Heatmap.Live: Crowd Mobility Tracking and Forecasting using IoT Sensors and Transformers

Steven Gao
hgao@exeter.edu

Phillips Exeter Academy

September 2020

Abstract

Crowd counting and mobility tracking supply valuable insights into public safety, event organization, and urban planning, in a time when the COVID-19 pandemic renders the demand for a reliable and privacy-preserving crowd tracking system even more exigent. In this paper, we develop a scalable Internet-of-Things sensor network that captures IEEE 802.11n Wi-Fi probe requests, a proven indicator of crowd activity. We then propose a data processing framework utilizing Google's Transformer architecture with multi-headed attention to generate forecasts on upcoming mobility patterns. Conducting comprehensive experiments using self-collected and public data sets, our model supports a large forecast horizon and outperforms existing models by a wide margin.

Keywords: crowd counting, mobility, IoT sensor, Wi-Fi probe request, deep learning, time series forecasting, multi-headed attention, universal transformer, urban planning

Contents

1 Introduction

Overcrowding is a prevalent issue in a modern-day society where populations are highly concentrated – busy streets of urban areas, college campuses, shopping malls – presenting serious challenges in management and planning to administrators, and, in the meantime, compromising the convenience and safety of people traveling in or occupying such spaces. A well-designed crowd counting and mobility tracking system alleviates this pressure and provides insights on an administrative level, which could benefit public safety, event organization, urban planning, etc.

In light of the COVID-19 pandemic, the demand for a reliable crowd counting system has come into prominence when coincided with social distancing protocols that diminish the capacity of many public spaces and creates uncertainty in pedestrian traffic, businesses and schools struggle in their reopening and disease control efforts. Accurate forecasting of crowd mobility bears even more significance as administrators can take preventive measures to limit, divert, or redirect upcoming traffic flows.

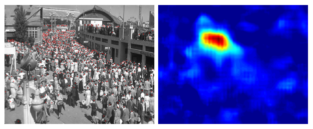


Figure 1: Crowd counting with CNN (Convoluted Neural Network) [1]

Conventionally, crowd counting is achieved through video cameras and image recognition a methodology with inherent shortcomings, which is the case demonstrated in Figure 1. However, many would resent deploying security cameras on a large scale as they undermine personal privacy, and the limited range indoors mandates that more cameras are installed to cover a specific area. Environmental conditions such as low-light and, for outdoor usage, unexpected weather all impact the performance and accuracy of vision-based systems. Image recognition and the computing resource thereof needed, combined with expensive video cameras, portray it as a costly and unsuitable solution for many use cases.

The challenge remains for constructing a reliable crowd counting system as several factors could potentially undermine the usability of such systems developed, including accuracy, coverage, cost, and privacy preservation. Whereas researchers have explored thoroughly video cameras and image recognition (e.g. [21], [11], [12], [1]), fewer have explored utilizing radio signals to determine crowd density.

The widespread and amplitude of commodity Wi-Fi devices elevates the usefulness of Wi-Fi traffic to a new level. With few exceptions where mobile devices are prohibited, almost all pedestrians now carry IEEE 802.11n Wi-Fi-capable devices such as their smartphones and tablets. Devices broadcasts probe requests to gather information on available access points nearby [5]. Probe requests could serve as an indicative metric of crowd activity at a given location.

Wi-Fi-based crowd counting addresses the limitations of vision-based systems, as the typical coverage ranges from 20 to 50 meters [14]. Wi-Fi bases systems can efficiently determine individuals' presence through counting the MAC (media access control) addresses embedded in the probe request. To address the population without smartphones, the count can simply be extrapolated by a scalar to fit the actual circumstance. The sensors cost less to manufacture and operate in all weather and lighting conditions.

Crowd monitoring alone could render little assistance without a robust forecasting system that can inform administrators of upcoming dynamics. Previously, researchers have experimented with RNNs (Recurrent Neural Networks) [18], [13]. However, these models have inherent drawbacks – they struggle to capture long-range context dependencies, encounter issues like gradient vanishing and explosion while training, and are unable to process input in parallel, areas in which the Transformer shines.

The counter these aforementioned shortcomings and problems, we propose an interconnected network of sensors, built upon the Raspberry Pi platform. To approximate the crowd size and yield meaningful data, sensors need to be non-intrusive yet highly precise. We then develop an API server that supports distributed computing to collect and process input data from the sensors. Finally, we construct a model based on Google's Transformer architecture, outlined in Attention is All You Need [16]. First, we test the model on data sets self-collected on the campus of Phillips Exeter Academy, and then compare our model with existing LSTM based models using a publicly available data set. We study the results of these experiments and demonstrate the efficacy of the model.

The main contributions of our work can be summarized as:

1. We design a scaleable and affordable IoT sensor network especially fit for usages on campuses and urban areas;

2. We propose a server framework that efficiently processes millions of input from the sensor network;

3. The Transformer-based model we constructed yields higher accuracy and better performance than existing LSTM models.

The following sections are organized as such: Section 2 examines the previous work in the two fields of interest – crowd counting methodologies and time series forecasting with machine learning, Section 3 details our design of different part of the system, Section 5 puts the system to test in real-world settings and examines the results in detail, Section 6 concludes the paper and discusses prospective studies that can be conducted in this direction.

2 Related Works

2.1 Crowd Counting

2.1.1 Vision-Based Crowd Counting

Driven by big data and machine learning, a variety of crowd counting techniques have drawn attention from researchers and industry practitioners. Researchers have long delved into the possibility of using camera vision to calculate crowd counts, and while many techniques rely on face recognition of individuals and aggregating that result [14], Mikel Rodriguez et al. was among the first to propose associate density information with individual positions in the image [9]. Lokesh Boominathan, Srinivas S S Kruthiventi, and R. Venkatesh Babu employed a combination of deep and shallow CNN to capture body and facial features in combination with low-level features of blobs and clusters [1]. Although this research betters the performance of the image recognition model, other weaknesses of this method have yet to be challenged.

2.1.2 Wi-Fi Based Crowd Counting

As researchers explore other methodologies in crowd counting, Wei Xi et al. were among the first to investigate 802.11n Wi-Fi CSI (channel state information) by the rationale that CSI is highly sensitive to disruptions in the environment [20]. The investigation was furthered by Simone Di Domenico et al., who analyzed the shape of the Doppler spectrum. Han Zou et al. optimized this approach by selecting the most representative feature of CSI and applying a kernel transfer learning kernel learning to adjust for the environmental and temporal discrepancies [22]. However, two sensors are required to be placed on two opposite ends to capture the activity in an allocated space in the approach with CSI, and experiments with groups larger than a dozen people have yet to be seen. Furthermore, CSI is highly dependent on the placement of the sensors, and change in the layout of the space could require re-fitting the data.

Wi-Fi probe requests came into spotlight with the work of J. Scheuner et al. and E. Vattapparamban et al. Probe requests are much less sensitive to alterations in the environment, expanding the coverage as a single sensor, equipped with a high-gain antenna, could capture signals within a large radius. Though probe requests cannot capture detailed activities and body gestures as CSI could, they are perfectly adequate for gathering crowd sizes. Processing probe requests is less strenuous compared to that of CSI as the unique MAC addresses can be aggregated to the sum of people.

2.2 Multivariate Time Series Forecasting

Many sequence DNN (Deep Neural Networks) models strive to tackle the challenge of time series forecasting, in which the input of a model is a time-dependent sequence and so is the output. Although NLP (Natural Language Processing) models enjoyed massive success in recent years, there have been incremental advancements in multivariate time series forecasting even though the two problems share many commonalities.

2.2.1 Recurrent Neural Network

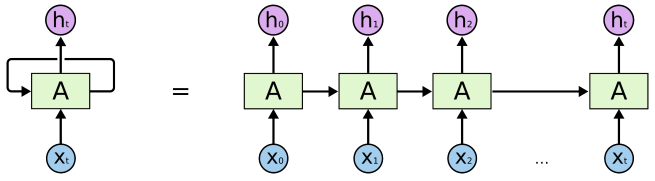


Figure 2: Typical RNN Cell [15]

RNN is apt to process sequential data like $(x_1, \ldots, x_n)$ where t is the time step index and $x_t \in R^d$. A typical RNN cell, whose structure once unrolled resembles that of lists and sequences, is shown in Figure 2. The RNN cell takes x_t as an input, and the output h_t is calculated from the input x_t and the previous hidden state. The cell runs the same function recurrently for every element in the list, allowing information from previous time steps to pass through.

2.2.2 Long-short Term Memory

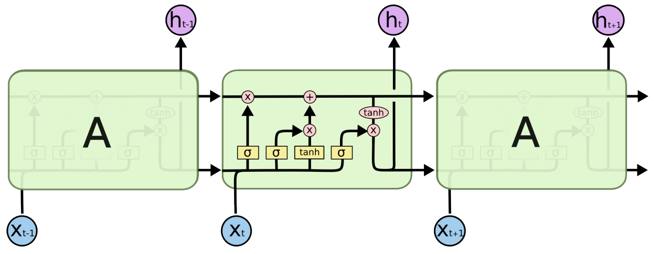


Figure 3: LSTM Cell [15]

When training recurrently over long sequences, typical RNN experiences gradient explosion and gradient vanishing, which LSTM mediates [6]. In contrast to a typical RNN cell, an LSTM cell has three cell gates that allow it to selectively add and remove information from the cell state, as illustrated in Figure 3.

Nonetheless, while addressing some deficiencies on long-range dependencies, LSTM still has its limitations. The hidden state of a time step strongly influences only the following few time steps, resulting in a challenge to capture temporal dependencies, which is critical in time series forecasting. For instance, the crowd count from 24 hours ago might be just as indicative of as that of the last hour. Due to the nature of LSTM as a Recurrent Neural Network, data is passed in sequentially, and the next time step could only be calculated from the hidden state of the previous one, rendering parallelism an impossible task.

2.2.3 Transformer

Breaking these constraints of LSTM, Ashish Vaswani et al. proposed a new architecture in deep learning, namely Transformer, which features an encoder-decoder structure and two attention mechanisms: self-attention and encoder-decoder attention. Since it processes all time steps in parallel, Transformer outperforms LSTM in long-range context dependencies while drastically improving the efficiency.

3 Sensor Network Architecture

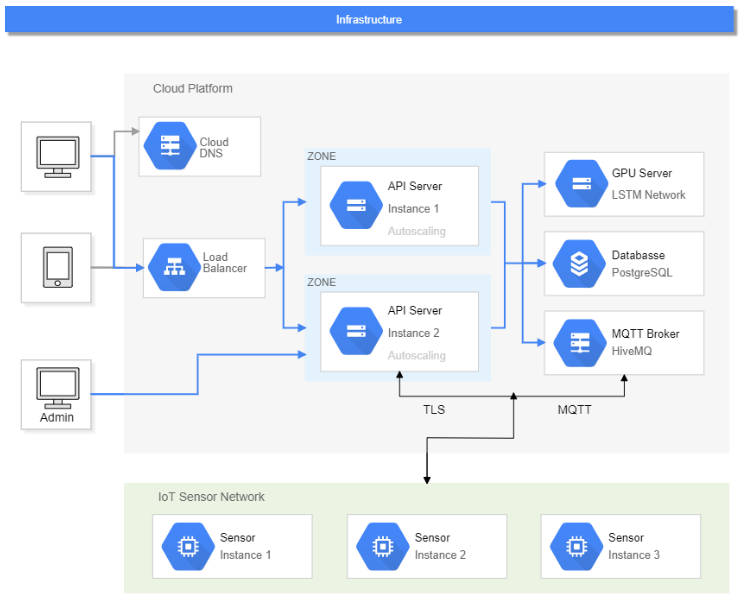


Figure 4: System Architecture

As illustrated in Figure 4, the infrastructure of the system includes three main components: the IoT sensor network, an API server that collects and processes the data, and a Transformer-based neural network that generates forecasts of upcoming traffic dynamics.

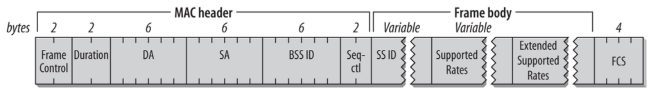


Figure 5: Structure of the Probe Request Frame [5]

3.1 Crowd Counting with IoT Sensors

3.1.1 Preliminaries: Wi-Fi Probe Request

Under the 802.11n protocol, Wi-Fi devices send out probe requests to gain information on available wireless networks nearby.[5] Probe requests are broadcast at various frequencies that are dependent on the operating system of the device. On average, smartphones send out 55 probe requests per minute and 2000 per hour in an experimental study conducted in 2015 [4].

Wi-Fi 802.11n, a standard that the smartphones use, operates on 2.4GHz with 14 channels. [5] The wireless module is set to monitor mode, and our software processes the probe requests it receives. Due to limitations of available hardware, the wireless module can only listen to one channel at a time, so we iterate all of the channels on a Weighted Round Robin basis, during which the most active channels are prioritized. We extract the mac address and signal strength from a probe request, whose structure is shown in Figure 5, and upload the data collected periodically.

To exclude stationary Wi-Fi devices located near the sensor, devices that are always present over an extended period will be excluded from calculations.

3.1.2 Hardware Architecture

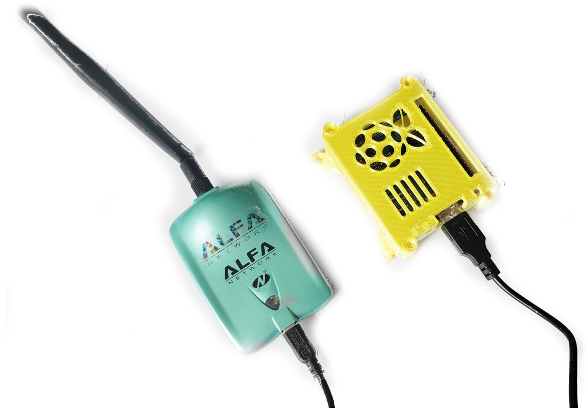


Figure 6: A Sensor Node Prototype

The IoT network consists of an array of sensor nodes, each has capabilities to monitor the traffic in the region where it is placed. To ensure expandability and robustness, the sensor should run on a network operating system equipped with a runtime environment, such as Linux distributions, for our self-developed software. To monitor nearby Wi-Fi traffic, the sensor should be able to interact with a Wi-Fi module that supports 802.11n monitor mode. [19].

We select Raspberry Pi Model 3 A+ [2] as the microprocessor of the sensor node, as it features a 64-bit SoC (System on a Chip) processor operating at 1.4GHz, 512 megabytes of RAM, and built-in Wi-Fi to report data to the server. Enclosed in a custom 3D-printed case, the Raspberry Pi communicates with an external Wi-Fi module through standard USB protocol. The Wi-Fi module is connected to a high-gain omnidirectional antenna to increase the effective range. An optional cellular module can be connected to the Raspberry Pi to allow communication in the absence of Wi-Fi coverage.

3.1.3 Software Architecture

The sensor node software is responsible for capturing data from the external antenna and upload the data to the server through either Wi-Fi or cellular network. Kali Linux[7] is selected as the operating system, for it provides driver support for packet capturing.

Written in Java, the sensor node software listens exclusively to probe requests and demonstrated predictable behavior and performance across different devices and environments. The sensor nodes communicate with the API server through two secured channels: MQTT over UDP and HTTP 1.2 with TLS encryption. The sensor defaults to using TLS and falls back to MQTT, a lightweight publish-subscribe protocol suitable for transmission over less ideal network environments, reporting its operational status as well as locally hashed mac addresses.

3.2 Data Collection and Processing

The API server is run on a cloud server to collect and process the data it receives from the sensor network. Integrated with a web user interface, the API server is written in php using the framework Laravel [8] to allow easy configuration for administrators. MySQL provides high-performance database service to the API server.

The API server computes crowd sizes and upcoming mobility patterns and records the data into the database and sends information to mobile applications where users can view real-time traffic and forecasts.

4 Mobility Forecasting with Transformer-Based Model

4.1 Data Pre-processing

Raw sensor data is processed per the following procedures.

1. Probe requests with a locally administered MAC address are removed as smart devices sometimes randomize their MAC addresses to protect privacy.

2. Devices located outside of the sensor bounding box is eliminated. The distance of a device from the sensor is calculated using the signal strength of the probe request using Free-Space Path Loss, defined as

$$FSPL(dB) = 20log_{10}(d) + 20log_{10}(f) + C \tag{1}$$

   where d is distance and f is frequency.

3. Stationary devices at the location that are not smartphones, laptops, or other portable electronics, are filtered out.

4. The raw data entries are grouped by an interval of 5 minutes and aggregated into the total number of unique MAC addresses, effectively the number of Wi-Fi devices, during that interval.

5. Then we generate the Data Sets 1 to 4 listed in Table 1 for training and testing using a sliding window of a fixed lookback period and a forecast horizon.

4.2 Transformer-Based Model

Illustrated in Figure 8, our Transformer-based crowd forecasting model retains the origin encoder-decoder design in Figure 7. Originally designed for NLP, we tailor the Transformer model to fit our specific use case.

4.2.1 Encoder

The encoder accepts a sequence $\big(x_{t-(L-1)}, \ldots, x_t\big)$ where $x_t \in \mathbb{R}^d$, in which d is the number of features and L represents the lookback period on historic data. The periodicity of the sequence is extracted in the One_hot encoding before the sequence is fed into a trainable Embedding Layer that formulates a representation of the time step in a global scope. Positional encoding is added for the Transformer cell to learn temporal features and dependencies within the sequence. A typical Transformer encoder block then accepts the processed sequence and passes it through multi-headed self-attention and a feed-forward network to the decoder block.

4.2.2 Decoder

Similarly, the decoding side accepts a sequence $\big(x_t, \ldots, x_{t+(H-1)}\big)$ where $x_t \in \mathbb{R}^d$, in which H represents the forecast horizon. The sequence serves as input for the decoder and is processed the same as it was for the encoder. The decoder has a similar design to the encoder with two attention mechanisms – self-attention and encoder-decoder attention. The self-attention mechanism

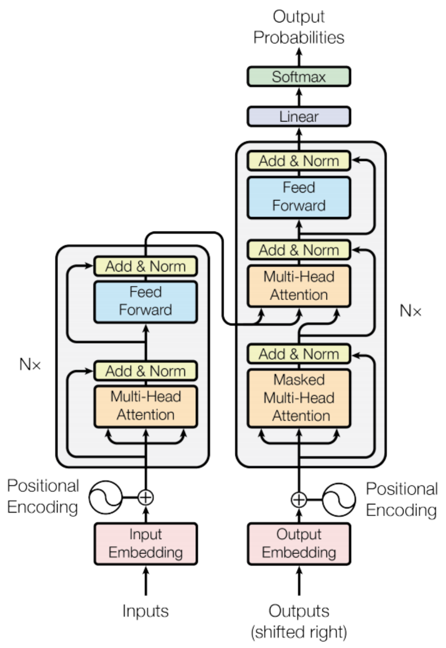


Figure 7: The Original Transformer Architexture

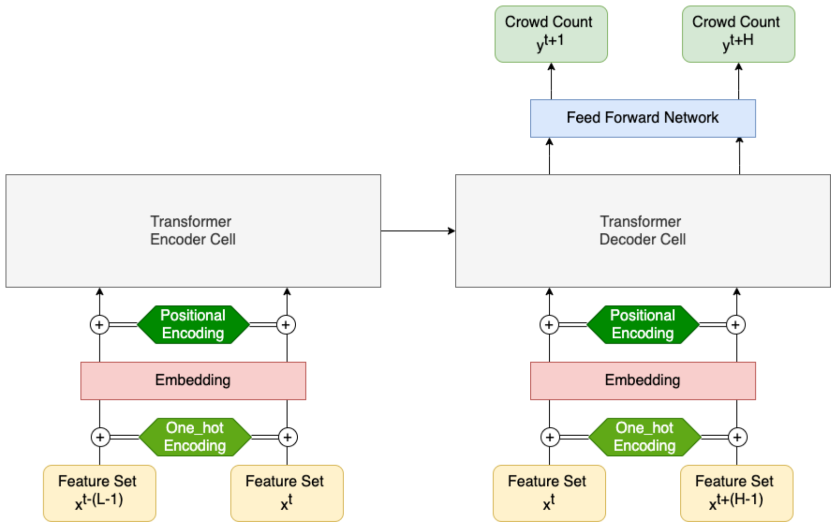


Figure 8: Our Transformer-Based Model

works similarly to that of the encoder, and the encoder-decoder attention block takes the concatenated input from both the encoder block and the decoder input. The decoder runs on an auto-regressive basis, which means for time step t the decoder takes the generated output of y^{t-1} as input.

4.3 Maintaining Sequence Order

4.3.1 Sequence-wide Positional Encoding

The attention mechanism considers each time step as equal if without positional encoding, and the time-dependent information would be lost as a result. In Attention is All You Need, Ashish Vaswani et al. ensured that the network captures the time dependency by adding alternating sine and cosine functions to the input of both the encoder and the decoder. The period of the sine and cosine function increases from 2π to $10000pi$ as the dimension increases, allowing the network to study the position of a time step across the whole sequence, as shown in equations 2 and 3 [16].

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{2}$$

$$PE_{(pos,2i+i)} = cos(pos/10000^{2i/d_{model}}) \tag{3}$$

4.3.2 Capturing Periodicity with One_hot Encoding

The Transformer model can keep track of long-range context dependency but grapples with the periodicity of the input sequence. For many cases in crowd forecasting, crowd mobility has explicit patterns in regard to the time of day, the day of the week, and whether its a weekday or a weekend. More variables, including class schedule and weather information, could be passed in as a state or condition that is present periodically. Thus, we add a layer of one_hot encoding on top of the embedding layer to extract this periodicity.

5 Experiments and Results

In this section, we conduct experiments to test our model and evaluate its performance by comparing to LSTM models.

5.1 Data Sets

We evaluate our model based on four real-world data sets, including two self-collected sets on the campus of Phillips Exeter Academy and two public sets collected in Belgium in the research of Utkarsh Singh et al., as described in Table 1. With maps available in Appendix I, Data Set 1 and 2 corresponds to sensor locations S1 and S4 respectively in Figure 16; Data Set 3 corresponds to sensor locations S14 to S18 in Figure 17; and Data Set 4 corresponds to sensor locations S1 to S7 in Figure 18.

Table 1: Data sets utilized in experiment

Data Set	Ref	Location	Setting
Data Set 1	Self-collected	Downer Family Fitness Center, Phillips Exeter Academy	Campus
Data Set 2	Self-collected	Passage, Phillips Exeter Academy	Campus
Data Set 3	[13]	La Bourse de Bruxelles, Bruxelles, Belgium	Urban
Data Set 4	[13]	Sainte-Catherine, Bruxelles, Belgium	Urban

It is worth noting the scarcity of publicly available data sets in Wi-Fi crowd monitoring, as Data Sets 3 and 4 collected in the research of Utkarsh Singh et al. are the only sets available to the best of our knowledge.

Splitting our data into training and testing sets of different proportionality based on test scenarios, we first review Data Sets 1 and 2 in a campus setting and then Data Sets 3 and 4 in a city setting. For the evaluation of Data Sets 3 to 4, we replicate the input sequence length and horizon of [13] to ensure comparability, though our model is capable of predicting with much longer input sequences and horizons larger than that of the experiments of Utkarsh Singh et al.

5.2 Experimental Details

5.2.1 Training and Hyperparameters

During training with ADAM as our optimizer, teacher-forcing is utilized so that the ground truth, rather than the generated output from the last time step, is passed as input to the decoder to improve training speed. We decided against using dropout layers to prevent over-fitting, as it may hinder the performance of a model on relatively small data sets. Instead, we employ regularization to prevent over-fitting.

Through extensive testing, we finalized our model with 4 attention heads and 6 attention layers. Look-ahead masking is implemented in the decoder self-attention block to prevent the decoder fitting to future time steps. That is, the attention relevance scores are set to zero for future time steps, which restricts the decoder from reading future information to make predictions.

To prevent our model from making predictions solely on a the time basis encoded as one_hot dimensions, we apply to all attention layers L1 regularization, expressed as

$$L_1 = (wx + b - y)^2 + \lambda |w| \tag{4}$$

where λ is 0.0001, and L2 regularization,

$$L_2 = (wx + b - y)^2 + \lambda w^2 \tag{5}$$

where λ is 0.001, adding additional penalty to the loss function.

5.2.2 Metrics

To evaluate the performance of our crowd forecasting model, we select MSE as our loss function during traning, defined as

$$MSE = \sum_{i=0}^{H-1} \frac{(\hat{y}_i - y_i)^2}{H} \tag{6}$$

where $\hat{y}$ is the ground truth and y_i is the predicted crowd size, and evaluate our model performance using RMSE:

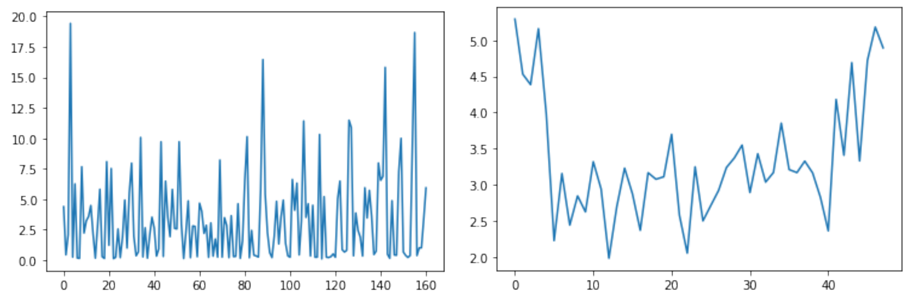
$$RMSE = \sqrt{\sum_{i=0}^{H-1} \frac{(\hat{y}_i - y_i)^2}{H}} \tag{7}$$

5.3 Forecasting with Self-Collected Data Sets

We put our model to test using data collected from the sensors. We choose 4 hours as both the lookback length and the forecast horizon from a practical standpoint.

5.4 Forecasting with Public Data Sets

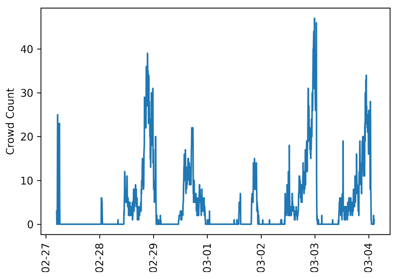
For Data Set 3 and 4, we train the model on data of the first-three days, and test the model on five days' worth of data. For forecast-horizon specific comparisons, please refer to Appendix C and D.



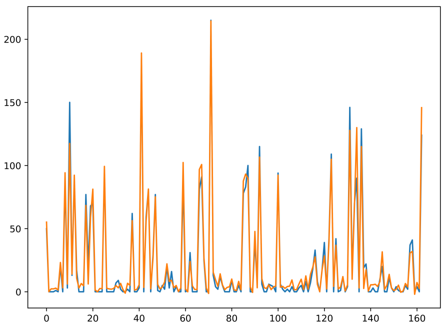
(a) MSE (y-axis) by Testing Sample Index (x-axis)

(b) MSE (y-axis) by Forecast Horizon (x-axis, each step is a 5-min interval)

Figure 9: Data Set 1: MSE

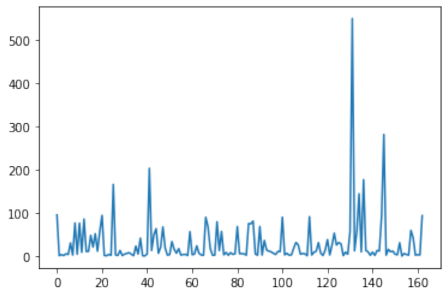


(a) Crowd Count

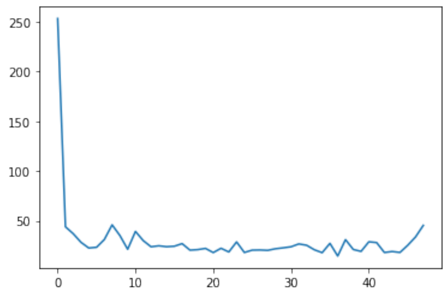


(b) Ground Truth (Blue) vs Prediction)

Figure 10: Data Set 1: Ground Truth and Prediction

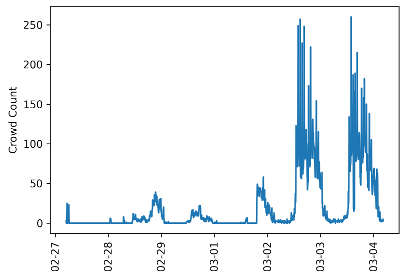


(a) MSE (y-axis) by Testing Sample Index (x-axis)

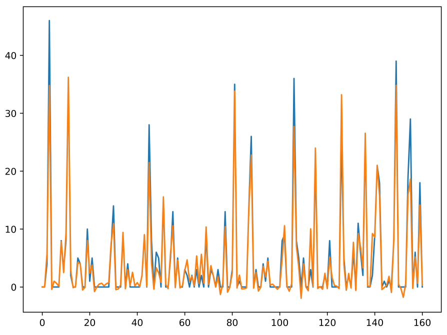


(b) MSE (y-axis) by Forecast Horizon (x-axis, each step is a 5-min interval)

Figure 11: Data Set 2: MSE

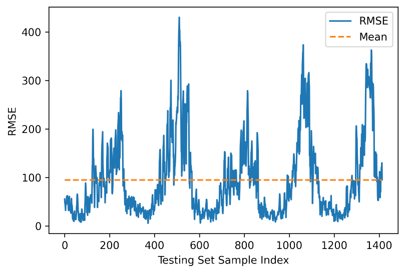


(a) Crowd Count

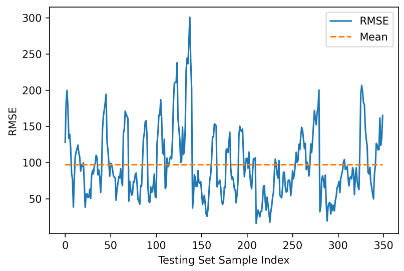


(b) Ground Truth (Blue) vs Prediction)

Figure 12: Data Set 2: Ground Truth and Prediction

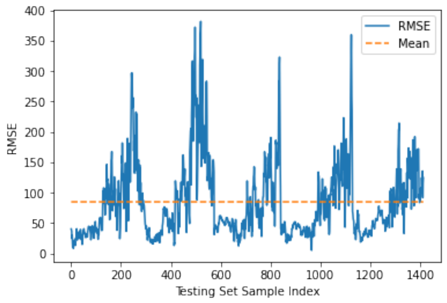


(a) Overall RMSE by Testing Sample Index

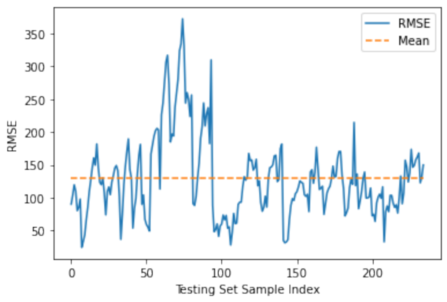


(b) RMSE during Peak Traffic (10:30 AM to 4:30 PM)

Figure 13: Data Set 3



(a) Overall RMSE by Testing Sample Index



(b) RMSE during Peak Traffic (4 PM to 8 PM)

Figure 14: Data Set 4

5.5 Results

In this section, we examine the results from our own data sets, and then compare our model to existing models proposed in the recent research, Crowd forecasting based on WiFi sensors and LSTM neural networks [13].

Training on data from sensors located on campus for the case of Data Set 1 and 2, Figure 10b and 12b represent the model's prediction of a 5-minute horizon and ground truth on testing sets that are shuffled. The results demonstrate our model's ability to capture mobility patterns over a long time span.

5.5.1 Comparasion with Existing Models

Utkarsh Singh et al. covered five different LSTM variants, including the Original LSTM, Bidirectional LSTM, Encoder-Decoder LSTM, Convolutional Neural Netowrk LSTM, and Convolutional LSTM, denoted as LSTM, BiLSTM, EDLSTM, CNNLSTN, ConvLSTM respectively in Table 2, and the authors included

Model	Data Set 3	Data Set 4	Percentage Improvement
RW [13]	229.40	274.46	-
LSTM [13]	158.49	214.30	26.01
BiLSTM [13]	146.59	223.17	26.61
EDLSTM [13]	149.64	197.44	31.12
CNNLSTM [13]	150.06	206.39	29.26
ConvLSTM [13]	142.68	190.18	33.94
Our Model	96.93	130.24	**54.91**

Table 2: RMSE Comparison with Existing Models in [13]

a RW (random walk) model as a baseline [13]. We referenced their data in Table 2 to compare with our results.

Prior research generally confirm that LSTM grapples with long-range context dependencies, and the we are unaware of the performance over longer input sequences in the work of Utkarsh Singh et al., as the input of the LSTM variants are crowd counts in the past 60 minutes grouped into 5-minute intervals, and the forecast horizon is set to 30 minutes, a relatively short period of time for some practical use cases.

To ensure comparability, we applied the exact same training data, lookback period, and forecast horizon. For Data Set 3 our model is trained on three days of data – Dec 25, Dec 26, and Dec 27 – and tested on a set of five days; similarly, for Data Set 4, our model is trained on data from Dec 01, Dec 10, and Dec 13, and evaluated on a testing set of five days as well.

As shown in Table 2, our Transformer-based model yields the best result across the board, observing the exact same training and testing setup. We significantly lowered RMSE for these two data sets, and leads the next best-performing model, namely ConvLSTM, by a 20.97% margin compared to baseline. These results further demonstrate that our model is versatile going from a campus setting with the maximum amount of traffic being around 100 to busy urban centers with thousands of people.

6 Conclusion

In this paper, we developed a full-fledged crowd tracking sensor network and offered a Transformer-based approach to generating upcoming crowd mobility forecasts. We encoded periodicity into the model using one_hot to express the periodic nature of our data. Tested on self-collected and public data sets, our model successfully navigates in different contexts, including college campuses and busy urban centers, for it captures long-range context dependencies in historic data, increasing the forecast horizon and accuracy by a substantial margin. Comparing our model to other models in a 2020 study [13], we lowered RMSE by 54.91% compared to the baseline model, and leading ConvLSTM by a wide margin of 20.97%.

With our mobile application shown in Appendix A, the sensor network is now in phased deployment at Phillips Exeter Academy.

6.1 Future Prospects

In our future efforts, we will explore the integration of Transformer with Graph Convolutional Network that captures the crowd mobility across different locations, which we were not able to explore due to limitations of available data sets.

List of Figures

References

[1] Lokesh Boominathan, Srinivas S S Kruthiventi, and R. Venkatesh Babu. “CrowdNet: A Deep Convolutional Network for Dense Crowd Counting”. In: *Proceedings of the 24th ACM international conference on Multimedia*. 2016, pp. 640–644.

[2] *Buy a Raspberry Pi 3 Model A+ – Raspberry Pi*. `https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/`. (Accessed on 09/07/2020).

[3] Simone Di Domenico et al. “Trained-once device-free crowd counting and occupancy estimation using WiFi: A Doppler spectrum based approach”. In: *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 2016, pp. 1–8.

[4] Julien Freudiger. “How Talkative is Your Mobile Device? An Experimental Study of Wi-Fi Probe Requests”. In: *Proceedings of the 8th ACM Conference on Security Privacy in Wireless and Mobile Networks*. WiSec ’15. New York, New York: Association for Computing Machinery, 2015. ISBN: 9781450336239. DOI: `10.1145/2766498.2766517`. URL: `https://doi.org/10.1145/2766498.2766517`.

[5] Matthew S. Gast. *802.11 wireless networks the definitive guide ;* OReilly, 2013.

[6] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`. eprint: `https://doi.org/10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[7] *Kali Linux — Penetration Testing and Ethical Hacking Linux Distribution*. `https://www.kali.org/`. (Accessed on 08/21/2020).

[8] *Laravel - The PHP Framework For Web Artisans*. `https://laravel.com/`. (Accessed on 09/08/2020).

[9] Mikel Rodriguez et al. “Density-aware person detection and tracking in crowds”. In: *2011 International Conference on Computer Vision*. 2011, pp. 2423–2430.

[10] J. Scheuner et al. “Probr - A Generic and Passive WiFi Tracking System”. In: *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. 2016, pp. 495–502.

[11] Vishwanath A. Sindagi and Vishal M. Patel. “A survey of recent advances in CNN-based single image crowd counting and density estimation”. In: *Pattern Recognition Letters* 107 (2017), pp. 3–16.

[12] Vishwanath A. Sindagi and Vishal M. Patel. “CNN-Based cascaded multi-task learning of high-level prior and density estimation for crowd counting”. In: *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. 2017, pp. 1–6.

[13] Utkarsh Singh et al. “Crowd forecasting based on WiFi sensors and LSTM neural networks”. In: *IEEE Transactions on Instrumentation and Measurement* 69.9 (2020), pp. 6121–6131.

[14] Utkarsh Singh et al. “Crowd Monitoring: State-of-the-Art and Future Directions”. In: *Iete Technical Review* (2020), pp. 1–17.

[15] *Understanding LSTM Networks.* URL: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[16] Ashish Vaswani et al. “Attention is All You Need”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 2017, pp. 5998–6008.

[17] E. Vattapparamban et al. “Indoor occupancy tracking in smart buildings using passive sniffing of probe requests”. In: *2016 IEEE International Conference on Communications Workshops (ICC).* 2016, pp. 38–44.

[18] Wei Wang et al. “Occupancy prediction through Markov based feedback recurrent neural network (M-FRNN) algorithm with WiFi probe technology”. In: *Building and Environment* 138 (2018), pp. 160–170.

[19] Wikipedia. *Monitor mode — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Monitor\%20mode&oldid=868693168`. [Online; accessed 07-September-2020]. 2019.

[20] Wei Xi et al. “Electronic frog eye: Counting crowd using WiFi”. In: *33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014.* 2014, pp. 361–369.

[21] Cong Zhang et al. “Cross-scene crowd counting via deep convolutional neural networks”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2015, pp. 833–841.

[22] Han Zou et al. “FreeCount: Device-Free Crowd Counting with Commodity WiFi”. In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference.* 2017, pp. 1–6.

A GUI and Mobile App

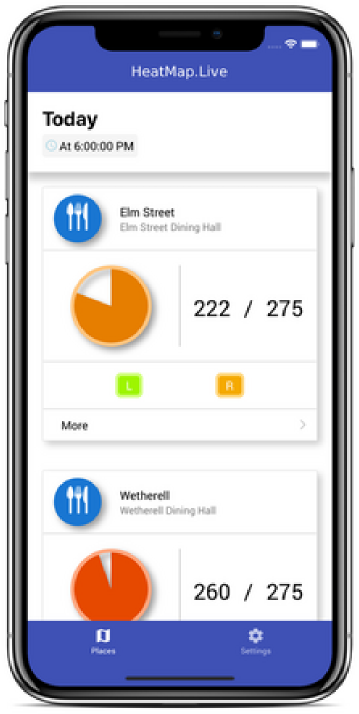


Figure 15: Mobile Application

B Sensor Locations

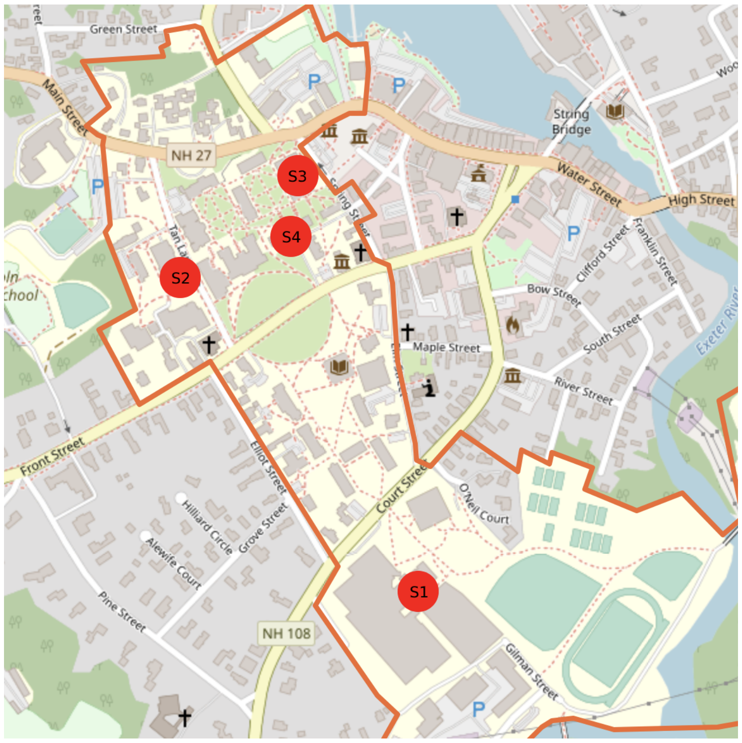


Figure 16: Sensor Locations of Data Set 1 and 2

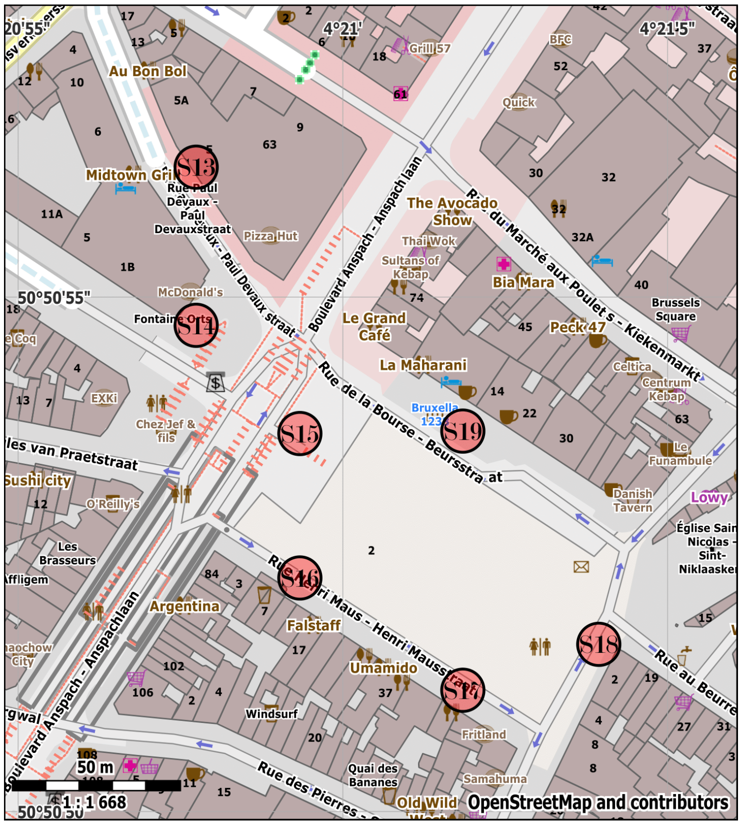


Figure 17: Sensor locations of Data Set 3 [5]

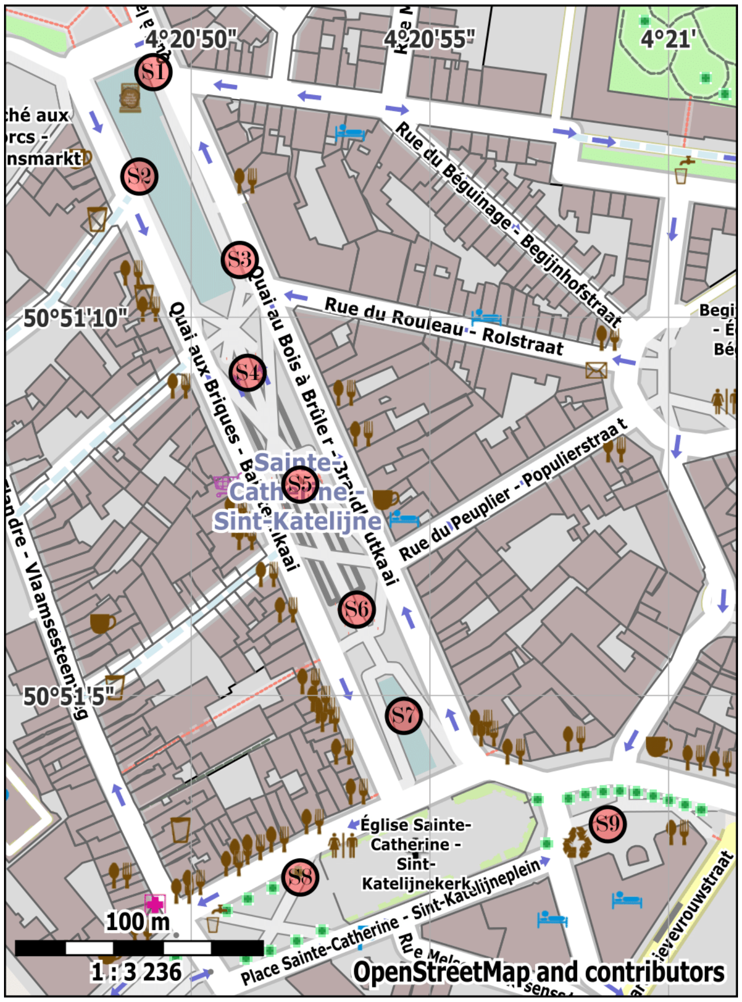
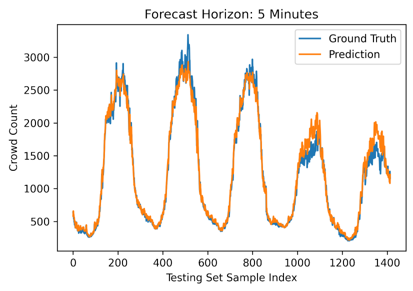
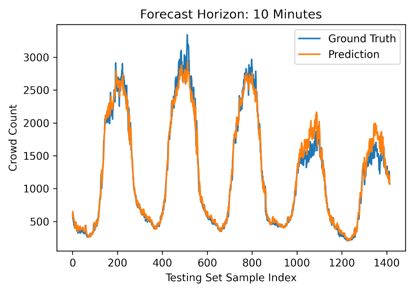
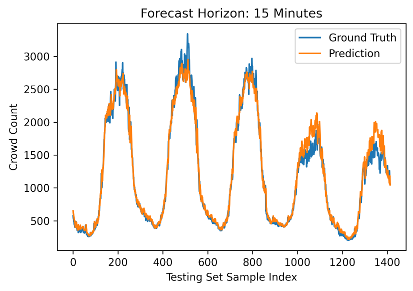

4°20'50"
4°20'55"
4°21'
S1
S2
S3
S4
S5
S6
S7
S8
S9
50°51'10"
50°51'5"
Quai aux Briques - Baksteenkaai
Quai au Bois à Brûler - Brandhoutkaai
Rue du Béguinage - Begijnhofstraat
Rue du Rouleau - Rolstraat
Rue du Peuplier - Populierstraat
Sainte-Catherine - Sint-Katelijne
Église Sainte-Catherine - Sint-Katelijnekerk
Place Sainte-Catherine - Sint-Katelijneplein
100 m
1 : 3 236
OpenStreetMap and contributors

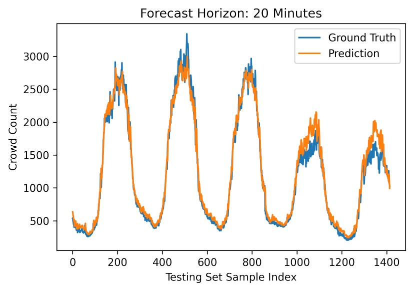
Figure 18: Sensor locations of Data Set 4 [5]

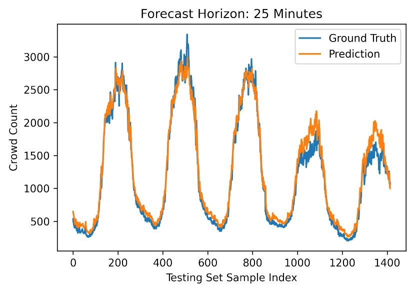
C Data Set 3 Results

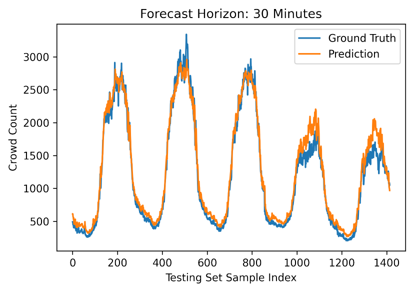


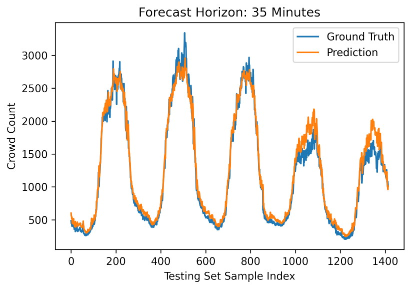


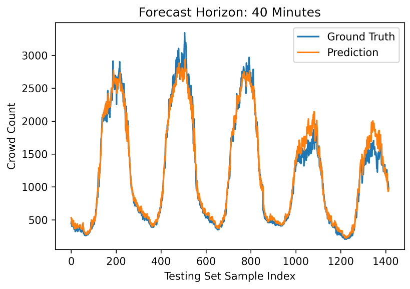


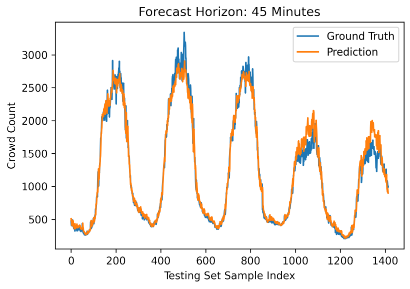


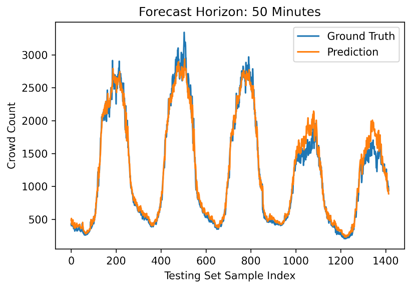


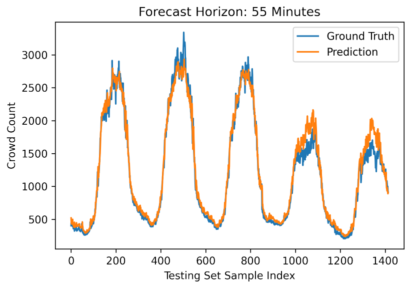


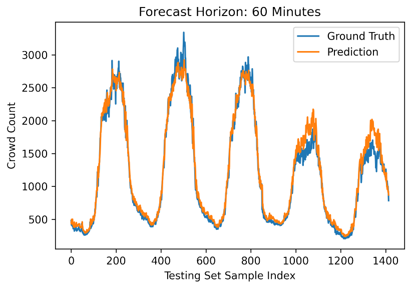




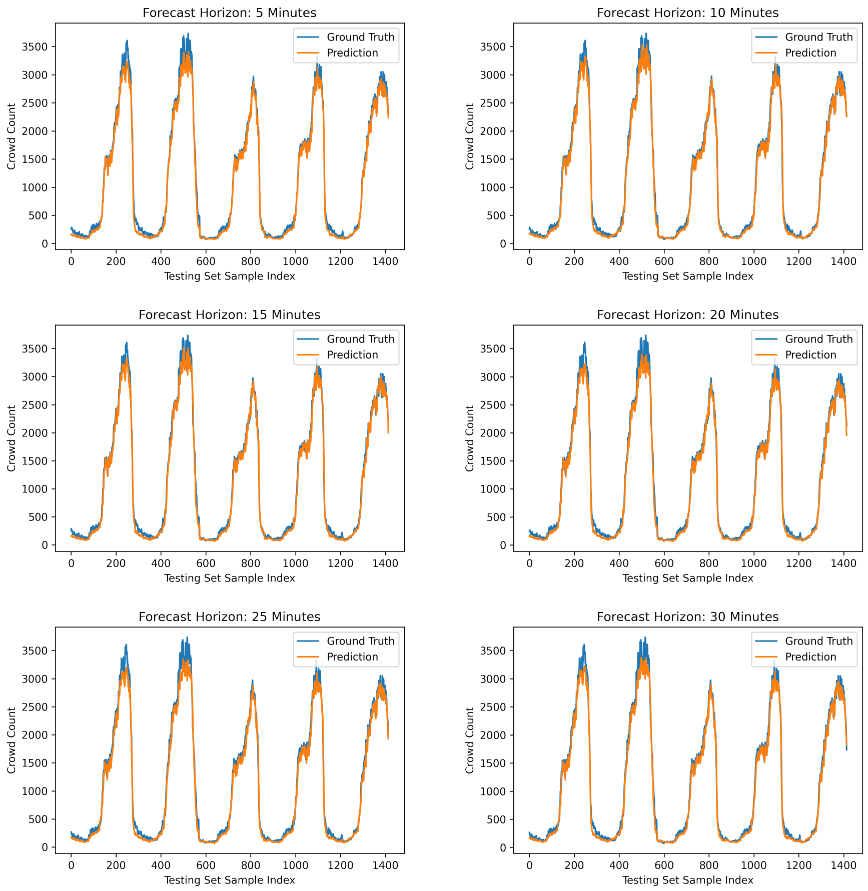


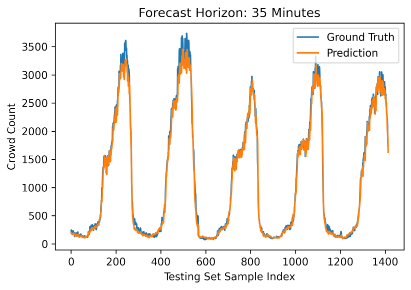


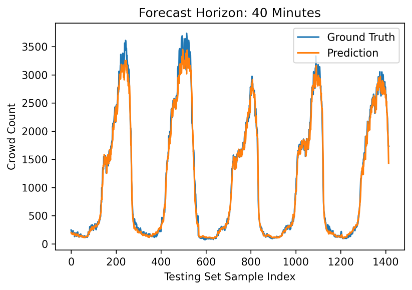


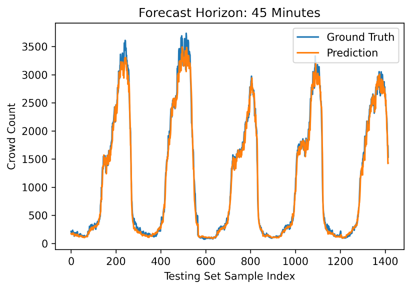


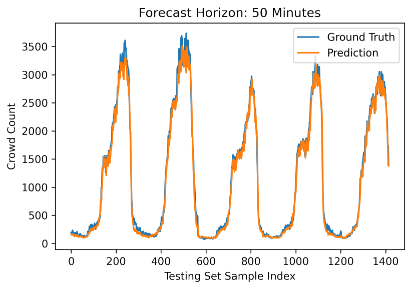
D Data Set 4 Results

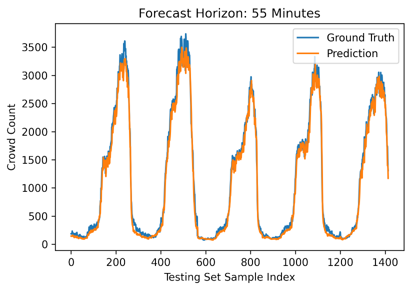

Forecast Horizon: 5 Minutes
Forecast Horizon: 10 Minutes
Forecast Horizon: 15 Minutes
Forecast Horizon: 20 Minutes
Forecast Horizon: 25 Minutes
Forecast Horizon: 30 Minutes
Ground Truth
Prediction
Crowd Count
Testing Set Sample Index
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400

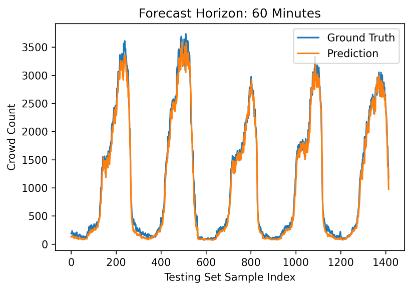

Forecast Horizon: 35 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index


Forecast Horizon: 40 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index


Forecast Horizon: 45 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index


Forecast Horizon: 50 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index


Forecast Horizon: 55 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index


Forecast Horizon: 60 Minutes
Ground Truth
Prediction
Crowd Count
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Testing Set Sample Index