Predicting the Unpredictable: Comparing Statistical Forecasting and Deep Learning Models for Forecasting Emergency Department Arrivals

Citation

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37367691

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:_dash.current.terms-of-use#LAA

Share Your Story
The Harvard community has made this article openly available. Please share how this access benefits you. Submit a story.

Accessibility
Predicting the Unpredictable: Comparing Statistical Forecasting and Deep Learning Models for Forecasting Emergency Department Arrivals

Michael Zelenetz

A Thesis in the Field of Software Engineering for the Degree of Master of Liberal Arts in Extension Studies

Harvard University
November 2020
Abstract

The hospital emergency department must be continuously staffed and ready to care for patients. Staffing and demand could be more easily matched if hospitals were able to better predict surges in emergency department visits. In this thesis, we examine methods for the development of forecasting models including feature engineering and model selection. Deep learning-based forecasting methods are compared to traditional statistical methods to predict emergency room arrivals using emergency room time series data over 48 months period. Specifically, we examine RNNs and LSTM architectures and compare them to ARIMA models. Each of the two approaches has tradeoffs in terms of complexity, flexibility and the ability to model external events. We discuss these tradeoffs as they pertain to predicting demand and how they affect the operationalization of a model in an emergency room setting.
Michael Zelenetz is a paramedic turned data scientist. From a young age he was interested in how data and models can be used to better understand the physical world and to make predictions about the future.

Michael studied Sociology as an undergraduate because of its focus on using data and model to describe and understand social behavior. He was particularly interested in how people react in disasters and how anomalous events impact behavior. He wrote his undergraduate thesis on looting and criminality in the wake of disasters.

Michael started his professional career working in New York City’s emergency medical system as a paramedic, drawn again, perhaps, by his interest in emergencies and anomalies.

Seeing a place for applications of data science, analytics, and machine learning in healthcare, Michael pursued a course of study at Harvard University in software engineering where he focused heavily on data science. Michael left EMS and began working full time in data science at New York Presbyterian. Much of his work there, again, focused on the emergency department, patient flow, and how external factors impact hospital operations. His work at New York Presbyterian was the inspiration for this thesis.
Dedication

To my wife Leah who has been a constant source of love, support and inspiration and who has always pushed me to achieve my goals.

To my children Alex, Noah, and Ella, the best children a parent could hope for.

My parents, Debora and Andrew Zelenetz, who have believed in me from day one and without whom none of this would have been possible.
Acknowledgments

Foremost I would like to express my sincere gratitude and thanks to my thesis advisor, Professor Hongming Wang, who pushed me and encouraged me when I felt I could not continue. Professor Wang, worked with me to hone my project and saw it through many iterations and project ideas. She has been a great source of information and sources. I am very grateful for having the opportunity to have worked with her. I would also like to thank Professor Sylvain Jaume, my research advisor, for his invaluable guidance and advice throughout.
Table of Contents

Author’s Biographical Sketch ........................................................................................................ iv
Dedication ........................................................................................................................................ v
Acknowledgments ........................................................................................................................ vi
List of Figures .................................................................................................................................. viii
Chapter I. On Predicting the Unpredictable .................................................................................. 1
Chapter II. Statistical Forecasting .................................................................................................. 2
  2.1. What is a Forecasting Problem? ..................................................................................... 2
  2.2. Trend Decomposition ................................................................................................ 3
  2.3. Exponential Smoothing ............................................................................................... 8
  2.4. Regression Models .................................................................................................... 9
Chapter III. Deep Forecast Models ............................................................................................. 12
  3.1. Deep Learning Foundations ...................................................................................... 12
  3.2. Recurrent Neural Networks ..................................................................................... 14
  3.3. Long Short Term Memory ............................................................................................ 17
Chapter IV. Predicting Emergency Department Arrivals ............................................................. 19
  4.1. Background ............................................................................................................. 19
  4.2. Methods .................................................................................................................. 22
Chapter V. Results and Discussion ............................................................................................. 28
Chapter VI. Conclusion ............................................................................................................... 48
Bibliography ................................................................................................................................. 50
List of Figures

Figure 1. Trend Decomposition ................................................................. 4

Figure 2. Best and Worst Residuals ........................................................... 7

Figure 6. RNN architectures ................................................................. 15

Figure 7. Simple representation of weights in an unrolled RNN .............. 16

Figure 9. Haifa Flu Trends ................................................................. 24

Figure 10. New York City search trends for depression ....................... 25

Figure 12. SARIMAX 1 hour forecast ...................................................... 29

Figure 13. SARIMAX 3 hour forecast ...................................................... 30

Figure 14. SARIMAX 6 hour forecast ...................................................... 30

Figure 15. SARIMAX 12 hour forecast .................................................... 31

Figure 16. SARIMAX 24 hour forecast .................................................... 31

Figure 17. SARIMAX 48 hour forecast .................................................... 32

Figure 18. Naïve Model ................................................................. 34

Figure 19. Feed Forward Network ......................................................... 35

Figure 20. Convolutional Network ......................................................... 37

Figure 21. Comparing Mean Absolute Error of different models .......... 38

Figure 23. NYC Daily Arrival Predictions .............................................. 40

Figure 24. Naïve Model ................................................................. 41

Figure 25. Single Hidden Layer Feed Forward Network ...................... 42

Figure 26. Convolutional Neural Network Forecast ............................ 43

Figure 27. Autoregressive LSTM on daily arrivals ............................. 44
Figure 28. Mean Absolute Errors of daily ED Arrivals using a number of models
Chapter I.

On Predicting the Unpredictable

The motivation for this project was straightforward: hospital emergency departments (ED) must be adequately staffed 24/7—however, appropriately staffing the ED is non-trivial. Having adequate nurse staffing in the ED is important for patient safety, operational efficiency, and patient satisfaction. (Recio-Saucedo et al., 2015)

If a hospital were able to better predict patient arrivals to the Emergency Department before they happened, they would be able to ensure adequate staffing ahead of anticipated surges.

There are predictable patterns in emergency department arrivals. These patterns may vary from one hospital to the next depending on the area it serves. Some emergency departments are slower on weekends whereas others are generally busier on weekends. Special events may alter calendric patterns in emergency department volume. An ideal model would be able to account for features like holidays, special events, extreme weather events, flu season, or other communal infectious outbreaks and would forecast out a number of hours or days into the future to give operational leaders sufficient time to prepare for an expected surge.

Traditional forecasting models using statistical methods are good at finding trend within the time series but do not directly account for external forces. Deep learning architectures are gaining traction in forecasting and may prove to be a useful tool in the forecaster’s toolbox.
Chapter II.
Statistical Forecasting

2.1. What is a Forecasting Problem?

In order to apply forecasting techniques to a problem, one must first ensure one is dealing with a time series problem. A time series analysis is an abstraction whereby we assume discrete events when summarized over a period of time are continuous. For example, we can consider the traffic over a bridge as a time series—each car driving over the bridge is a discrete event but if we sum the number of cars per minute or per hour we can model the expected traffic over a given time period.

To continue the example of traffic over a bridge: we can construct a naïve model whereby we assume that however many cars crossed over the bridge in the last hour will cross over the bridge in the next hour. It is obvious to anyone who has had to leave home early in order to beat rush hour that this model will fail.

In bridge traffic, as with other time series problems, there are different trends that increase or decrease the expected volume. For example, there are daily traffic patterns across the bridge—it’s busier during morning and evening rush hour—and weekly patterns—busier during a work week than during the weekend. There may also be calendric patterns—there may be less traffic during the summer than during the school year, for example.
The basic components of a time series forecast are trend, seasonality and cycles. Trend is the underlying growth rate over time. As the population of a city increases, so too the overall traffic over the bridge will increase. Seasonal trends are the calendric trends (daily, monthly, yearly, etc.). Cyclic trends are less common—these are larger term fluctuations like economic conditions, housing markets, etc. (Hyndman and Athanasopoulos, 2008)

2.2. Trend Decomposition

When analyzing a time series and/or trying to fit a forecast it is helpful to remove cyclical patterns from the time series. Removing regularly recurring patterns such as hourly, weekly, or monthly cycles can help expose the underlying trend. Alternatively, when trying to create a forecast, stripping away cyclical trends can help in finding additional features to improve the prediction. In other words, one could ask once the cyclic effects have been removed if there is additional signal or if there only noise left.

To put this into the context of the emergency department: once we have adjusted the number of Monday arrivals to remove the expected Monday surge we can then ask if this Monday is somehow different than other Mondays.

Trend decomposition is used to separate the calendric or cyclical trends in a time series from the other signals. A time series can be represented as

\[ y_t = S_t + T_t + \epsilon_t \]
For time, \( t = 1 \) to \( N \). Where \( y \) is the value of the time series at time, \( t \). \( S \) is the seasonal component at \( t \). \( T \) is the trend at \( t \), and \( \varepsilon_t \) is the residual error term.

A sample trend decomposition diagram as generated by the Prophet package from an urban New York City Emergency Department is shown in figure 1. The first figure
shows the underlying trend over time. This is the baseline expected daily arrivals over time. The second row shows the adjustment for US legal holidays to the underlying trend—all of the holidays have negative impacts meaning holidays have a damping effect on daily emergency department visits overall. Next we see the impact of weekday on the expected daily arrivals. On Sunday once can expect 20 fewer visits from the value in the first row of the figure whereas on Monday one can expect 20 more visits. Finally the bottom figure shows the trend over the course of the year. January to March tends to see an increase in volume with a dip in the spring followed by increase in volume over the summer.

STL was described by Cleveland et al. as an algorithm for seasonal-trend decomposition using locally weighted regression (loess). The goals of STL were to be easy to use first and foremost. The algorithm should allow for flexibility in specifying the amount of variation in the seasonal and trend components, and in the number of observations per cycle in the seasonal component. The algorithm should gracefully handle missing values and extreme values in the data (Cleveland et al., 1990).

Loess weights more recent values—or, formally, values closer to the target value—higher by taking “neighborhood” of values and weighting the neighbors based on proximity to the time step of interest. Cleveland then fits a polynomial curve over the weighted data to get an ordinary least squares regression line. Finally, STL computes a coefficient to be multiplied to the weighting to ensure robustness over messy data. This procedure is run in a series of loops to update the trend and seasonal components using loess smoothing while the outer loop updates the robustness coefficient.
Time series decomposition is a critical step in understanding both seasonal impact to a time series as well as change in underlying trend. In building a forecast model it is important to take both seasonality and trend into account. Once one has removed seasonality and trend from the original data, examining the residuals can be helpful to find additional signal to model external to seasonal effects. For example, in looking at emergency department data, after removing the trends I looked at the most extreme errors and looked to see where the model failed (see figure 2). I looked at the dates and searched for news or events that may have contributed to the number of ED arrivals being off. I found that the days were the model overestimated the number of arrivals were often legal holidays such as Thanksgiving. Interestingly, the day after Thanksgiving (Black Friday) had fewer ED arrivals, like a legal holiday. The other major depressive effect on ED arrivals was *predicted* blizzards—not the actual snow totals. Conversely, looking at days that had many more arrivals than expected, I noticed many days that were the first business day after a holiday (for example, cyber-Monday). These findings are discussed later in more detail.
Figure 2. Best and Worst Residuals

Snapshot of the most extreme residuals from a model using NY Emergency Department Data
2.3. Exponential Smoothing

Exponential weighted averages were first described in 1957 by the Office of Naval Research (Holt, 2004) and has a few nice properties:

- Older data is weighted less than more recent data.
- Simple and efficient computation
- Requires little data

In order to compute the exponential weighted average one iterates over the array of values calculating the weighted average of the last step and the value at the current step. This has the advantage of reacting quickly to regime changes where there is a sudden increase in the value of the data in the series as opposed to taking an average over the whole series where a sudden change will be attenuated by the other values in the series.

```python
def moving_avg(vec):
    last_step = None
    new_vec = []
    for value in vec:
        if last_step is None:
            last_step = value
            new_vec.append(value)
            continue
        else:
            res = last_step + value / 2
            new_vec.append(res)
    return new_vec[-1]
```

```python
a= [1,2,4,1,4,2,8,10,15]
moving_avg(a)
Out[17]: 8.5
```

```python
statistics.mean(a)
Out[18]: 5.222222222222222
```
From this basic implementation of the moving average, we see the moving average returns a higher value than the mean of the array because the values at the end of the array are higher.

2.4. Regression Models

Until now we have focused mainly on time series forecasts where we model the trends within the time series without regard for the impact of external regressors. We have not explicitly considered the impact of events like weather, holidays, or special events, not to mention other time series like flu cases.

Linear models take the form:

\[ y_t = \beta_1 d_{t1} + \beta_1 w_{t1} + \beta_2 w_{t2} + \cdots + \beta_k d_{tk} + \beta_k w_{tk} + \epsilon_t \]

Where \( y \) is the target value at time, \( t \). \( \beta_1, \ldots, \beta_k \) are the coefficients of the variables \( w, d \) at time steps \( t \) where \( t = 1, \ldots, K \). \( \epsilon \) is the error term at time \( t \). When evaluating the results of a linear model we look at the residual error terms to ensure that the residuals have a mean of zero and are normally distributed—else there is systematic bias in the model and perhaps there is still useful signal in the error term. We must also look out for autocorrelation as that is also a sign to look for additional signal. (Hyndman et al., 2019)

In order to train the model, split into training and test, treat each time step as an independent observation and train using the lagging time steps as predictors. One may use lagging time steps of another time series, as well. (Hanck et al., 2020)
To prepare a time series for a regression model, one may pivot the time series from a long format to a wide format where each feature, \( X_i \), represents a time step.

Not all the time steps are predictive. For example, in the hospital data, in predicting the number of arrivals on the next shift (12-24 hours in advance), the most important time steps where the last shift, the shift before that, then the shift 7 days before the target shift, 14 days before and 21 days. Removing the other time lags did not impact the model’s performance.

The Granger Causality Test tests the lags and identify those lagged features, \( X_i \), where are predictive for \( y_t \). (Granger, 1969) (Hanck et al., 2020)

Lagged models can be further combined with exogenous variables.
Figure 4. Including exogenous variables to a linear regression model with time lag

Transforming time steps into features allows the analyst to more easily include other features that may be useful to the model. For example, days of the week, months of year. One hot encoding categorical variables is a common method for representing nominal categorical variables.

Figure 4 shows as an example how a categorical variable, day of week, is encoded at each time step. Encoding days of week as dummy variables or one-hot encoding as opposed to an categorical variable is useful when there is not intrinsic ordering. Slow, medium, fast would be an example of an intrinsic ordering or an ordinal categorical variable, whereas day of the week is a nominal categorical variable with no intrinsic hierarchy. When encoding days of the week notice the example in figure 4 excludes Sunday. Because linear regression assumes there is no multicollinearity between features, we drop one of the encoded variables. (Gordon, 1968)
3.1. Deep Learning Foundations

Frank Rosenblatt, inspired by the human brain, first described a perceptron model in 1957. He wrote,

…it should be feasible to construct and electronic or electromechanical system which will learn to recognize similarities or identities between patterns of optical, electrical, or tonal information, in a manner which may be closely analogous to the perceptual processes of a biological brain. The proposed system depends on probabilistic rather than deterministic principles for its operation, and it gains its reliability from the properties of statistical measurements obtained from large populations of elements. (Rosenblatt, 1957)

Rosenblatt’s description proved to be ahead of its time. The neural network, as it became known, fell out of fashion as there was not a good solution for how to build a multilayer network.

Rumelhart et al. described the backpropagation algorithm which allowed for building multilayer artificial neural networks in 1986. Backpropagation adjusts the weights of connection between neurons “to minimize … the difference between the actual output vector of the net and the desired output vector.” (Rumelhart, 1986)

For a layer $j$ the input of $x$ is a function of the output of the incoming units $y$ with weights, $w$, on those connections.

$$x_j = \sum_{i} y_i w_{ij}$$

In addition to the weights connecting one neuron to the neuron in the next layer, there may also be connection weight between a bias and the neuron. The goal is to limit the total error. When passing an input vector through the network the goal is to minimize
the distance between the output vector from the network and the known value. The total error, E is defined by Rumelhart et al.:

\[ E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2 \]

Where c is the index of cases (input-output pairings), j is the index over the input features, y is the output of the network and d is the expected output. Rumelhart uses gradient descent to minimize E by taking the partial derivative of E with respect to each weight, w. The weights in the network are then adjusted.

Figure 5. Simple Neural Network

*Feed forward neural network takes features x₁...m and passes each to a neuron which is connected to neurons on subsequent layers. The connections are weighted. Each neuron also has a bias input, a₀.*

Backpropagation laid the groundwork for multilayer neural networks. Figure 5 illustrates a simple architecture of an artificial neural network. The inputs, X, are fed into the network in the first layer. The layers are fully connected, meaning there are weights
between a neuron in one layer to any neuron in the next layer. There are bias weights represented in this image by $a_0$.

The network in figure 5 is an example of a feed forward neural network. (Raschka et al., 2019) A feed forward network has no looping—inputs pass from one layer to subsequent layer but never to prior layers. The final layer is called the output layer, the first layer is called the input layer. The layers between input and output are known as hidden layers because we don’t see the output of these layers. (Goodfellow et al., 2016)

3.2. Recurrent Neural Networks

Until now we have limited our discussion to feedforward networks. While feedforward networks work well for tabular data, they are not optimal for sequential data. Sequences include data like speech or sound, sensor data, biometric waveforms like EKG and EEGs, text generation, and character level language models, to name a few examples. The key feature of sequential data is that order matters—the state at time $t$ will impact the state at time $t+1$. (Raschka et al., 2019)

Generally in supervised machine learning as presented until this point, there is an underlying assumption that the data is independent and identically distributed (Raschka et al., 2019), in other words for a inputs $X_1, \ldots, X_n$ the ordering of the inputs does not matter and the inputs are completely independent of one another. However, in sequential problems the ordering is key. For example, if one wanted to predict the next word in a sentence one would need to feed the sentence to the model in a certain order. We can come up with a clear prediction for the most likely masked word when we see the sequence in the correct order: “Better to have loved and ____” as opposed to the wrong
order “loved to have and better ____”. (There are non-sequence based approaches that would probably perform well on this task like using a bag of words, however this should demonstrate the intuition to the reader.) Time series data is a type of sequential data, but not all sequential data is time series. Text, for example, is sequential but is not a time series.

Forecasting Emergency department arrivals is a good example of a sequence problem: using the arrival patterns over the past $n$ days, predict the future $n+3$ days. RNNs come in many flavors including, one to many, many to many (synchronized and unsynchronized), many to one.

Figure 6. RNN architectures

*There are a number of RNN architectures, which makes RNN useful for a number of use cases. Those architectures include: many to one, where the input is a sequence and the output is a single value. Many-to-many synchronized: where the prediction is happens at the same time step as the input. Many-to-many unsynchronized: where the model predicts a future sequence. One-to-many: a model takes a single value as an input and predicts a sequence as an output.*
Figure 7. Simple representation of weights in an unrolled RNN

The RNN can be unrolled to be represented similar to a feed forward neural network. There are weighted connections between one hidden layer and the next, or one time step and the next.

Many-to-one models are useful for taking a series and making a single prediction like a classification. As an example, taking a EKG tracing as a sequence and classifying the presenting rhythm is one example of a many to one classification problem.

Many–to-many models come in two flavors: synchronized and unsynchronized. Synchronized models are those where the output prediction is used in real time. For example, in Recurrent YOLO and LSTM-based IR single pedestrian tracking, Yun et al., describe using a sequence model to label pedestrians in a video feed. This paper extends the model described in You Only Look Once: Unified, Real-Time Object Detection from a convolutional neural network (CNN) to an RNN.
Unsynchronized many-to-many models are useful for tasks like forecasting where given an input sequence one wants to predict future sequences. Other examples include text generation where given the beginning of a sentence the model is asked to complete the sentence.

Similar to the exponential weighted moving average discussed previously, the weight and state a value at t-1 is passed to t. The unrolled RNN in Figure 7 looks very similar to a feed forward neural network.—a simple representation of an RNN is simply stacked feed forward networks where the output of one network is an input to the next network.

In figure 7 the weight $W_{xh}$ is the weight between the input at $t_i$ the weight $W_{hh}$ is the weight between two time steps, $t_{i-1}$ and $t_i$ and $W_{hy}$ is the weight between the hidden unit and the predicted output at $t_i$. Then the final output can be represented with the formula:

$$o(t) = \phi_o(W_{ho} h^{(t)} + b_0)$$

Where $o$ is the output, $\phi_o$ is the activation function, $W_{ho}$ is the weight between the hidden layer and the output, $h^{(t)}$ is the hidden layer at $t$ and $b_0$ is the bias. (Raschka et al. 2019)

3.3. Long Short Term Memory

One of the weaknesses of RNNs is that they do not perform well over long periods of time. They are prone to vanishing or exploding gradients—when doing backpropagation over a large number of recurrent weights, if the weights are greater than
one they will get very large and if the weights are small, they will get very small. One way to avoid the exploding gradient problem is to use fewer time steps, however, you will start to lose important pieces of information.

Imagine trying to carry on a conversation with someone who could only keep 20 seconds of running short-term memory—they would have a hard time carrying out a back and forth conversation. Conversation topics would pretty much be limited to commenting on the weather or asking what time it is. This was a problem with the early AI assistants like Alexa and Google Home, they did not maintain state or memory between questions. It could answer “what is the weather today?” but would not understand the question if after reporting the day’s weather it was asked “and how about tomorrow?”

Hochreiter et al. first described the long short-term memory (LSTM) in 1997. The LSTM is a memory cell used as the hidden layer of an RNN. Instead of using weights, which can cause exploding a vanishing gradient, the LSTM updates the output or state of the prior cell. The input features at time $t$ as well as the output state from the prior cell ($h_{t-1}$) are fed into the memory cell (Hochrieter, 1997).

Within the memory cell are an input gate, an output gate, and a forget gate. (Gers, et al. 2000). The forget gate resets the cells state. The input gate updates the state based on the input features $X_t$ the bias $b_t$ and the input state, $h_{t-1}$. The output gate deals with updating the values of the hidden layers. (Raschka et al., 2019)
Chapter IV.

Predicting Emergency Department Arrivals

4.1. Background

The idea of predicting patients arriving in the emergency department first came to me when I was working in a large New York City Hospital. I had been a paramedic for many years before moving into analytics and data science and maintained a close relationship with many of my emergency department colleagues.

After a particularly bad flu season which had caused record numbers of emergency department visits, hospital admissions, and ICU stays; the hospital decided to invest in preparing for surges. It would agree to dedicate additional staff, move nursing to the ED, open up CT scanners that were generally reserved for inpatients or outpatients to patients in the Emergency Department. They agreed to lift overtime caps to allow for maximal staffing, even to postpone elective procedures to make room in the hospital for a surge of patients. These interventions were extremely costly and disruptive. It was agreed that there would be scaled surge levels with level 1 being a heightened state of awareness that would have some implications for Emergency Department staffing but would not require action from the rest of the inpatient units except to advise them that there were a lot of patients awaiting beds and that they should discharge anyone who could reasonably be discharged. Surge level 2 indicated the hospital was truly in crisis and that the whole hospital had to mobilize to help move patients out of the Emergency Department.
In exploring the patterns of throughput in the Emergency Department is became evident that there were very clear patterns in the data—more than expected. As a paramedic it was widely believed that Mondays were the busiest day of the week, especially after a holiday weekend. It was also believed that extreme heat or cold drove up the number of ED visits, and that holidays were generally quiet. Now, having seen the patterns with my own eyes manifest within the data it occurred to me that there may be predictable patterns.

We believed that the number of arrivals was a good indicator of how busy the emergency department was on any given day—that when the number of arrivals were high the emergency department would be “busy.” We struggled with what it meant to be busy—was is physical demand, cognitive demand, resource demand? One thing was for sure though, the number of arrivals did not seem to crowding in the Emergency Department. There were days when there would be large spikes in the number of patients arriving but the Emergency Department census quickly normalized, the length of stay was not drastically increased. And yet, other days, a small spike in arrivals seem to cause the emergency department to spin out of control for days with extremely long stays in the emergency department and a large number of patients in the emergency department.

Why, I wondered, did the emergency department barely seem to notice a spike in arrivals and other days even the slightest bump in arrivals caused a prolonged surge condition in the ED lasting days? It occurred to me that the conditions that cause these surges may be predictable.

I spoke with the emergency department leadership and suggested that instead of simply worrying about how the current state of the emergency department compares to
historical states of the emergency department and whether that should qualify as surge level 1 or 2, perhaps they should experiment with using forecasting to help head off surges and to staff up in preparation for anticipated surges.

In staffing in the emergency department there are two primary tasks: long term staffing and short term staffing. Long term staffing is looking out 6 months to a year and planning how many staff members need to be on any given shift. For example, for a given Monday in June, how many nursing shifts should be on the schedule? In other words, how many patients are expected on that given Monday in June?

The second consideration in staffing an emergency department is short term staffing. How many nurses are needed given that it’s the weekend of a large marathon? How many additional staff should be scheduled to cover anticipated sick calls? Ideally there would be a useful forecast model for predicting these short term deviations from the anticipated number of arrivals in order to prepare for surges before the surge happens.

I suggested to the emergency department leadership that this may, in fact, be possible. They were very excited by the possibility of preparing for surge ahead of surges. We discussed how far in advance they would need to look in order for the forecast to be actionable. Depending on the leadership the general feeling was that 12-24 hours would be ideal as that would give enough time to bring in additional staff. Smaller windows, 3-6 hours, could also be useful for smaller adjustments like preparing equipment, discharging patients or rebalancing staff assignments in the emergency department.
4.2. Methods

For the purposes of this thesis I used an open source data set. However a lot of my preparatory work and inspiration was based on work that was done in New York City. The dataset used in this thesis came from the Rambam Medical Center in Haifa, Israel. The dataset was made available through a joint collaboration with Rambam Medical Center and the Technion University. This collaboration open-sourced a number of datasets about hospital operations for the sake of operations research, ED Arrivals in one of those datasets.

First I explored the dataset looking for patterns in the data, seasonality, trend, and obvious anomalies. The first approach was a naïve approach—predict the next hour’s arrivals will have the same number of arrivals as the past hour. Any model should be able to outperform that naïve model.

From there I layered on additional complexity starting with statistical models such as ARIMA models, the results of the prophet package which uses time shifted linear models.

Then I looked to see if we could improve performance by adding exogenous variables. Through my knowledge of the emergency department and through my interviews with emergency department staff and leaders, I identified a number of features that were believed to impact emergency department arrivals. I looked to see if I could identify a correlation and to create meaningful features from those inputs. One such example was weather. It is a commonly held believe in the emergency department that the weather can cause surges in the emergency department. There are a number of proposed mechanisms including: snow causing more trip and falls or injuries from
shoveling, rain causing traffic accidents or an increase in pedestrians hit by cars, heat, it’s believed, causes additional strain on frail patients with underlying comorbidities such as heart disease, diabetes, or COPD, and it exacerbates their chronic conditions. Similarly cold weather can impact vulnerable populations like the homeless and cause an increase in emergency room visits.

Inspired by a Google Research paper that made headlines in mainstream media (Ginsberg et al. 2009) I thought looking a Google search trends may be helpful in understanding both flu season and other cyclical maladies. Flu surveillance data is collected by the department of health but is delayed because it is collected from doctors and hospitals after a diagnosis has been made; therefore the flu surveillance data that is publically released lags the actual flu prevalence in the community. Looking at searches for “flu” or “flu symptoms” may be useful for identifying the start and end of flu season as well as the relative severity compared to past flu seasons.

Google provides a search trend API that can give hourly results based on location, however the trends are relative and are scaled between 0 and 100. Looking over the maximum time window keeps the current score relative to past data. Hourly data is only available after 2014.

Using the search trends API I included the relative number of searches for “flu” or “flu symptoms” in the Haifa area. Then I looked for other conditions that may also have some calendric cyclicality. The only other condition I identified that appears to have a regular cycle is depression.
It was difficult to get historical Israeli weather data. I leaned primarily on my experience with New York City data and weather to find features for weather data. More detailed findings regarding weather are discussed later, however, for the purposes of defining features and methods for comparing models, I will discuss some of my approach to weather data here.

Figure 8. New York City 5 year Flu Trend

*Flu symptom search trends from the Google Search Trends API from 2016-2020.*

Figure 9. Haifa Flu Trends

*In both figure 8 and figure 9 the COVID pandemic overshadows much of prior years. 2017-2018 was an exceptionally bad flu season in New York as is reflected in the data.*
Generally the flu seasons are more noticeable in the New York data as opposed to the Haifa data.

Figure 10. New York City search trends for depression

Figure 11. Israeli search trends for depression

The New York City search trends have a more obvious cyclicity whereby there appears to more depression around the holidays and in the winter and less in the summer. The Israeli search trends reveal a spike in early-mid 2017 when there was a conflict with Syria and many in Haifa lived under the threat of missile attacks.

At first I used temperature and precipitation as continuous variables but saw no correlation between temperature and the number of emergency department arrivals, same with precipitation. This was counter to the feelings of those I interviewed and to my own experience. Even when grouping observations into buckets I did not see a correlation. I was prepared to exclude weather. I was looking at the residuals from one of my models,
Figure 2, and noticed that some of the days that where far off the expected numbers of arrivals were holidays (Christmas, Christmas Eve, Thanksgiving, Black Friday). Other days, often in the winter, were not holidays but had far fewer visitors to the emergency department than expected. I searched for news from those days on Google and found that almost all of them were days where there was an *expected* blizzard or extreme weather condition. In a number of cases the blizzard never actually came to fruition and there was no precipitation or accumulation, however, it seems, just the fact that the news warned an impending blizzard was enough to keep people from going to the hospital.

One of the other considerations where using weather data is that ideally I wanted to use only data that was available at the time that forecast would be made—it is unfair to use the actual weather at the time of the forecast if that wouldn’t necessarily be available to the model. If, for example, we will compute a forecast value for the next hour ($h_{t+1}$) at the top of every hour and it’s expected to snow within that hour, we should be careful not to leak future data in model training, for example that it will snow during the course of the current hour ($h_t$).

I created an extreme weather indicator. Using past national weather service extreme weather warnings I set the flag to true if there is an active warning and false if there is not. This approach covers the cases were the national weather service sent a blizzard warning even though snow never fell and limits the leakage in training. A model that use precipitation would not perform well in these examples because, it turns out, people do not stay home because of the inclement weather, they stay home because of *fear* of inclement weather. Furthermore, this was advantageous for this project as I was able to find Israeli extreme weather warning history.
It should be noted, however, that the climate of Haifa is more temperate than that of New York City, it does not snow and it does not get as cold so it’s possible that the weather may be less important in a place like Haifa.

As noted previously, in exploring the data, I noted there was a holiday effect so including a holiday indicator would be important. However, the holiday effect was not limited only to the holiday itself but also sometimes a day before or after the holiday. For example, Christmas Eve or Black Friday. There is also sometimes a large influx of patients the day after a holiday or long weekend, therefore creating a “first business day back” feature proved useful. When Christmas fell out over a weekend, for example, there may be a slight increase in volume the day after a holiday but there was a larger than normal daily visit volume that Monday.

After collecting the data and engineering features, I started with less complicated models and worked up to more complex deep learning models. The goal was to compare the efficacy of statistical models as compared with more complex deep learning models. Accuracy of a model is not the only consideration when building models for operations, the lifecycle of the model and the usefulness of the outcome must also be considered. How important is model interpretability, for example.
Chapter V.
Results and Discussion

5.1. Results

The baseline model is a naïve model—any model should be an improvement over the naïve model that is simply guess the next hour will be the same as the previous hour. I also looked a number of prediction intervals from 1 hour to 48 hours.

One consideration that I will discuss later is choosing the correct outcome metric, after all the hour-to-hour prediction may be less important than the number of arrivals per shift or the peak hourly arrivals.

<table>
<thead>
<tr>
<th>Prediction Interval</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1hr</td>
<td>2.8</td>
<td>2.03</td>
</tr>
<tr>
<td>3hr</td>
<td>3.11</td>
<td>2.37</td>
</tr>
<tr>
<td>6hr</td>
<td>3.59</td>
<td>2.8</td>
</tr>
<tr>
<td>12hr</td>
<td>4.1</td>
<td>3.31</td>
</tr>
<tr>
<td>24hr</td>
<td>2.98</td>
<td>2.23</td>
</tr>
<tr>
<td>48hr</td>
<td>3.02</td>
<td>2.27</td>
</tr>
</tbody>
</table>

Table 1. Naïve Model Evaluation Metrics

*Root Mean Square Error and Mean Average Error based on time shifted, carry forward, prediction.*
Looking at the output of ARIMA models. One can get a sense for where the model performs well and where it will struggle. This is an important consideration regardless of the strength or weakness of the outcome metrics.

Figure 12. SARIMAX 1 hour forecast
Figure 13. SARIMAX 3 hour forecast

Figure 14. SARIMAX 6 hour forecast
Figure 15. SARIMAX 12 hour forecast

Figure 16. SARIMAX 24 hour forecast
At first glance it is obvious that the model does not predict the extreme values well. The model does seem to capture the general seasonality and calendric trends well. The actual values also appear to lag behind the actual values slightly.

Visualizing the outputs of the models is important when considering how useful the model will be to end users? In the context of this project: do these projections help emergency department leaders prepare for surges? Do these forecasts help leaders create daily/long term staffing schedules?

SARIMA model performance showed an improvement over the naïve model.
<table>
<thead>
<tr>
<th>Prediction Interval</th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1hr</td>
<td>2.66</td>
<td>1.89</td>
</tr>
<tr>
<td>3hr</td>
<td>2.67</td>
<td>1.89</td>
</tr>
<tr>
<td>6hr</td>
<td>2.6</td>
<td>1.8</td>
</tr>
<tr>
<td>12hr</td>
<td>2.41</td>
<td>1.67</td>
</tr>
<tr>
<td>24hr</td>
<td>2.42</td>
<td>1.78</td>
</tr>
<tr>
<td>48hr</td>
<td>2.56</td>
<td>1.87</td>
</tr>
</tbody>
</table>

Table 2. SARIMA Model Evaluation Metrics

The training inputs were where first 70% of the dataset and the test data was the last 30% of the dataset. I took the log of the arrival values and trained the model on the test values. At each time step we record the prediction then append the actual value from the test set to the input of the next time step—mimicking what would happened in a true forecast model. Sample outputs of those models with different forecast horizons are visualized above. The SARIMA model only uses data from the time series to make the forecast, it does not accept exogenous variables in the model.
Another representation of the Naïve model—predicting the value at the last hour will be the value of the next hour. Three different time samples.
Deep learning model with a single dense layer using the past 32 hours as input and predicting one hour into the future.

Deep forecasting models may outperform statistical models as they learn non-linear relationships and can include exogenous variables. We see in the above figure using a single dense layer accepting 32 hours of data to predict out one time step seems to preform reasonably well. Again, the model’s maximum peak is not nearly as high as the actual maximum peak. Looking at the weights we can see which features are the most important inputs for this model.

A more useful task is to extrapolate out beyond 1 hour. This example illustrates using a feed forward neural network with two hidden dense layers and a third dense output layer. The network takes the last 48 hours as input and outputs the expected
number of arrivals in 12 hours.

![Graph showing 12 hours of arrivals predicted with actual values overlaid.](image)

Figure 19. Predicting 12 hours in advance

*Three examples of 48 hours of in arrivals in blue with the prediction (red) and actual values (green) overlaid on one another.*

Using a convolutional neural network as a multiple-to-multiple sequence model I built a model that accepted 48 hours of data, as above, and outputs all of the predictions over the next 24 hours.
Figure 20. Convolutional Network

*The model uses 48 hours of data to predict 24 hours into the future.*

While we see the final 12 hour prediction is fairly close, the forecast in this example does not follow the pattern of the actual data. The model shown above has one convolutional layer followed by a hidden dense layer and a final dense output layer. The optimal model architecture for this task is an area for future work.

Next I used an LSTM model with one LSTM layer followed by a dense output layer.
Comparing Mean Absolute Error of different models

Comparing the training and validation MAE for different deep model architectures. LSTMs performed best.

<table>
<thead>
<tr>
<th>Deep Learning</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve</td>
<td>2.04</td>
</tr>
<tr>
<td>1 Layer</td>
<td>1.86</td>
</tr>
<tr>
<td>Deep NN</td>
<td>1.87</td>
</tr>
<tr>
<td>Mult Step Dense</td>
<td>1.83</td>
</tr>
<tr>
<td>CNN</td>
<td>1.8</td>
</tr>
<tr>
<td>LSTM</td>
<td>1.71</td>
</tr>
</tbody>
</table>

Table 3. Deep Learning Model Evaluation
Another interesting architecture I tried was an Autoregressive LSTM. The major difference between autoregressive networks and other RNNs is that the result of the prior steps is simply fed to the next step as one of the model features, not through a hidden layer. It is more similar to a feed forward network where at each step of time series the prior prediction is passed through the network as in input. Autoregressive RNNs are similar to ARIMA models discussed previously. They also don’t maintain a long memory.

Below are sample predictions from the autoregressive model. It outperforms the similar task on the convolutional network.

Figure 22. Autoregressive LSTM

*Using 24 hours of data to predict the next 24 hours of data using autoregressive LSTMs*
As is evident, both the statistical models and the deep learning models perform relatively well on this task, with some exceptions. However, both have the problem that they perform relatively poorly when asked to estimate the maximum daily arrivals.

Figure 23. NYC Daily Arrival Predictions

*Output from the NYC deep learning forecast. Actual Daily Arrivals: Red, Predicted: Blue*

Figure 23 is an example of the output of a GRU network for predicting the number of daily arrivals in NYC. This illustrates well the way the model captures the general trend well but fails to predict the extreme values. This is to be expected as generally in training machine learning models we penalize extreme predictions. Models are meant to capture the underlying trends but, generally, are not designed to forecast anomalies. The task here is essentially anomaly prediction—the emergency department’s
operational leadership wants to know the chances of seeing anomalous behavior in the near future.

Another way to frame the problem that may be more useful for staffing decisions is not to predict the number of hourly arrivals but to focus on the number of arrivals per day or per shift, instead. I looked to see if the model performance would be better looking at daily arrivals that are less prone to random fluctuations. The number of hourly arrivals to the emergency department is relatively low—fewer than 10 per hour, generally, and random effects can cause noise in the trends.

Figure 24. Naïve Model
I recycled the previous models to evaluate how the same models performed when looking just at daily arrivals, instead of hourly arrivals.

First, looking at the single dense layer feed forward network. The performance of the feed forward network is similar to that of the Naïve model.

Figure 25. Single Hidden Layer Feed Forward Network

*Predicting total daily arrivals using a single hidden layer feed forward network.*
Figure 26. Convolutional Neural Network Forecast

*Predicting total daily arrival in the ED using a 1D CNN.*

The convolutional neural network had better performance in the daily arrivals as compared to hourly arrivals. However, it does not appear that the network has learned the pattern so much as the underlying pattern has become less volatile and therefore has better scoring metrics.
Figure 27. Autoregressive LSTM on daily arrivals

Using 25 days of data to predict the next 25 days of daily arrivals using an AR LSTM.

The model that performed best for the daily forecast, the autoregressive LSTM, performed poorly on daily data. Comparing the mean absolute errors of the models on daily arrivals, the convolutional network performs the best. There is much more variation in model performance than in the hourly arrival models.
Figure 28. Mean Absolute Errors of daily ED Arrivals using a number of models

Results of the models predicting ED daily arrivals
5.2. Discussion

The most important question to answer at the outset of any operation machine learning task is to understand what success looks like, how the model will be operationalized, and what information decision makers expect the model to give them that will help them make a decision.

Understanding how the model outputs will inform decision-making is crucial for choosing a model, features, and serving. There are considerations for model complexity, availability of features, compute requirements for training and inference, and the engineering around a model to get the features into the model and to get the predictions in front of decision makers.

In this project it is important to understand how the emergency department is staffed, and what decisions can be made with the information from this model and in what time frame. Originally I focused on smaller time horizons—a few hours—then, in speaking to operational leaders, learned that small time horizons did not give leaders enough time to react to the predictions. Furthermore, small surges that last only a few hours do not warrant bringing in additional staff or activating surge plans. Instead focusing on mid-range time horizons, 12-24 hours, are most useful for emergency department planning.

Statistical forecasting such as ARIMA modeling, is a good option for this sort of forecasting task, especially for long term planning where the inputs are exogenous variables are less important. It has the advantage of being relatively computationally lightweight and taking only the time series itself as input. It would not take holidays into
account or other local events like marathons, weather, or flu season. More complex statistical models and linear regression models can account for these external variables and prove to be a good option for forecasting.

Both deep learning models and statistical models penalize extreme predictions. Predicting emergency department surges is tantamount to forecasting anomalies; further research is required to develop methods to handle that class of problem. Standard methods are well suited for finding how many people will arrive at the emergency department between noon and 1 pm on a Tuesday in July, they are not well suited to determine the chance that a given Tuesday in July is likely to have far more arrivals than average.

Deep forecasting models proved to be useful for forecasting emergency department arrivals. There is likely room for improvement with more hyper parameter tuning and network architecture tuning. The input features require a considerable amount of preprocessing before being fed to the network. This introduces additional complexity into the development and deployment process.

Some model architectures work better for one class of the problem than others. For example, autoregressive RNNs worked well for hourly forecasts but performed poorly when predicting the total daily arrivals in the emergency department. This demonstrates that there is no one size fits all approach to deep learning—each use case may require its own model.
Chapter VI.

Conclusion

Time series forecasting is a common problem in many domains. Forecasting is a well-studied field, with research going back decades. The explosion of deep learning in recent years has paved the way for new developments and methods in forecasting. This thesis compared classical statistical forecasting methods to modern deep learning models.

A hospital emergency department is a particularly interesting application for forecasting. While emergencies are unpredictable, there is a predictable pattern in emergency department visits. There are factors outside of time and calendric effects that factor into emergency department visits. Better forecasting of arrivals in the hospital allows for safe cost-effective staffing, faster care, and shorter wait times.

We discussed the process of identifying features for inclusion in a forecasting model. We explored leading indicators of disease in the community such as Google searches for flu symptoms or depression. We also looked to include features such as holidays and weather that may impact traffic into the emergency department. We used feature-engineering techniques such as one hot encoding days of the week and months of year as a vector of binary values. Other feature engineering included looking for patterns in the data and model residuals that could be better expressed through more representative features. For example, using extreme weather warnings instead of values like precipitation or temperatures. Similarly, we discussed the importance of using
features in training that would be available at the time the model will be served in production; so, for example, we used forecasted weather as opposed to actual weather.

We trained a variety of models from statistical models like SARIMA models targeting a number of different forecast horizons from 3 hours to 48 hours. We also looked at deep learning models such as feed forward deep neural networks, RNNs, and LSTMs.

We found that for forecasting hourly arrivals the autoregressive LSTM performed the best. The performance of SARIMA models was fairly impressive—not much worse than that of much more complex and expensive deep learning models—solidifying them as a viable candidate for use in forecasting problems. Convolutional neural networks and multistep dense networks performed well for forecasting daily arrivals.

Deep learning models are a viable option for forecasting models and merit further study. Areas for future work may include investigating models for predicting anomalies in the forecast and for predicting extreme values.


