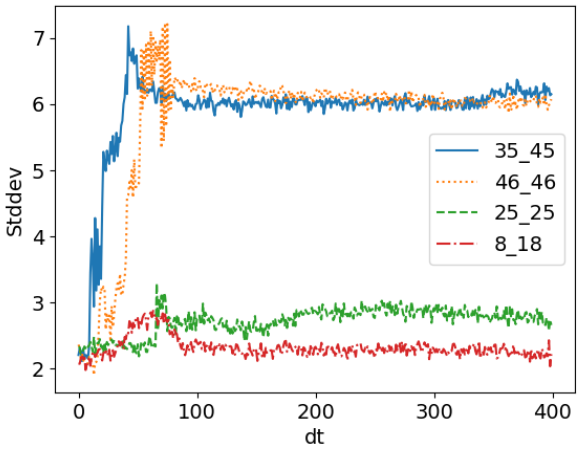

(b) Top-Right Corner missing in training set

Fig. 3: Euclidean Distance between prediction and real source

Fig. 4 shows how the standard deviation of the model changes over time. Fig. 4b shows the standard deviation produced by the model trained on the "top-right corner missing" dataset; it is evident that the standard deviation for the prediction of sources inside the top-right corner (46_46 and 35_45) is way higher than for the other sources. On the other hand, Fig. 4a shows that the standard deviation is more similar among all the sources when the model is trained uniformly.




(a) Uniform Training Set




(b) Top-Right Corner missing in training set

Fig. 4: Standard Deviation from model sampling

## V. Conclusion and Future Work

Despite their huge modeling power, regular neural networks do not provide information about the uncertainty related to the prediction. This lack of information makes neural networks not suitable in urgent decision making, where a probabilistic cost-benefit analysis is usually used. In this work, a Bayesian neural network-based source term estimation example is presented. The model is able not only to provide a pretty accurate location of a heat source in a synthetic environment, but also a measure of the confidence of the prediction. As future work, we plan to focus on a multi-source term estimation model with aleatoric uncertainty quantification.

## References


[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *arXiv e-prints*, page arXiv:1505.05424, May 2015.

[3] Marija Boznar, Martin Lesjak, and Primoz Mlakar. A neural network-based method for short-term predictions of ambient so2 concentrations in highly polluted industrial areas of complex terrain. *Atmospheric Environment. Part B. Urban Atmosphere*, 27(2):221 – 230, 1993.

[4] François Chollet et al. Keras. https://keras.io, 2015.

[5] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[6] Murray Dale, Jon Wicks, Ken Mylne, Florian Pappenberger, Stefan Laeger, and Steve Taylor. Probabilistic flood forecasting and decision-making: an innovative risk-based approach. *Natural Hazards*, 70(1):159–172, Jan 2014.

[7] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matthew D. Hoffman, and Rif A. Saurous. Tensorflow distributions. *CoRR*, abs/1711.10604, 2017.

[8] Mark Finney, Isaac C. Grenfell, and Charles W. McHugh. Modeling Containment of Large Wildfires Using Generalized Linear Mixed-Model Analysis. *Forest Science*, 55(3):249–255, 06 2009.

[9] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] K. Kumar and Y. Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4):51–56, April 2010.

[11] Denglong Ma and Zaoxiao Zhang. Contaminant dispersion prediction and source estimation with integrated gaussian-machine learning network model for point source emission in atmosphere. *Journal of Hazardous Materials*, 311:237 – 245, 2016.

[12] A. Pelliccioni and T. Tirabassi. Air dispersion model and neural network: A new perspective for integrated models in the simulation of complex situations. *Environmental Modelling Software*, 21(4):539 – 546, 2006. Urban Air Quality Modelling.

[13] Domagoj Podnar, Darko Korain, and Anna Panorska. Application of artificial neural networks to modeling the transport and dispersion of tracers in complex terrain. *Atmospheric Environment*, 36(3):561 – 570, 2002. Seventh Internatioonal Conference on Atmospheric Science and Appl ications to Air Quality (ASAAQ).

[14] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *arXiv e-prints*, page arXiv:1402.1128, Feb 2014.

[15] Hasim Sak, Andrew W. Senior, and Franoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*, 2014.

[16] Osman Sarood, Esteban Meneses, and Laxmikant V. Kale. A 'cool' way of improving the reliability of hpc machines. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 58:1–58:12, New York, NY, USA, 2013. ACM.

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, Oct 2016.

[18] Bing Wang, Bingzhen Chen, and Jinsong Zhao. The real-time estimation of hazardous gas dispersion by the integration of gas detectors, neural network and gas dispersion models. *Journal of Hazardous Materials*, 300:433 – 442, 2015.

[19] Rongxiao Wang, Bin Chen, Sihang Qiu, Liang Ma, Zhengqiu Zhu, Yiping Wang, and Xiaogang Qiu. Hazardous source estimation using an artificial neural network, particle swarm optimization and a simulated annealing algorithm. *Atmosphere*, 9(4), 2018.

[20] John C. Whitehead. One million dollars per mile? the opportunity costs of hurricane evacuation. *Ocean Coastal Management*, 46(11):1069 – 1083, 2003.

[21] Margaret Carlin Anne World Bank Kreimer, Alcira Arnold. *Building Safer Cities*. The World Bank, 2003.

[22] S. Yi, Z. Hao, Z. Qin, and Q. Li. Fog computing: Platform and applications. In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, pages 73–78, Nov 2015.