DeepER: A Deep Learning based Emergency Resolution Time Prediction System

Gissella Bejarano, Adita Kulkarni, Xianzhi Luo, Anand Seetharam, Arti Ramesh
Department of Computer Science, SUNY Binghamton, USA
(gbejara1, akulka17, xluo22, aseethar, artir)@binghamton.edu

***Abstract*—Accurately predicting resolution time for emergency incidents is crucial for public safety and smooth functioning of cities as it helps in planning resources that will be available for immediate assistance. In this paper, we present DeepER, a deep learning based emergency resolution time prediction system that predicts future resolution times based on past data. DeepER is an encoder-decoder based sequence-to-sequence model that uses Recurrent Neural Networks (RNNs) as the neural network architecture. The basic cell in DeepER is a Long Short-Term Memory (LSTM) cell. We perform experiments on the NYC Emergency Response Incidents data provided by NYC Open Data. We compare the performance of the model with ARIMA and Linear Regression using two metrics— Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). DeepER achieves an average performance improvement of 3% and 16% with respect to RMSE and 10% and 27% with respect to MAE over ARIMA and Linear Regression, respectively.**

I. Introduction

A number of emergency incidents (e.g. fire, building collapse, etc.) are reported on a regular basis in cities around the world. It is important that city officials are able to allocate sufficient resources to ensure public safety, smooth functioning of cities and address such incidents in a timely manner. To engage the public in coordinating these emergency response services smoothly, governments and city officials have made such data openly available to everyone. By adopting a data-driven approach, cities can efficiently allocate resources, plan prudently, and thus minimize the loss to human life and property and improve resolution time.

One of the major challenges in this regard is estimating the resolution time for emergency events in the future. While emergencies are unpredictable by nature and often happen unexpectedly, it is possible to leverage past resolution time data to predict the resolution time of future events. This is because the nature of the event (e.g., fire), the extent of damage, and the number of personnel and equipment available on site are keys factors that dictate the total time needed to address the issue. For example, if multiple emergencies occur in a colocated manner, then it is likely that the time needed to address each of these issues will be higher than usual because of the division of resources. Therefore, if a data-driven analysis suggests that resolution time for future events will be higher than a desired value for a particular incident type, then this analysis can provide insights into budget spending, personnel hiring, and resource allocation.

Hence, in this paper, we design **DeepER**, a deep learning based emergency resolution time prediction system that predicts the future resolution time of incidents based on historical data. We consider three important emergency incident types, namely, *Fire*, *Law* and *Structural*. We model emergency resolution time prediction as a time series prediction problem. At the core of DeepER there is a sequence-to-sequence encoder-decoder neural network architecture. Both the encoder and the decoder in DeepER are Recurrent Neural Networks (RNNs) and the basic cell is an LSTM cell. The encoder receives the previous resolution times as input and encodes them into a hidden context vector. This hidden vector is given as an input to the decoder, which generates future resolution times.

To evaluate the performance of DeepER, we perform extensive experiments on the publicly available NYC Emergency Response Incidents dataset [1]. We use the data for a period of approximately eight years for the three incident types. This dataset is challenging from the perspective of time series analysis and prediction because emergency events by nature occur at random times (i.e., lack periodicity), have limited correlation to each other and may not follow seasonal trends. Because of this reason, we design DeepER as a sequence-to-sequence model so that it can unearth the dependencies among the data points in the sequence even though the time period between two consecutive events in the sequence is varying. DeepER leverages these hidden patterns in data to make superior predictions.

We compare the performance of DeepER with two widely used baselines— Linear Regression and Auto Regressive Integrated Moving Average (ARIMA). We use two metrics to evaluate the models— Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). DeepER achieves an average performance improvement of 3% and 16% with respect to RMSE and 10% and 27% with respect to MAE over ARIMA and Linear Regression, respectively. Our results demonstrate that DeepER is a practically viable system that provides superior prediction performance and can be used to aid city planning and management. We conclude the paper with a discussion of some of the insights we obtain while conducting this investigation.

The rest of the paper is organized as follows. In Section II, we present related work. We present the dataset and discuss the problem investigated in this paper in Section III. We then describe the DeepER system in Section IV and the implementation details in Section V. We present experimental results in Section VI and discuss some of our insights from this work in Section VII.

II. Related Work

With the growth and development of smart cities and cyber-physical systems, a variety of machine learning approaches have been adopted to address different problems in these domains [2], [3], [4], [5], [6], [7]. In this section, we first present work related to assisting the operations of emergency and non-emergency services and then discuss prior research related to smart cities.

In the recent years, a number of research papers have adopted data-driven approaches to aid the functioning of emergency and non-emergency services in cities. For example, DeFazio et al. use Gaussian Conditional Random Fields (GCRFs) to predict response times of non-emergency 311 calls in NYC [8]. The authors in [9] adopt a rolling forecast model to predict the number of emergency calls based on the number of 911 calls in NYC. Similarly, the authors analyze NYC non-emergency call requests and present a Random Forest model to predict the number of requests [10].

In [11], authors analyze the intra-region temporal correlation and the inter-region spatial correlation of data collected from NYC and build a framework to predict the number of crimes for certain regions. Similarly, the authors propose a neural network based continuous conditional random field model for fine-grained crime prediction in Chicago and NYC [12]. The potential of deep learning models for a variety of time series prediction tasks has also been explored recently. For example, deep learning models have been adopted for emergency event prediction in [13]. Similarly, the authors use LSTM based deep models for gas consumption and occupancy detection using WiFi beacons in [14] and [15], respectively. Recurrent Neural Network (RNN) based encoder-decoder models similar to the one designed in this paper have also been used for prediction problems in a variety of different domains. For example, such models have been used for water consumption, gym center occupancy, wireless channel quality and air pollution prediction [16], [17], [18], [19].

In contrast to existing work, we design DeepER, a deep learning model to predict the resolution time of emergency services and validate the efficacy of the model using the emergency incidents response data from NYC collected over a period of approximately eight years.

III. Data

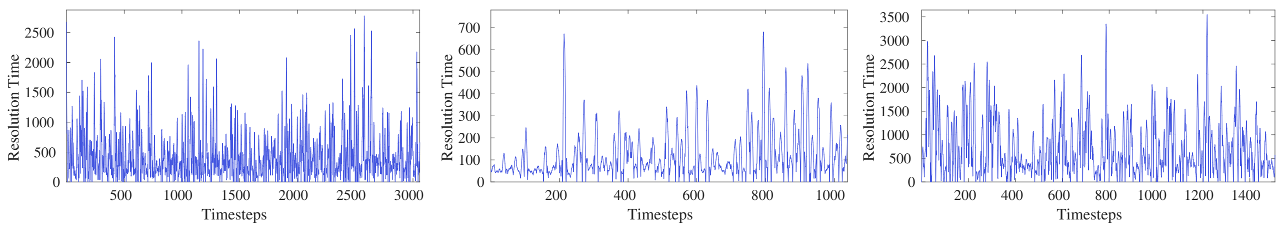
We use the emergency response incidents data from NYC Open Data provided by the Office of Emergency Management [1]. We use around 8 years of data starting May 2011 to December 2019. The dataset consists of 13 incident types with 7 attributes each. We use three attributes from the dataset — Incident type, Creation date, and Close date. In this study, we focus on three of the most important and frequent emergency types — *Fire*, *Law*, and *Structural*. We calculate the *resolution time* for each incident by subtracting the creation date from the close date and converting it to minutes. Figure 1 shows the resolution times for these incidents. We observe that the time required to resolve events related to *Law* is the least followed by *Fire* and *Structural*, respectively. We also observe from Figure 1 that there is significant variation in resolution time for events belonging to the same incident type.

Additionally, we observe from the data that each of these three incident types have multiple subtypes, which we extract from the type description. *Fire*, *Law*, and *Structural* have 52, 29, and 73 subtypes, respectively. Figure 2 shows the incident subtypes that contribute the most events for each of the most general incident types. We observe that *2nd alarm*, *Suspicious Package (denoted as Package)*, and *Collapse* are the subtypes that account for the highest number of events in *Fire*, *Law*, and *Structural*, respectively.

A. Preprocessing

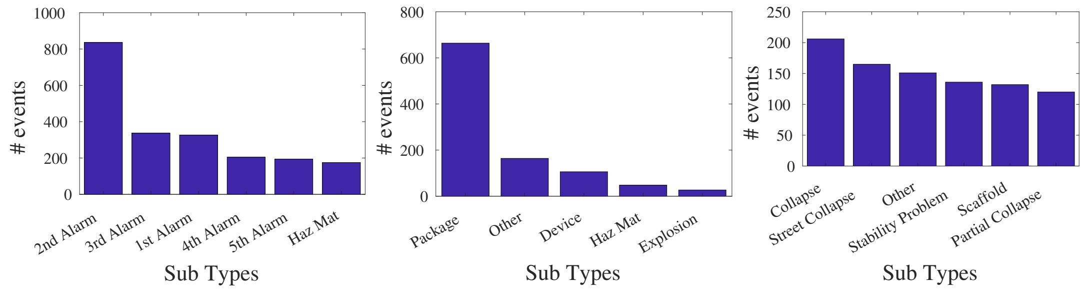
As is the case with most data-driven solutions to real-world problems, the first step involves pre-processing the data to identify missing values and outliers. We observe that the dataset contains non-trivial number of missing values for events. A missing value is encountered when an event does not have a valid close date. In the entire dataset, we observe that *Fire*, *Law*, and *Structural* have 32%, 12%, and 23% missing values, respectively. For each incident type, we replace these missing points by sampling from the actual distribution of the remaining points. To determine the actual distribution for each incident type, we fit the data to more than 80 different distributions. We perform the Kolmogorov-Smirnov goodness of fit test (KS test) and use the *p-value* of the KS test to pick the best distribution for the dataset under consideration.

We also observe that the dataset contains some outliers—values that are significantly different from the rest of the data points. By studying the values, we believe that such values might be the result of manually closing some unfinished entries at a later date. For example, we observe some extreme outliers in the dataset (greater than 100,000 minutes). We



(a) Fire (b) Law (c) Structural

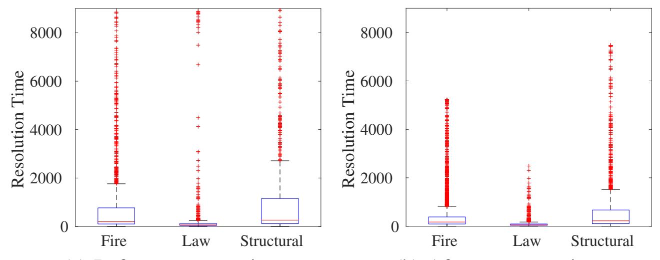
Fig. 1: Trends in datasets



(a) Fire (b) Law (c) Structural

Fig. 2: Incident Types

identify outliers as those points whose resolution time is greater than the quantile 90 of that incident type. For *Fire*, *Law*, and *Structural*, we observe that there are 7%, 9%, and 8% outliers, respectively. Figure 3 shows the distribution of the three different incident types before and after preprocessing. We observe from the figure that the raw dataset has a large number of outliers. We once again replace these outliers by sampling from the distribution of the valid data points. In comparison to Figure 3a, we observe that Figure 3b presents a significantly refined distribution.



(a) Before preprocessing (b) After preprocessing

Fig. 3: Datasets before and after preprocessing

We observe some other interesting issues in the dataset. We observe that for *Fire* and *Structural* most of outliers are located in the first few years of the dataset. In comparison, most of the missing points are located during the last few years. Additionally, for *Fire* and *Structural* we observe larger resolution times during the initial years than the last few years. However, the opposite is true for *Law*, where the resolution times during the initial years is lower than the resolution times in the later years.

IV. Problem Statement and Model

In this section, we first discuss the future resolution time prediction problem studied in this paper and then describe DeepER, a deep learning based system that predicts future resolution time based on past data.

A. *Problem Statement*

Our aim is to design a system that accurately predicts future resolution times of incidents from historical data. For this purpose, we cast the problem as a time series prediction problem. We consider a sequence of n events with resolution times X = x_1 , x_2 ..... x_n, and predict the resolution time of the next k events Y= y_1, y_2, ... y_k. What makes this problem challenging and different from classic time series prediction problems is that though these events occur chronologically, the actual time elapsed between two consecutive events varies. This is because each event corresponds to an emergency (i.e., unplanned) and thus the time when it occurs is completely random. Hence, in some cases one may have considerable time between two consecutive events, whereas in other cases multiple events can occur in a short duration of time. Therefore, our goal is to design a flexible model that examines the resolution time of a sequence of prior events and predicts the resolution time of future events and does

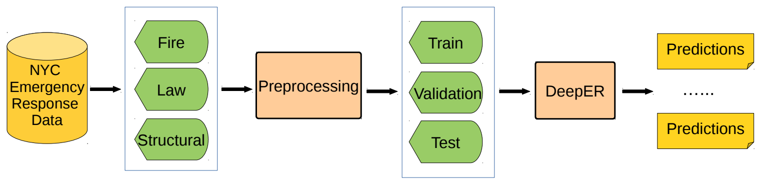


Fig. 4: DeepER System Overview

not depend on the actual time frame in which the events occurred.

B. *DeepER System Details*

In this subsection, we describe DeepER, a sequence-to-sequence based encoder-decoder model that considers the resolution times of a sequence of prior events to predict the resolution time of future events. Figure 4 provides an overview of the DeepER system. DeepER consists of a data preprocessing block that splits the data by incident types and prepares the training, validation, and test datasets (details in Section V). The preprocessing block also replaces the outlier and missing values according to the steps outlined in Section III-A. The system then presents the data sequences as input to the deep learning model that uses them to generate the predictions.

1) Sequence-to-Sequence Models: Before delving into the system details, we discuss the appropriateness of sequence-to-sequence models for the emergency resolution time prediction problem studied here. In comparison to classic statistical regression models, sequence-to-sequence models are better suited for this problem as they map entire input sequences to output sequences and do not just focus on capturing simple trends in the data. Additionally, as the points in the dataset are not equally spaced in time and the events lack a seasonal and periodic behavioral pattern, we design deep learning based sequence-to-sequence models because such models are capable of learning the underlying dependencies and correlations in the data using an interconnected neural network architecture during the training phase and are especially useful when the dependencies are not apparent and cannot be easily defined. The trained model leverages this knowledge to make accurate predictions at test time by considering the current input sequence.

2) Encoder-decoder based RNN Model: DeepER consists of two components, an encoder and a decoder as shown in Figure 5. Both the encoder and the decoder comprise of recurrent neural networks (RNN). An RNN is a network of neural nodes that are arranged in layers. Internally, the RNN has a hidden state h_t that is updated at each time step t using the input x_t and the previous hidden state h_{t-1}. At each time

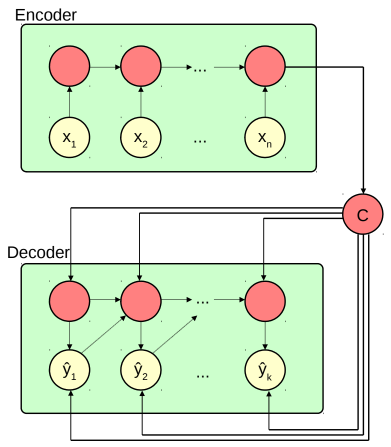


Fig. 5: Encoder-decoder based RNN

step t, the hidden state of the RNN is given by,

$$h_t = \phi(h_{t-1}, x_t) \quad (1)$$

where, ϕ is any non-linear activation function and $1 \leq t \leq n$.

As we can observe from Figure 5, the encoder takes an input sequence x_1, x_2, ..., x_n that corresponds to the resolution time of events for the last n time steps. The encoder then generates a hidden encoded vector C. After the entire input has been processed, the summary C is provided as input to the decoder. The decoder then generates $\hat{y}_1$, $\hat{y}_2$, ..., $\hat{y}_k$, the predicted resolution times for the future k events. The loss function used is the sigmoid activation function and it is applied to the output of the decoder. This ensures that the predicted values are in the [0-1] range.

The basic cell structure used in the encoder is LSTM that captures the important dependencies in the data. LSTM cells also possess the ability to 'forget' that enables them to overcome well-known vanishing/exploding gradient. To achieve this an LSTM cell has three main gates—input, output, and forget. The input gate receives the pertinent information in the current step and the output gate determines the hidden state for the next step. The forget gate is responsible for discarding unimportant information so that the model can capture the relevant long-term dependencies. We refer the reader to [20], [21] for additional details.

V. Implementation Details

In this section, we discuss implementation details regarding training, validation, and testing as well as important design decisions (e.g., hyper-parameter selection).

From our discussion in the Section III-A, we observe that missing points and outliers are not spread uniformly throughout the duration of the dataset. Additionally, the entire duration of the dataset is approximately nine years and therefore, we observe gradual changes in the average resolution times of events for the same incident type. We attribute these variations to possible changes adopted by the different emergency management agencies. These issues inherent to the dataset necessitate some important design decisions. As the underlying characteristics of the data change over time, if we adopt a simple approach and split the data chronologically into training and test, then we will end up training solely on the data for the initial few years and testing on the last few years. This is unlikely to provide good performance because the distribution of the test data sequences are different from the distribution of the training sequences. Hence, we adopt a more careful approach where we ensure representation of data from each year in training, validation, and test datasets.

Sequence Length	Learning Rate	Units in Hidden Layer
10-3	0.01	10
10-3	0.001	50
10-3	0.0001	100
15-5	0.01	10
15-5	0.001	50
15-5	0.0001	100

TABLE I: Hypararameter combinations for experiments

To do so, we divide the entire dataset into eight periods: seven periods of one year each and one period of approximately one and half year, approximately. We split each of these years in training, validation, and test sets following the usual percentages of 50%, 25%, and 25%, respectively. This ensures that the training, validation, and test sets contains data from all the years. Additionally, to remove any form of seasonal dependencies that may exist in the dataset, we permute the split order of training, validation and test within each year. Such permutation ensures that the training, validation, and test data contain samples from all months of the year; in the absence of such reordering the training data will be confined to primarily the first few months, the validation confined to the middle months, and test containing data from the last few months of each year. In addition to helping in achieving good prediction performance across the entire duration of the dataset across the years, this important pre-processing step also makes our model more readily extensible to real-world deployment as the model is trained on the different variations that may be present in the data.

A. Training, Validation, and Testing

We use Pytorch to implement the deep learning model. We train our models on a Linux machine with 8-core Intel i7 processor and 64 GB RAM. We use a sliding window approach with a stride of 1 to transform the time series into instances of sequences of length n and prediction of length k. We use unguided training as the training methodology. In this approach, the previous predicted output is used by the decoder as an input to the next step of the decoder during both training and test. Unguided approach is likely to provide better results at test time because it allows greater exploration of the state space during training. The loss function used to guide the training is the mean squared error.

During the training phase, we experimented with various hyperparameters and then finally decided upon 6 hyperparameter combinations that are best suited for our dataset (Table I). A sequence length of 10-3 in Table I means that the model takes 10 points are input and predicts 3 points into the future. For each hyperparameter combination, we iterate over 100,000 epochs, saving the state of the model after every 5000 iterations. We then select the particular model combination that provides the best performance on the validation set.

For quantifying the best performance, we do not simply pick the model with the lowest loss. Such an approach usually works well when the data has overall less variation and exhibits seasonality. However, in our dataset, events occur at random times and are of varying intensity (as each event is an emergency), thus resulting in higher variation among the values. This makes our dataset challenging to predict, resulting in the model adopting a safe approach and predicting values close to the mean value. Therefore, in addition to the loss function, we also consider another heuristic, the magnitude of the standard deviation among the predictions on the validation set, to select the best model. This approach ensures that the trained model provides superior quantitative and qualitative performance. Additionally, testing on a validation set and selecting from the myriad of combinations ensures that we do not overfit the model to the training dataset.

VI. Experimental Results

In this section, we compare the performance of DeepER with two baselines: i) Linear Regression and ii) Auto-Regressive Integrated Moving Average (ARIMA).

Linear Regression is a statistical model that fits the best straight line to the given data.

ARIMA is a statistical model that has three components — AR (autoregressive term), I (differencing term), and MA (moving average term), specified by the parameters p, d, and q, respectively. We use the *pmdarima* toolkit in python for our experiments, which picks the optimal combination of the parameters for the input data.

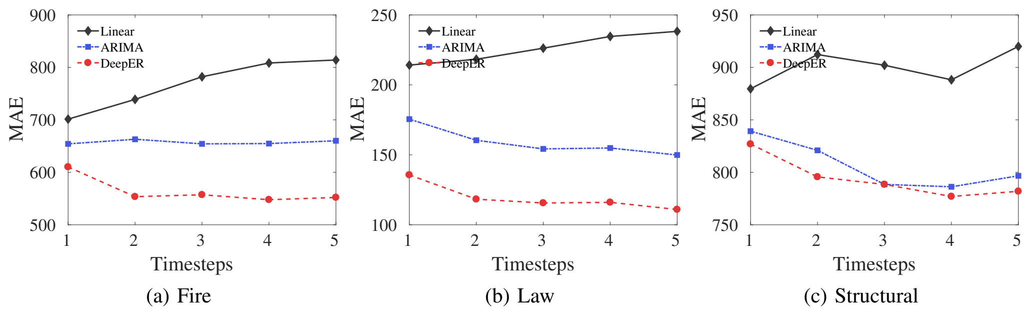


Fig. 6: MAE Results for the 15-5 setting

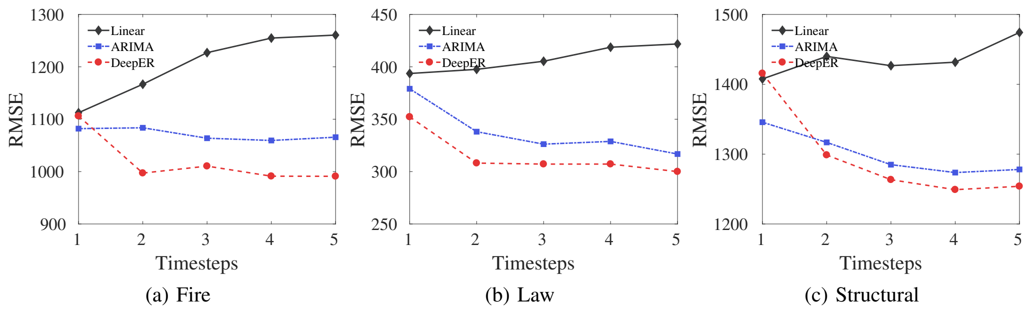


Fig. 7: RMSE Results for the 15-5 setting

We use two well-known metrics for evaluation—Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Equations 2 and 3 show how they are calculated, where y_{ij} is the i^{th} test sample for j^{th} time step, $\hat{y}_{ij}$ is the predicted value of y_{ij}, and h is the total number of test samples.

$$RMSE_j = \sqrt{\frac{\sum_{i=1}^{h} (\hat{y}_{ij} - y_{ij})^2}{h}} \tag{2}$$

$$MAE_j = \frac{\sum_{i=1}^{h} |\hat{y}_{ij} - y_{ij}|}{h} \tag{3}$$

A. *RMSE and MAE*

In this section, we discuss the RMSE and MAE results for all the models. Table II shows the average RMSE and MAE results for sequence lengths 10-3 and 15-5, where the average performance is calculated over 3 and 5 time steps, respectively. Recall that a sequence length 10-3 means that the model takes 10 events as input and predicts 3 events as output into the future. We see that DeepER outperforms the baselines with respect to both MAE and RMSE for all incident types on average for the 15-5 setting. For the 10-3 setting, DeepER outperforms both baselines for *Fire* and *Law*. However, for *Structural*, it outperforms Linear but not ARIMA. Therefore, from Table II, we observe that DeepER provides overall better performance for the 15-5 setting.

Figures 6 and 7 shows the MAE and RMSE results for the three incident types for the 15-5 sequence setting as it provides the best predicitons. We observe that DeepER significantly outperforms the baselines with respect to both MAE and RMSE. The only exception is the one step prediction for *Structural* where DeepER exhibits poor performance. Interestingly, from the figures, we observe that DeepER is able to better predict the resolution time of events further into the future. This is in contrast to most time series prediction problems where the prediction performance deteriorates as the model predicts further into the future. The primary reason behind this behavior is that the data points in our dataset correspond to emergency events and hence lack seasonality, strong correlation, and trends. DeepER is still able to generate better predictions for our challenging problem than the baselines because of its sequence-to-sequence behavior that maps entire input sequences to output sequences and ability to glean complex underlying dependencies in the data that are not apparent.

Incident Type	**Model**	**10-3**		**15-5**	
		MAE	**RMSE**	**MAE**	**RMSE**
Fire	Linear	786	1256	769	1204
	Arima	660	1087	657	1071
	DeepER	**584**	**1069**	**564**	**1019**
Law	Linear	186	381	226	407
	Arima	133	323	159	338
	DeepER	**116**	**309**	**119**	**315**
Structural	Linear	965	1504	900	1436
	Arima	**818**	**1299**	806	1300
	DeepER	839	1307	**794**	**1296**

TABLE II: Average RMSE

VII. Discussion

In the previous section, we demonstrated that DeepER provides superior prediction performance than the baseline models. In this section, we discuss some additional learnings from our exploration of this dataset.

A. *Qualitative Results*

We next discuss the qualitative prediction performance of DeepER as well as the baselines. Figure 8 shows the 1-step prediction performance for DeepER, ARIMA, and linear regression for *Fire*. From the figure, we observe that all models struggle to predict the values accurately. From our experience of working with similar models in the past [16], [17], [18], we have observed that sequence-to-sequence models are generally able to make really superior predictions. This does not appear to be the case always for this prediction task primarily because of the challenging non-periodic and non-seasonal nature of the emergency events dataset.

We observe from Figure 8 that the resolution times for some events is significantly higher in comparison to majority of the points. Because of this pattern, any prediction model will find it difficult to accurately predict such high peaks. But, despite this challenging nature of the dataset, we observe that DeepER provides a significantly smoothened prediction performance in comparison to the baselines and accurately predicts the underlying pattern. If we overlook the peaks, we can see that the prediction performance of DeepER for the remaining data points is good.

In comparison, we observe that the next step predictions for both ARIMA and Linear Regression closely mirror the actual resolution time of the previous time step. This occurs because both these baselines only use the past trend to predict the future. This is the root cause behind their poor performance because the resolution time of the current request is significantly different from the previous one.

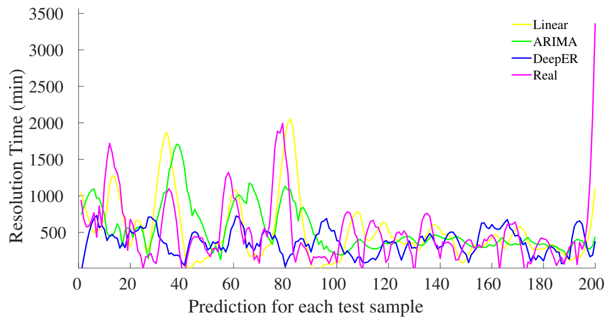


Fig. 8: Fire: One step Real vs Predicted Performance

B. *Further Insights into Data Preprocessing*

As mentioned in Section III-A, we ensure that the training, validation, and test data include sequences from all years of the dataset. While cross-validation is commonly used to establish the significance of the results, we design this well-crafted split of the dataset to render greater credibility to our results primarily because of the fairly limited number of data points. Additionally, as noted earlier, the dataset contains a non-trivial number of outliers and missing values. Table III shows the number and percentage of missing and outlier points in each of the training, validation and test sets if the data is split chronologically. This uneven distribution of the outlier and missing values further necessitates a carefully constructed split to ensure that training, validation, and test sets contain a more uniform distribution of such points. We note that while the dataset is relatively small in size, each event corresponds to an emergency and therefore, it is crucial to use all available data points and generate superior predictions because such predictions are critical to improving human safety.

Incident Type		**Training**	**Validation**	**Testing**
Fire	Total	1529	764	764
	Outliers	8%	6%	5%
	Missing	14%	47%	53%
Law	Total	519	260	260
	Outliers	8%	6%	13%
	Missing	2%	17%	27%
Structural	Total	754	377	377
	Outliers	8%	6%	13%
	Missing	2%	17%	27%

TABLE III: Statistics of Outliers and Missing values in a Chronological Split

C. *Limitations of Enriching DeepER*

We observe from Section III that each incident type consists of multiple subtypes. We attempt to perform resolution time prediction at the subtype level, but realize that because this is an emergency events dataset, the number of data points is not sufficient for training, validation, and testing of deep learning models for each subtype separately. We also use these subtypes as features in DeepER, but observe that this enhanced model did not improve prediction performance. We believe that dearth of data at the subtype level is the primary reason behind it not contributing to DeepER's prediction performance.

D. *Practicality of DeepER*

With the increase in computational power over the last decade, deploying deep learning based systems to solve real-world problems is becoming relatively easy. As is the case with most deep learning models, DeepER requires some computational time for training. However, once trained, DeepER requires limited amount of time to generate predictions, a desired attribute in a practical system. Additionally, as more data becomes available, DeepER can be easily retrained thus enabling it to adapt to changing situations. We anticipate DeepER to be retrained at comparatively infrequent intervals (i.e., only when significant number of new emergency events have been resolved).

VIII. Conclusion and Future Work

In this paper, we presented DeepER, a deep learning based emergency resolution time prediction system that predicts future resolution times based on past data. We performed experiments on the NYC Emergency Response Incidents data provided by NYC Open Data. We compared the performance of DeepER with ARIMA and Linear Regression using two metrics— Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). DeepER achieved an average performance improvement of 3% and 16% with respect to RMSE and 10% and 27% with respect to MAE over ARIMA and Linear Regression, respectively. We also draw upon important learnings and insights from the data, which can be utilized for designing deep learning models for data in the emergency response domain and other related domains where the data can lack an overt predictable trend. As part of our future work, we plan to extend this analysis to other cities so that it gives greater validity to our results. We want to also engage with city officials so that DeepER can be adopted to aid the planning and preparation of city emergency response systems.

References

[1] "Nyc open data," https://data.cityofnewyork.us/Public-Safety/Emergency-Response-Incidents/pasr-j7fb.

[2] I. Fox, L. Ang, M. Jaiswal, R. Pop-Busui, and J. Wiens, "Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 1387–1395. [Online]. Available: http://doi.acm.org/10.1145/3219819.3220102

[3] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, April 2015.

[4] L. Peng, L. Chen, Z. Ye, and Y. Zhang, "Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, pp. 1–16, 07 2018.

[5] Y. Guan and T. Plötz, "Ensembles of deep lstm learners for activity recognition using wearables," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 2, pp. 11:1–11:28, Jun. 2017. [Online]. Available: http://doi.acm.org/10.1145/3090076

[6] W. Cheng, Y. Shen, Y. Zhu, and L. Huang, "A neural attention model for urban air quality inference: Learning the weights of monitoring stations," 2018. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16607

[7] A. A. Varamin, E. Abbasnejad, Q. Shi, D. C. Ranasinghe, and H. Rezatofighi, "Deep auto-set: A deep auto-encoder-set network for activity recognition using wearables," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MobiQuitous '18. New York, NY, USA: ACM, 2018, pp. 246–253. [Online]. Available: http://doi.acm.org/10.1145/3286978.3287024

[8] D. DeFazio, A. Ramesh, and A. Seetharam, "Nycer: A non-emergency response predictor for nyc using sparse gaussian conditional random fields," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MobiQuitous '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 187–196. [Online]. Available: https://doi.org/10.1145/3286978.3287010

[9] A. Chohlas-Wood, A. Merali, W. Reed, and T. Damoulas, "Mining 911 calls in new york city: Temporal patterns, detection, and forecasting," in *AAAI Workshop: AI for Cities*, 2015.

[10] Y. Zha and M. M. Veloso, "Profiling and prediction of non-emergency calls in nyc," in *AAAI 2014*, 2014.

[11] X. Zhao and J. Tang, "Modeling temporal-spatial correlations for crime prediction," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM 17. New York, NY, USA: Association for Computing Machinery, 2017, p. 497506. [Online]. Available: https://doi.org/10.1145/3132847.3133024

[12] F. Yi, Z. Yu, F. Zhuang, and B. Guo, "Neural network based continuous conditional random field for fine-grained crime prediction," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, 2019, pp. 10–16.

[13] B. Cortez, B. Carrera, Y.-J. Kim, and J.-Y. Jung, "An architecture for emergency event prediction using lstm recurrent neural networks," *Expert Systems with Applications*, vol. 97, pp. 315 – 324, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417308606

[14] N. Pathak, A. Ba, J. Ploennigs, and N. Roy, "Forecasting gas usage for big buildings using generalized additive models and deep learning," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2018, pp. 203–210.

[15] B. Qolomany, A. Al-Fuqaha, D. Benhaddou, and A. Gupta, "Role of deep lstm neural networks and wi-fi networks in support of occupancy prediction in smart buildings," in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2017, pp. 50–57.

[16] G. Bejarano, A. Kulkarni, R. Raushan, A. Seetharam, and A. Ramesh, "Swap: Probabilistic graphical and deep learning models for water consumption prediction," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 233242. [Online]. Available: https://doi.org/10.1145/3360322.3360846

[17] A. Kulkarni, A. Seetharam, and A. Ramesh, "Deepfit: Deep learning based fitness center equipment use modeling and prediction," in *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ser. MobiQuitous 19. New York, NY, USA: Association for Computing Machinery, 2019, p. 394403. [Online]. Available: https://doi.org/10.1145/3360774.3360803

[18] A. Kulkarni, A. Seetharam, A. Ramesh, and J. D. Herath, "Deepchannel: Wireless channel quality prediction using deep learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 443–456, 2020.

[19] V. Reddy, P. Yedavalli, S. Mohanty, and U. Nakhat, "Deep air: Forecasting air pollution in beijing, china," 2018.

[20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.