Circle Attention: Forecasting network traffic by learning interpretable spatial relationships from intersecting circles

Abstract. Accurately forecasting traffic in telecommunication networks is essential for operators to efficiently allocate resources, provide better services, and save energy. We propose Circle Attention, a novel spatial attention mechanism for telecom traffic forecasting, which directly models the area of effect of neighboring cell towers. Cell towers typically point in three different geographical directions, called sectors. Circle Attention models the relationships between sectors of neighboring cell towers by assigning a circle with learnable parameters to each sector, which are: the *azimuth* of the sector, the *distance* from the cell tower to the center of the circle, and the *radius* of the circle. To model the effects of neighboring time series, we compute attention weights based on the intersection of circles relative to their area. These attention weights serve as multiplicative gating parameters for the neighboring time series, allowing our model to focus on the most important time series when making predictions. The circle parameters are learned automatically through back-propagation, with the only signal available being the errors made in the traffic forecasting of each sector. To validate the effectiveness of our approach, we train a Transformer to forecast the number of attempted calls to sectors in the Copenhagen area, and show that Circle Attention outperforms the baseline methods of including either all or none of the neighboring time series. Furthermore, we perform an ablation study to investigate the importance of the three learnable parameters of the circles, and show that performance deteriorates if any of the parameters are kept fixed. Our method has practical implications for telecommunication operators, as it can provide more accurate and interpretable models for forecasting network traffic, allowing for better resource allocation and improved service provision.

Keywords: Forecasting · Telecommunications Networks · Attention.

1 Introduction

Accurately predicting mobile traffic is crucial for effective network management, resource allocation, and to optimize energy consumption. Nonetheless, it is a challenging task to achieve precise traffic predictions due to the complex spatial-temporal correlations involved. Telecommunication networks are designed to handle a massive number of users concurrently. Whenever required, additional capacity is provided by constructing more radio towers or upgrading existing ones. However, maintaining continuous power supply to these towers is inefficient

due to the fluctuating traffic load. Hence, precise prediction of future traffic can significantly reduce costs and improve energy efficiency. Telecommunications towers typically have antennas pointing in three different geographical directions, which are called *sectors*. Each sector has multiple *cells* that correspond to different frequencies and communication technologies (i.e., 2G, 3G, 4G). These cells can be divided into two groups: those belonging to the *coverage layer* and those belonging to the *capacity layer*. Cells in the coverage layer must remain powered on at all times to meet critical infrastructure requirements, such as for emergency service calls. In contrast, cells within the capacity layer can be switched off during periods of low demand, which is desirable from an energy savings perspective. However, turning off cells during periods of high demand can lead to service disruptions for customers, which is undesirable from the perspective of customer satisfaction. Accurately forecasting the traffic can allow the telecom operator to switch off cells in the capacity layer during periods of low demand. It is thus essential to be able to capture the spatial relations between the towers in order to understand beforehand how power saving on one tower would affect the neighboring ones.

Several previous attempts have been made to concurrently model the intricate spatial and temporal interdependencies between multiple radio towers. Deep learning methods have also been utilized, primarily employing Recurrent Neural Networks (RNNs) to model the temporal relationships and Convolutional Neural Networks (CNNs) or Graph Convolutional Networks (GCNs) [7] to capture the spatial relationships [13,1,2,6]. However, all of these techniques depend on constructing graphs manually, by using certain measures of correlation between the time series of various towers. We propose a method that models the spatial relationship between sectors of towers automatically and end-to-end, as part of the training process of a neural network. To the best of our knowledge, our approach is the first to learn spatial relationships in this way instead of predefining them.

Our method, called Circle Attention, first assigns a circle with learnable parameters to each sector. Then, the spatial relationship between sectors is modeled by the area of intersection between circles. The method does not depend on any specific network architecture, rather it is a general method for modeling spatial relationships. Circle Attention has several desirable properties. First, by employing a geometric design, our approach provides an interpretable way for experts to verify the plausibility of the modeled effects by plotting the learned circles on a map. Second, circular areas of effects are also somewhat plausible from a physical perspective. Consequently, it might be possible to infer *true* physical properties of the world, given a strong enough learning signal. Third, the area of intersection between two circles is relatively easy to compute, compared to other plausible geometric shapes. Each step of the computation is differentiable, which allows for end-to-end learning with back-propagation.

To evaluate the usefulness of our method, we learn to forecast the number of attempted cell phone calls to telecommunication towers in the Copenhagen area. We use a Transformer [10] as the baseline model for our experiments, and investigate whether the use of Circle Attention improves the model's forecasting

performance. However, Circle Attention is not a replacement for regular attention over time, as it works exclusively in the spatial domain, and not in the temporal domain.

Overall, the proposed method offers a customizable, interpretable, and computationally efficient way to model the spatial relationships between towers, and has the potential to infer physical properties of the world. The major contributions of this work can be summarized as follows. Firstly, the proposed Circle Attention method is a novel approach that models the spatial relationships between telecommunication towers using the area of intersecting circles. The method offers a customizable and interpretable way to model these relationships, which is important for understanding the complex spatial dependencies in cell phone traffic. Secondly, we show that Circle Attention improves accuracy on the task of forecasting cell phone traffic. Lastly, we perform ablation studies to investigate the importance of the different components of the proposed method. Understanding the importance of these components is important for potential further improvements of Circle Attention and other similar methods in the future.

The contributions of this paper provide insights into the effectiveness of modeling spatial relationships in cell phone traffic data and offer a novel approach that can be applied to other spatial data prediction tasks. The proposed method has the potential to infer physical properties of the world and can be utilized in various applications in the telecommunication industry.

The work in this paper was done using proprietary data, which cannot be made publicly available. We also cannot share code related to processing of data or other data specific details. However, we have made a public implementation of our proposed method available online[1].

2 Related Work

Network traffic prediction has been an active research topic for quite some time. Traditional approaches for modeling traffic patterns involve either statistical time series methods [9,16], or statistical learning methods [14,5]. However, these approaches either treat base stations independently or rely on manually designed features to capture spatio-temporal correlations, and therefore do not fully account for the spatial dependencies and interrelationships between base stations.

Recent advances in deep learning have opened up promising avenues for modeling spatial relations using both grid-based and tower/cell-based approaches [13,6,4,2,15]. For modeling spatial relations, both GCNs [2,6] and CNNs [13,1] have been adopted, while RNNs have been used primarily for capturing temporal connections. A popular approach has consisted in combining the "spatial" models with the "temporal" models by constructing a graph capturing spatial relationships, and then replacing matrix multiplications in RNN-based models with matrix convolutions with respect to this graph. Many studies have utilized static graphs, constructed based on topological information or overall correlations

[1] A public implementation is available at https://anonymous.4open.science/r/Circle-Attention-435A/.

between time series patterns [6,4,2,1]. However, recent research has explored the use of dynamic graphs [6,8]. GRUs and LSTMs have been widely adopted for mobile traffic forecasting, but their inability to capture long temporal relationships has led researchers to augment hourly input windows with corresponding daily and/or weekly patterns [2]. Another approach involves training hourly, daily, and weekly models in parallel, and then concatenating their outputs [15]. To address the limitations of RNNs, researchers have explored alternative approaches such as Transformer [10] architectures. Liu et al. [8] were the first to use a Transformer to forecast mobile network traffic, incorporating the spatial component to achieve superior results compared to state-of-the-art methods. The authors' approach models spatial relations using a dynamic correlation matrix between time series windows of grids, similar to graph-based techniques. In contrast, our method focuses on learning the interactions between neighboring sectors directly during model training, which allows us to capture more fine-grained spatial dependencies.

3 Method

3.1 Problem definition

In this work we focus on forecasting the Key Performance Indicator (KPI) associated with the number of attempted calls to cell towers in the Copenhagen area. A *tower* is a critical part of telecommunication networks, providing wireless communication services to millions of people. Their performance is essential for ensuring reliable service to end-users. Each tower has up to three *sectors*, which can be thought of as a grouping of antennas and cells pointing in the same direction. Different sectors are associated with different geographical areas around the towers. Figure 1 illustrates the concepts of sectors and towers.

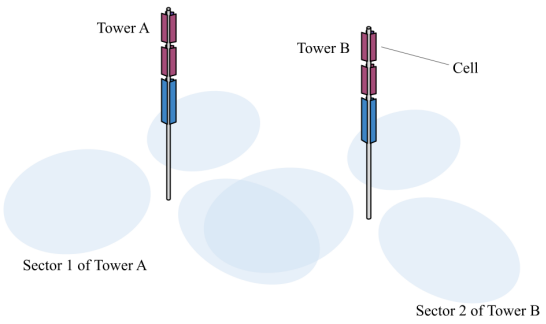

Fig. 1. Two telecommunication towers, each with three associated sectors. The cells with red coloring belong to the capacity layer, and the cells with blue coloring belong to the coverage layer.

For each sector, a number of KPIs are measured and recorded as time series data. The KPIs include measurements such as the number of attempted calls, the number of successful calls, the call duration, and data usage. The main focus of this work is to forecast the KPI that counts the number of attempted calls to a sector of a tower. Forecasting this KPI is crucial for understanding the performance of the telecommunication network, as it provides insights into the traffic demand and usage patterns for different sectors. This task is important for ensuring that the telecommunication network is operating efficiently and providing reliable service to users.

We have a total of $N_{\text{towers}} = 510$ towers in our dataset, each with its antennas assigned to one of three sectors. (In reality some towers have less than three sectors, but for ease of explanation, and without loss of generality, we will consider all towers to have three sectors.) We will use the index variable $a \in \{1, \ldots, 3 \cdot N_{\text{towers}}\}$ to refer to sector number a. Each sector a originates from a tower at a geographical location given by a coordinate pair $(x_a^{\text{TOWER}},\ y_a^{\text{TOWER}})$. Moreover, each sector a points in an azimuth direction $\alpha_a \in [0, 2\pi)$, measuring the clockwise angle from north.

The dataset contains time series that are sampled at hourly frequency. A time step t is defined as $t \in \{1, \ldots, T\}$, where T is the total number of hours in the dataset. We use a dataset consisting of 410 days, so T is $410 \cdot 24 = 9840$. We denote by z_a the time series associated with sector a. Another subscript is used to denote a time index, such that $z_{a,t}$ is the number of attempted calls within hour t for sector a.

The *neighborhood* of a sector a is chosen to be the $N_{\text{neighbors}} = 16$ sectors with the smallest Euclidean distance between their associated towers, including the sector a itself. In the next section we specify a method for modeling relationship between neighboring sectors, by associating each sector with a circle.

3.2 Circle Attention

Circle Attention models the relationship between neighboring sectors by associating a circle with each sector. The strength of the relationship between two sectors is given by the intersection of the areas of the circles, relative to the area of the circles. Figure 2 provides a graphical illustration of the formula for the intersection of two circles.

The area of intersection between two circles can be found adding two *circular segments*, which are defined by the line going through the two points of intersection of the circles. These are the purple areas of Figure 2. The area of a circular segment can be found by first finding the areas of two associated geometrical objects. First, find the area of the isosceles triangle formed by the center of the circle and the two intersection points. These triangles are shown in Figure 2 with hatched filling. Second, find the area of the associated *circular sector*, which is exactly identical to the area covered by both the circular segment and the isosceles triangle. This area is easily computed by using the inverse cosine function on the vertex angle of the triangle. Now we can define the area of the circular segment as the area of the circular sector, minus the area of the isosceles triangle.

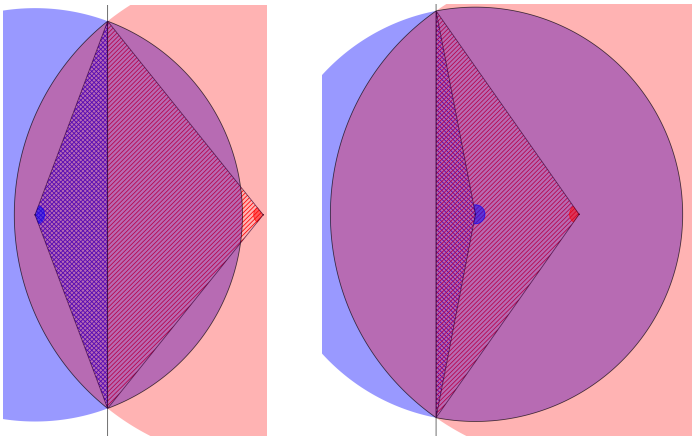

Fig. 2. Left: The intersection between two circles is given by the sum of two *circular segments*, shown here as the two purple areas to either side of the vertical line. The area of a circular segment is equal to the area of a *circular sector* (not shown), minus the area of an *isosceles triangle* (shown with hatched filling). The area of a circular sector is a proportion of the total area of the circle, which is given by the angles indicated in dark blue and dark red. **Right:** The line of intersection is "behind" the center of the blue circle, but the same method applies, except that the area of the blue triangle is instead *added* to the area of the circular sector (of the blue angle).

Note that this approach generalizes to the case where the line of intersection is "behind" the center of the circle (i.e. the right sub-figure of Figure 2), by instead adding the area of the isosceles triangle to the area of the sector.

We model the relationships between sectors by associating a circle with each sector. Each sector a has a parameter $R_a \in \mathbb{R}^+$ determining the radius of the circle, a length $L_a \in \mathbb{R}^+$ determining the distance between the coordinates of the tower and the center of the circle, and an azimuth angle α_a. As the length and radius are only allowed to take positive values, we learn the logarithm of these parameters instead of modeling the values directly. We are able to learn these parameters by back-propagation by using the exponential function, similarly to how variances are typically parametrized in variational methods. The log-values are initialized from normal distributions:

$$\log(R_a) \sim \mathcal{N}(\mu_R,\ \sigma_R^2) \tag{1}$$

$$\log(L_a) \sim \mathcal{N}(\mu_L,\ \sigma_L^2), \tag{2}$$

where $\mathcal{N}(\cdot, \cdot)$ represents the normal distribution. Consequently, the initial values of the radius and the length form a log-normal distribution. We set $\mu_R = \mu_L = 0$, and $\sigma_R^2 = \sigma_L^2 = 0.1^2$, which means that the initial circles have an expected length and radius of ca. 1 km. The azimuth α_a is also learnable, but is initialized from the value recorded in the dataset. Using the coordinates of the tower $(x_a^{\text{TOWER}},\ y_a^{\text{TOWER}})$, the length L_a, and the azimuth α_a, we can determine the coordinates of the

center of the circle for sector a:

$$x_a^{\text{CIRCLE}} = x_a^{\text{TOWER}} + L_a \sin \alpha_a \tag{3}$$
$$y_a^{\text{CIRCLE}} = y_a^{\text{TOWER}} + L_a \cos \alpha_a \tag{4}$$

We now describe the steps to find the area of intersection between two circles, each associated with a sector. Let a be the index of the first sector, and b be the index of the second sector. The first step is to compute the Euclidean distance between the centers of the circles. Additionally, a small ϵ is added to avoid numerical issues due to divisions by zero in later computations.

$$D_{a,b} = \sqrt{(x_a^{\text{CIRCLE}} - x_b^{\text{CIRCLE}})^2 + (y_a^{\text{CIRCLE}} - y_b^{\text{CIRCLE}})^2 + \epsilon} \tag{5}$$

Second, compute the triangle segment lengths V:

$$V_{a,b} = \frac{D_{a,b}^2 + R_a^2 - R_b^2}{2D_{a,b}} \tag{6}$$

To avoid numerical issues when the circles for a and b do not intersect, we have to ensure that each segment length is bounded by the radius of its circle. Next, we ensure that the absolute value of the segment length is bounded by the radius of its circle, minus a small ϵ', such that the gradients of later computations stay well-defined.

$$\widetilde{V}_{a,b} = \begin{cases} -R_a + \epsilon' & \text{if } V_{a,b} \leq -R_a + \epsilon', \\ R_a - \epsilon' & \text{if } V_{a,b} \geq R_a - \epsilon', \\ V_{a,b} & \text{otherwise} \end{cases} \tag{7}$$

Now we can safely compute circular sectors S and signed triangle areas T:

$$S_{a,b} = R_a^2 \arccos\left(\frac{\widetilde{V}_{a,b}}{R_a}\right) \tag{8}$$
$$T_{a,b} = \widetilde{V}_{a,b}\sqrt{R_a^2 - \widetilde{V}_{a,b}^2} \tag{9}$$

Finally, we compute the intersection by adding the areas of the two circular segments. The areas of the circular segments are found by subtracting the area of a signed triangle from the area of a circular sector:

$$I_{a,b} = (S_{a,b} - T_{a,b}) + (S_{b,a} - T_{b,a}) \tag{10}$$

In order to ensure numerical stability, we do an additional masking step to cover the case where circle a is completely inside circle b. In this case the intersection is equal to the area of the circle:

$$\widetilde{I}_{a,b} = \begin{cases} A_a = \pi R_a^2 & \text{if } D_{a,b} + R_a \leq R_b, \\ I_{a,b} & \text{otherwise} \end{cases} \tag{11}$$

While $\widetilde{I}_{a,b}$ should equal $I_{a,b}$ also in the case when the circles overlap due to the masking performed in Equation 7, we consistently find that using $\widetilde{I}_{a,b}$ leads to better performance. We hypothesize that this is due to numerical instabilities during training resulting from small numerical values of the distance $D_{a,b}$ between overlapping circles. Alternatively, this might be due to differences in the gradient computations between the case when the boundaries of the circles meet at a tangent point, and the case when the boundaries of the circles are overlapping but not tangent.

We can now define the *intersection score* or *Intersection over Area* (IoA) for a circle a relative to circle b:

$$\text{IoA}_{a,b} = \frac{\widetilde{I}_{a,b}}{A_a} \tag{12}$$

The intersection scores are used to create an input matrix X, with shape $T_{\text{window}} \times N_{\text{features}} = 64 \times 40$. The matrix consists of four stacked sub-matrices, corresponding to different types of input features, such that:

$$X = \left[X^{(\text{NEIGHBORS})} \mid X^{(\text{IoA})} \mid X^{(\text{LOCATION})} \mid X^{(\text{TIME})} \right] \tag{13}$$

The first type of feature is the neighboring time series, masked by intersection scores. The neighborhood of a sector a is defined by the function $\text{NEIGHBOR}(a, i)$, which returns the sector index of the ith closest sector, as measured by the Euclidean distance between the towers of the sectors. A sector a is its own first neighbor, i.e. $\text{NEIGHBOR}(a, 1) = a$. The size of the neighborhood is defined by a fixed hyper-parameter $N_{\text{neighbors}} = 16$. The circle intersection attention mechanism is implemented by multiplying the time series of the neighboring sectors by their intersection score. In other words, for a sector to be forecasted a, column number $i \in \{1, \ldots, N_{\text{neighbors}}\}$ of the matrix $X^{(\text{NEIGHBORS})}$ is the time series for sector $b = \text{NEIGHBOR}(a, i)$, multiplied by the intersection score between a and b:

$$X_i^{(\text{NEIGHBORS})} = \tilde{z}_b \cdot \text{IoA}_{a,b} \tag{14}$$

Here, $\tilde{z}_b$ is defined as the time series z_b divided by its maximum value in the training set. Second, we include the intersection scores directly, without multiplying with its associated time series. The primary motivation for this choice was to give the model a more reliable way to update the parameters of the circle, regardless of the values of the time series $\tilde{z}_b$. The intersection scores do not depend on time, so we extend the scores such that they form a constant sequence. Using the same notation as previously, this is:

$$X_i^{(\text{IoA})} = \text{IoA}_{a,b} \tag{15}$$

Third, we include a location feature based on the coordinates of the cell tower. The location feature is the coordinate pair $(x_a^{\text{TOWER}},\ y_a^{\text{TOWER}})$, with a suitable normalization applied to ensure an approx. standard normal distribution of values:

$$X^{(\text{LOCATION})} = \begin{bmatrix} (x_a^{\text{TOWER}} - m_x)/s_x \\ (y_a^{\text{TOWER}} - m_y)/s_y \end{bmatrix}^T, \tag{16}$$

where m_x, m_y, s_x, s_y are fixed scaling parameters. Because the location does not vary in time, we extend these values such that they are constant for all time steps. Fourth, we include seasonal encodings of time, which consists of 3 pairs of sine/cosine encoded seasonal features: hour of day, day of the week, and day of the year:

$$X^{(\text{TIME})} = \begin{bmatrix} \sin(2\pi \cdot \text{HourOfDay} \;/\; 24) \\ \cos(2\pi \cdot \text{HourOfDay} \;/\; 24) \\ \sin(2\pi \cdot \text{DayOfWeek} \;/\; 7) \\ \cos(2\pi \cdot \text{DayOfWeek} \;/\; 7) \\ \sin(2\pi \cdot \text{DayOfYear} \;/\; 365) \\ \cos(2\pi \cdot \text{DayOfYear} \;/\; 365) \end{bmatrix}^T \tag{17}$$

3.3 Transformer model

As our forecasting model, we use an autoregressive decoder-only Transformer [10]. More specifically, we use a recently proposed Transformer variant, called the PI-Transformer [3]. The main benefit of the PI-Transformer is the use of *Persistence Initialization*, which ensures that the initial forecasts are equal to that of a simple persistence model. This is done by adding a skip connection and a gating parameter γ. Because the PI-Transformer assumes time series with only one feature (i.e. univariate inputs), we need to modify the definition of the inputs to the first layer of the PI-Transformer. In order to transform our input matrix X to the required shape for the Transformer model, we use a weight matrix W_{in} of shape $N_{\text{features}} \times d_{\text{model}}$:

$$X_0 = XW_{\text{in}} \tag{18}$$

X_0 is then used as the input to the first Transformer layer, which produces an output X_1, and so on, until the output of the final layer; X_N. The output of the final layer is projected to a univariate series by a weight matrix W_{out} of shape $d_{\text{model}} \times 1$. This output is then multiplied by the gating parameter γ and added to the value for the previous time step, such that the (max-scaled) forecast for $z_{b,t+1}$ becomes:

$$\hat{\tilde{z}}_{b,t+1} = \tilde{z}_{b,t} + \gamma \cdot \left(X_N W_{\text{out}}\right) \tag{19}$$

4 Experimental Settings

This section describes our approach to windowing, sampling of windows, validation and test sets, optimization, and model hyperparameters. We consider telecom data from multiple cell tower antennas in the area of Copenhagen, where some towers and/or sectors may not be operational at time $t = 1$. For those sectors, we consider the time series to start at the first non-zero value. Time series with more than 50% missing or zero values are excluded from the dataset. The data is sampled on an hourly frequency, and we chose to set the forecasting horizon size to be 24 hours. To evaluate the performance of our proposed model, we split the data into training, validation, and test sets in time. The test and validation

sets each cover a period of 7 days. In other words, values of the test set are given by time indices in the range $t \in \{T_{\text{val}} + 1, \ldots, T\}$, where $T_{\text{val}} = T - 7 \cdot 24$. Similarly, the values of the validation set are given by time indices in the range $t \in \{T_{\text{train}} + 1, \ldots, T_{\text{val}}\}$, where $T_{\text{train}} = T_{\text{val}} - 7 \cdot 24$. Finally, the values of the training set are given by indices in the range $t \in \{1, \ldots, T_{\text{train}}\}$. To handle the large amount of data in our dataset, we use a windowing technique where we split the time series into windows of total size $T_{\text{window}} = 64$. For the training set, we use a stride of 1 to generate training windows, while for the validation and test set, a stride of 24 is used. The model is trained using teacher forcing [11]. The last 24 entries of each window are used as both inputs and targets to be learned with teacher forcing, while the first 40 entries serve only as inputs. For the test set, forecasts within a 24-hour window are generated auto-regressively. This means that it is necessary to forecast the values of all sectors for a single time step, before the next time step can be generated.

To ensure that the training data is representative of the overall population of sectors, we first sample sectors from the training data with uniform probability, and then sample a window within that sector's time series with conditional uniform probability. This is done in order to balance the dataset, by compensating for the fact that some sectors have fewer valid windows (because they started recording data at a later time than the initial time $t = 1$).

We use the Mean Absolute Error (MAE) in the max-scaled space as the loss function. However, we report performance metrics in the original scale. Optimization was done using the Lamb [12] optimizer, with bias correction, gradient clipping for norms greater than 10, and otherwise default parameters. A training epoch is defined to consist of 128 training batches, with each batch containing 1024 randomly sampled windows. We use an early stopping strategy to dynamically stop training if the validation loss does not improve within 8 training epochs.

Our Transformer model consists of 4 layers, each with 4 attention heads. We set $d_{\text{model}} = 64$ and $d_{\text{feedforward}} = 256$, which results in a total of around 200,000 learnable parameters within the Transformer model.

5 Results and Discussion

5.1 Experiment 1: Baseline comparison

In this section we present a comparison of Transformer models with Circle Attention (CA) relative to two baseline Transformer models, on the task of network traffic forecasting. The two baseline Transformer models are called "All neighbors", which has $X_i^{(\text{NEIGHBORS})} = 1$ for all neighbors, and "Self only", which has $X_i^{(\text{NEIGHBORS})} = 0$ for all neighbors except the sector to be forecasted (i.e. itself). In addition, we also compare against a Transformer which uses Circle Attention, but without using the intersection scores directly as features, i.e. with Equation 15 replaced by $X_i^{(\text{IoA})} = 0$. We perform 101 repeated experiments for each of the four model types. Table 1 contains the median MAE scores on the

Table 1. Median MAE test set scores of 101 repeated experiments. We also include the naïve predictor as a simple baseline to give some context regarding the magnitude of the numbers provided. S refers to the seasonality of the naïve predictor, such that $S = 24$ is the model that always predicts the value from the current hour of the previous day, and $S = 1$ is the model that always predicts the value from the previous hour. Note that Naïve with $S = 1$ has access to more information than the Transformer models, as these generate forecasts of the entire horizon auto-regressively, and therefore cannot similarly use the true previous values in their forecast.

	Naïve $S = 24$	Naïve $S = 1$	All neighbors	Self only	CA w/o feat.	CA w/ feat.
Median MAE	23.68	19.48	15.59	15.14	15.04	**14.49**

test set for each model type. We also include the scores of simple naïve predictors to give additional context. Figure 3 shows box-plots of MAE scores on the test set, and Figure 4 shows loss curves for the training and validation sets.

The models using the Circle Attention (CA) with intersection scores as features clearly outperform the three other methods on the test set. However, models using CA without the intersection scores as features only perform marginally better than the "Self only" baseline. We hypothesize that having access to the intersection scores directly as features results in more stable gradients for the circle parameters. This might be due to the amount of variation within the time series of our dataset. (The gating of the time series by the intersection scores means that the circle parameter gradients are proportional on the values of the time series in the case where the intersection scores are not included as features.)

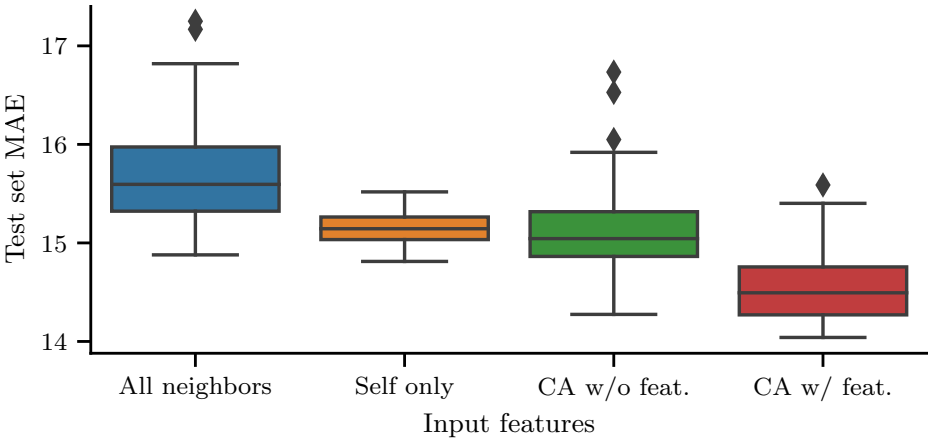

Fig. 3. Box-plot of MAE scores on the test set for the four model types of Experiment 1. Each box represents 101 repeated experiments.

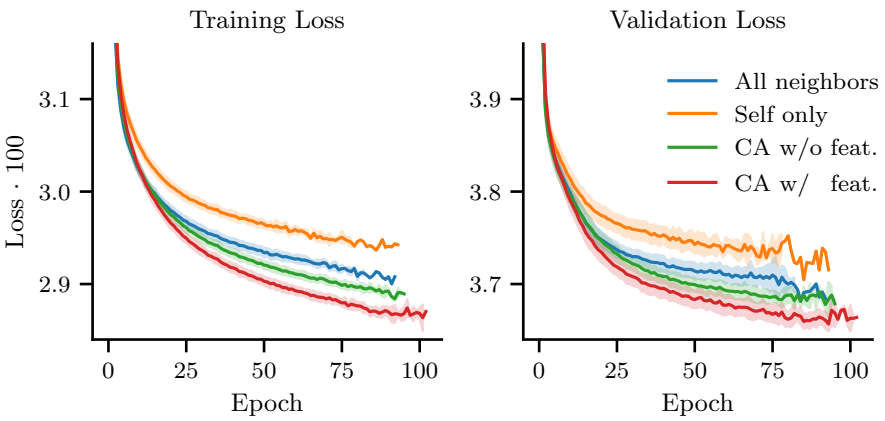

Fig. 4. Loss curves for the four model types of Experiment 1. Each curve represents the average loss of 101 repeated experiments, and the shaded regions represent the standard deviation. Please note that the plot has a survivorship bias due to the training process being stopped if the validation loss does not decrease for 8 consecutive epochs.

Interestingly, the loss curves in Figure 4 show a clear difference in both training and validation loss for the four methods. The models with Circle Attention achieve lower loss values than the two baselines, and having access to the intersection scores as features is also clearly beneficial for training. Interestingly, the "All neighbors" baseline achieves lower loss values than "Self only" baseline, while performing worse on the test set. This can likely be explained by the differences in how the forecasts are generated for the training and validation sets, versus the test set. For the validation set, teacher forcing is used to generate a full horizon of one-step predictions in one model evaluation. In contrast, the forecasts on the test set are generated auto-regressively, by iteratively feeding the previous forecasts as inputs, including forecasts of neighboring sectors. It is well known that such auto-regressive forecasts typically accumulate errors as the forecasting horizon increases. Consequently, it seems likely that the "All neighbors" baseline accumulates errors to a greater degree than the other models, which do not rely as heavily on the neighboring forecasts. This can be seen as a form of overfitting, as the model might learn multiple spurious relationships from neighboring sectors far away, which could instead be replaced with a single relationship with a nearby sector. In this context Circle Attention can be seen as a way to provide inductive bias regarding which neighbors are likely to contain useful information.

From the box-plot in Figure 3, it is clear that the "Self only" baseline has a significantly lower amount of variance in test set performance than the other models. This is further indication that an auto-regressive forecasting procedure can result in large accumulation of errors when neighboring forecasts are included as inputs to the model. However, the improved performance of the models with CA shows that having access to neighboring sectors also has the potential to improve auto-regressive forecasts, despite the additional error accumulated. In

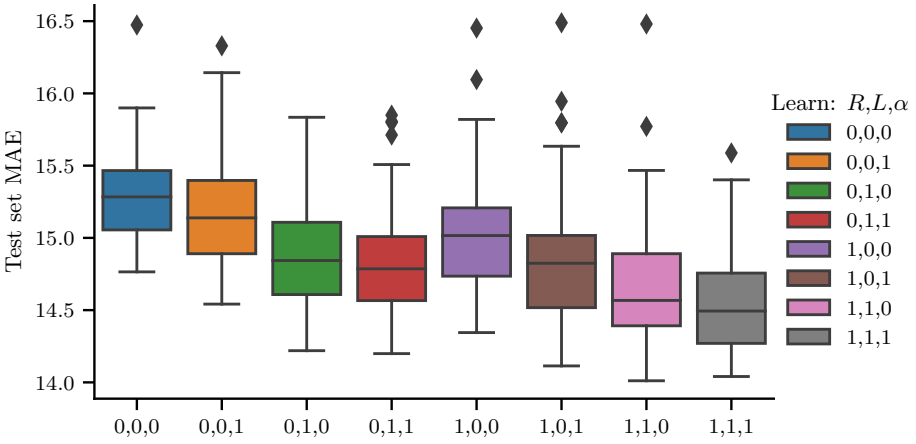

Fig. 5. Box-plot of MAE scores on the test set for the 8 ablation settings of Experiment 2. We label the boxes by a binary code, where 1 corresponds to "learnable", and 0 corresponds to "fixed". The ordering of the code is: radius, length, azimuth. Each box represents 101 repeated experiments. The gray box (i.e. where all three parameter types are learnable) is identical to the red box of Figure 3 (i.e. the box labeled "CA w/ feat.").

other words, there seems to be trade-off between access to additional information at training time, and increased accumulation of errors at test time.

5.2 Experiment 2: Ablation Study of Circle Parameters

In this section we present an ablation study on the effect of the three circle parameter types; the azimuth α, the length L, and the radius R. We perform the ablation by keeping the values of each of parameter type fixed (i.e. not learned). While we showed in the previous experiment that CA performs better than baselines which include all or none of the neighboring sectors, it is not clear that learning of all the circle parameters is required. The baselines of the previous experiment represent the extremes of possible *effective* neighborhood sizes, and models with CA fall somewhere in between these two extremes (one way to mathematically define the effective neighborhood size could be by the sum of the intersection scores of each neighborhood). Hence, it is not clear from the previous experiment that learning of the parameters is necessary, as it might be the case that simply having an effective neighborhood between the extremes would be enough. Furthermore, as the azimuth values are initialized directly from values recorded in the dataset, the model should be able to perform at a similar level regardless of whether this parameter is learned or not.

To investigate the importance of learning of the parameters, we test all the $2^3 = 8$ combinations of learned and fixed parameters, by performing 101 repeated experiments for each combination. Figure 5 shows the box-plots of the test scores of this experiment. As can be seen from the figure, learning of the parameters

appears to be critical for good forecasting performance, as learning improves performance for each of the three parameter types. However, as expected, the radius and the length are of greater importance than the azimuth, which was already initialized to the value recorded in the dataset. Interestingly, the model which has all circle parameters fixed performs better than the "All neighbors" baseline of the previous experiment, which indicates that simply reducing the effective neighborhood size is beneficial.

6 Conclusion

In this paper, we propose Circle Attention (CA), a general method to model the spatial relationships between telecommunication towers, and for improving forecasting accuracy on the task of cell phone traffic prediction. The proposed method uses the area of intersecting circles to model spatial relationships and learns the circle parameters end-to-end through back-propagation. We compare the performance of two versions of the CA method with two strong baseline Transformer models and conduct ablation studies to investigate the importance of the circle parameters. The experiments show that the models using Circle Attention outperform the other models on the test set. Additionally, the ablation studies demonstrate that learning of the circle parameters is critical for good forecasting performance, with radius and length being of greater importance than azimuth. Overall, our paper presents a novel approach to forecasting cell phone traffic, and contributes to advancing the understanding of how spatial relationships can be effectively modeled for improved forecasting accuracy in the domain of telecommunication data. We suggest several potential directions for further work. First, the concept could be extended to three dimensions by modeling the height of towers and pitch of cell antennas. Second, the idea could be extended to include variations due to temporal dynamics. Third, regularization terms could be added to constrain the circles to realistic ranges of values. Finally, a variational probabilistic approach could be used to improve interpretability and estimate uncertainty.

7 Ethical Statement

This research project focuses on forecasting telecom network traffic by analyzing one key performance indicator (KPI) aggregated per sector. Specifically, we examine the *number of attempted calls* made in each sector. To protect the privacy, security, and confidentiality of the data and the individuals whose data were used, we conducted this research project in compliance with ethical guidelines and standards.

The data we used for this project were collected at the radio tower level and do not include any personal information about the users. Instead, the data only represent the current load of the tower. We collected data in accordance with the legal and regulatory requirements of the relevant data protection agencies. We obtained informed and written consent from the telecommunications provider to use the data for this research project, ensuring that our use of the data was aligned with their terms of service and privacy policy.

Our proposed method aims to restrict the use of radio towers whenever they are underused, thereby reducing energy emissions. This method has no ethical impact apart from giving the telecommunications provider guidance on how to reduce their environmental impact.

We also emphasize the importance of ethical considerations in machine learning research. By conducting research in an ethical and responsible manner, researchers can ensure that the use of data and machine learning methods is beneficial to societal progress.

References

1. Andreoletti, D., Troia, S., Musumeci, F., Giordano, S., Maier, G., Tornatore, M.: Network traffic prediction based on diffusion convolutional recurrent neural networks. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 246–251. IEEE (2019)
2. Fang, L., Cheng, X., Wang, H., Yang, L.: Mobile demand forecasting via deep graph-sequence spatiotemporal modeling in cellular networks. IEEE Internet of Things Journal **5**(4), 3091–3101 (2018)
3. Haugsdal, E., Aune, E., Ruocco, M.: Persistence initialization: A novel adaptation of the transformer architecture for time series forecasting. arXiv preprint arXiv:2208.14236 (2022)
4. He, K., Chen, X., Wu, Q., Yu, S., Zhou, Z.: Graph attention spatial-temporal network with collaborative global-local learning for citywide mobile traffic prediction. IEEE Transactions on mobile computing **21**(4), 1244–1256 (2020)
5. Hong, W.C.: Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting. Neural Computing and Applications **21**, 583–593 (2012)
6. Kalander, M., Zhou, M., Zhang, C., Yi, H., Pan, L.: Spatio-temporal hybrid graph convolutional network for traffic forecasting in telecommunication networks. arXiv preprint arXiv:2009.09849 (2020)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=SJU4ayYgl

8. Liu, Q., Li, J., Lu, Z.: ST-Tran: Spatial-temporal transformer for cellular traffic prediction. IEEE Communications Letters **25**(10), 3325–3329 (2021)
9. Taylor, S.J., Letham, B.: Forecasting at scale. The American Statistician **72**(1), 37–45 (2018)
10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
11. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural computation **1**(2), 270–280 (1989)
12. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=Syx4wnEtvH
13. Zhang, K., Chuai, G., Gao, W., Liu, X., Maimaiti, S., Si, Z.: A new method for traffic forecasting in urban wireless communication network. EURASIP Journal on Wireless Communications and Networking **2019**, 1–12 (2019)
14. Zhang, S., Zhao, S., Yuan, M., Zeng, J., Yao, J., Lyu, M.R., King, I.: Traffic prediction based power saving in cellular networks: A machine learning method. In: Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems. pp. 1–10 (2017)
15. Zhao, N., Ye, Z., Pei, Y., Liang, Y.C., Niyato, D.: Spatial-temporal attention-convolution network for citywide cellular traffic prediction. IEEE Communications Letters **24**(11), 2532–2536 (2020)
16. Zhou, B., He, D., Sun, Z.: Traffic modeling and prediction using ARIMA/GARCH model. In: Modeling and Simulation Tools for Emerging Telecommunication Networks: Needs, Trends, Challenges and Solutions. pp. 101–121. Springer (2006)