Spatiotemporal Disease Case Prediction using Contrastive Predictive Coding

Anish Susarla*
Thomas Jefferson High School for Science and Technology
Alexandria, Virginia, USA
2023asusarla@tjhsst.edu

Austin Liu*
Los Altos High School
Los Altos, California, USA
ajzliu@protonmail.com

Duy Hoang Thai
George Mason University
Fairfax, Virginia, USA
hthai7@gmu.edu

Minh Tri Le
George Mason University
Fairfax, Virginia, USA
mle35@gmu.edu

Andreas Züfle
Emory University
Atlanta, Georgia, USA
azufle@emory.edu

ABSTRACT

Time series prediction models have played a vital role in guiding effective policymaking and response during the COVID-19 pandemic by predicting future cases and deaths at the country, state, and county levels. However, for emerging diseases, there is not sufficient historic data to fit traditional supervised prediction models. In addition, such models do not consider human mobility between regions. To mitigate the need for supervised models and to include human mobility data in the prediction, we propose Spatial Probabilistic Contrastive Predictive Coding (SP-CPC) which leverages Contrastive Predictive Coding (CPC), an unsupervised time-series representation learning approach. We augment CPC to incorporate a covariate mobility matrix into the loss function, representing the relative number of individuals traveling between each county on a given day. The proposal distribution learned by the algorithm is then sampled by the Metropolis-Hastings algorithm to give a final prediction of the number of COVID-19 cases. We find that the model applied to COVID-19 data can make accurate short-term predictions, more accurate than ARIMA and simple time-series extrapolation methods, one day into the future. However, for longer-term prediction windows of seven or more days into the future, we find that our predictions are not as competitive and require future research.

KEYWORDS

Contrastive Predictive Coding, COVID-19, Metropolis-Hastings, Mobility Data, Spatiotemporal Prediction

*These authors contributed equally to the paper.

SpatialEpi '22, November 1, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9543-4/22/11…$15.00
https://doi.org/10.1145/3557995.3566122

ACM Reference Format:
Anish Susarla, Austin Liu, Duy Hoang Thai, Minh Tri Le, and Andreas Züfle. 2022. Spatiotemporal Disease Case Prediction using Contrastive Predictive Coding. In *The 3rd ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology (SpatialEpi '22) (SpatialEpi '22), November 1, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3557995.3566122

1 INTRODUCTION

The COVID-19 pandemic has changed the way that billions of people across the globe go about their daily lives. The disease is a highly contagious respiratory illness that spreads mainly through the vectors of contact and respiratory droplets [3], and researchers have found that it has a high median R_0 of 5.8 [12].

To mitigate the spread of an emerging infectious disease, it is vital to be able to predict how, where, and when the disease is going to spread using mathematical and computational models. These models can help public health officials and partners make informed decisions about pandemic planning, resource allocation, and implementation of public health interventions like social distancing and stay-at-home measures.

One traditional model of disease spread is the susceptible-infected-recovered (SIR) model, a compartmental model that can be approximated analytically or simulated [26]. Another common approach is to treat disease as a time series and use data on the number of cases and deaths at a time point to predict future time points. However, many popular supervised time series forecasting models like the Holt-Winters method and ARIMA models require a high number of observations for the model to provide adequate predictions, a privilege that isn't afforded to researchers when fighting emerging infectious diseases like the COVID-19 pandemic in 2020 and monkeypox in 2022 [9].

Recent advancements in machine learning, however, have made it possible to create robust and effective models with limited data [8]. One such algorithm is Contrastive Predictive Coding (CPC), an unsupervised learning approach that extracts useful representations from high-dimensional data [20]. The model employs autoregressive models to predict the future in a learned latent space and optimizes its representations using a probabilistic contrastive loss.

Since emerging infectious diseases, such as COVID-19 in 2020, spread through close proximity between humans, we hypothesize

that including human mobility in the prediction should improve the prediction of future cases. Intuitively, if one spatial region (such as a country, state, or county) exhibits a spike in cases, then nearby regions are more likely to show an increase in cases in the near future.

In this study, we propose the use of CPC to predict the spread of emerging infectious diseases. Additionally, we put forth a method of improving the performance of the model using real-world foot-traffic data by embedding spatial information into the model architecture. We further propose a method of translating the learned representations from CPC to predictions probabilistically through the use of the Metropolis-Hastings algorithm. The resulting model is computationally efficient and able to more easily learn effective representations from data to predict future disease spread at the county level.

Our approach, referred to as Spatial Probabilistic Contrastive Predictive Coding (SP-CPC), preserves the ability of the model to maximize the mutual information gained from past COVID-19 data to better predict future disease spread while adding in an essential covariate feature that has a major impact on future trends. We test SP-CPC on past COVID-19 data to determine the performance of the model in predicting future COVID-19 cases.

The remainder of this study is organized as follows. Section 2 provides background on CPC and the loss function the model utilizes. Section 3 reviews the related work on CPC and the application of mobility data to predict trends in disease. Section 4 outlines the COVID-19 spread and foot-traffic data used in the SP-CPC. Section 5 describes the architecture of SP-CPC and goes through a deep dive into each of its components. Our experimental evaluation is found in Section 6 and shows that SP-CPC outperforms traditional time-series prediction approaches. We provide a link to our GitHub repository in Section 7 to allow researchers to easily use SP-CPC and reproduce our results. Finally, the concluding statements are made in Section 8.

2 BACKGROUND

This section provides a brief overview of Contrastive Predicting Coding [20] which our proposed SP-CPC approach is based upon and provides references for the interested reader to find more information.

2.1 Contrastive Predictive Coding

CPC was introduced by Oord et al. in [20] and is a self-supervised method of learning representations from complex, high-dimensional data. The approach is inspired by the theory of predictive coding from computational and cognitive neuroscience, which proposes that the brain has a generative model of the environment that outputs predictions of the future. The model's predictions are then compared against true sensory input, and the goal of the brain is to minimize the error between the prediction and the true input [15]. To mimic this behavior, CPC's goal is to extract useful representations from high-dimensional data by predicting future latent states using autoregressive models. The predictions are compared against the true future latent states and used to update the model. In their paper, Oord et al. [20] found that this approach was effective in extracting latent representations across a variety of domains, including speech recognition, text classification, and image classification.

At its simplest, the structure of CPC consists of three components: an encoder (like a multilayer perceptron (MLP) or a convolutional neural network (CNN)), an autoregressive network (such as a long short-term memory (LSTM) model or a gated recurrent unit (GRU)), and a predictor (like a single layer perceptron (SLP) or a CNN) that will output encoded form of predictions for future time steps. Each input to CPC consists of a contiguous array (such as a time series) of high-dimensional data $\{x_{t-(w-1)}, ..., x_t\}$, where w is the number of "windows" or samples of high-dimensional data in the array; each window $x_i, i \in \{t-(w-1), ..., t\}$ in the array, such as a time point (or a series of time points) in a time series or a word embedding (or a group of word embeddings) in a sentence, is then individually passed through the encoder to obtain the latent representation z_i. An autoregressive model runs continuously through $\{x_{t-(w-1)}, ..., x_t\}$ to produce a context vector c_t, which is used to predict latent representations z_{t+1}, z_{t+2}, ..., z_{t+k} several future intervals k in the series. The latent representations can then be used for predictive downstream tasks such as change-point detection [4]. We note that CPC, despite having the word "predictive" in its name, does not predict the actual future intervals x_{t+i} but only provides latent representations at future times. To close this gap and leverage CPC for prediction, we propose to use a probabilistic sampling approach based on the Metropolis-Hastings algorithm to estimate the future x_{t+i} intervals.

2.2 InfoNCE

InfoNCE, proposed with CPC in [20], is a variation of Noise Contrastive Estimation (NCE), which was introduced by Gutmann and Hyvärinen in [6]. In short, InfoNCE loss is the mathematical equivalent of a cross-entropy loss calculated from the density ratio, which is the ratio between the predictive posterior and the proposal distribution. More information on the density ratio and the mathematical explanations of InfoNCE loss can be found in Equations 3 and 4 in [20].

3 RELATED WORK

Our model utilizes mobility data in the CPC algorithm to maximize the mutual information gained in representations. This section provides an overview and related work of these techniques for solving real-world tasks.

3.1 Contrastive Predictive Coding

CPC has been applied to many different tasks across data mediums, including speech recognition in audio [23], image recognition [8], and natural language processing [1]. However, to the best of our knowledge, CPC has not yet been explored in modeling disease spread.

We specifically look to natural language in our initial research because it is similarly targeted toward predicting time points in the future (in this case, a word), but in almost all cases, CPC has been used as a feature extraction method for classification tasks. An implementation of CPC applied to natural language processing can be found in [1].

3.2 Mobility Data for Disease Spread

Mobility data has been identified as an important tool to understand and predict disease spread [16]. Mobility data has not only been used to model but also to trace COVID-19 transmission. This process is more commonly known as contact tracing, the process of identifying individuals who have recently been in contact with someone who has tested positive for COVID-19. For example, research by Kato et al. in [11] used trajectory data (mobility data that records the location of moving objects, like people, at certain moments) to build a contact tracing framework.

As mentioned prior, mobility data has also been used to model the spread of COVID-19, including in approaches without machine learning. For example, Rambhatla et al. in [22] drew on mobility data to give risk scores for each geographic region for COVID-19 spread.

Mobility data has also been confirmed to be associated with COVID-19 spread: Elarde et al. modeled changes in human mobility during COVID-19 using Principal Component Analysis and clustering and found that counties located near each other not only have similar mobility patterns, but also similar COVID-19 spread [5]. This corroborates Tobler's First Law of Geography which states that "everything is related to everything else, but near things are more related than distant things."

This association means that mobility data is critical as a covariate feature in predicting future COVID-19 spread. One such study of COVID-19 that exploited mobility data was Nikparvar et. al. in [18], which used a deep LSTM to predict future COVID-19 cases and deaths one to four weeks into the future at the county level. The study concluded that the model run with the mobility data achieved the lowest root mean square error in the prediction of new COVID-19 cases and deaths compared to other variations of the model.

4 DATASETS

This study aims to develop a mobility-based CPC algorithm to predict future COVID-19 spread at the county level. In this section, we describe the COVID-19 data and foot-traffic data provided to SP-CPC and how the data configured was configured.

4.1 COVID-19 Spread

Data on daily COVID-19 cases and deaths provided at the county level was obtained from the New York Times [19]. The data was provided in three CSV files for three years (2020, 2021, and 2022) and then combined in memory using pandas and filtered to only contain the counties of interest. Our models use a rolling 7-day average of new case data for each day, but the New York Times only contains data on total cases, so pandas was used to compute the difference between each day and the rolling average over the new cases. This data was then used as the features and targets for this study.

4.2 FIPS Codes

The New York Times COVID-19 case data aligns each day of data with its associated county with a Federal Information Processing Standards (FIPS) code. To obtain a list of all the FIPS codes in the continental United States plus Hawaii and Alaska (for this study, US territories like Puerto Rico and Guam were disregarded), a CSV file containing the FIPS codes and the state they are located in was obtained from [7]. The importance of this dataset will be highlighted in Section 4.3.

4.3 Human Mobility Data

Mobility data for this project was obtained from the University of Wisconsin @ Madison Human Mobility Flow Dataset [10]. This dataset registers the total daily visitor and population flow between each geographic region/county (and its associated FIPS code) in the United States. Each FIPS code is assigned an origin GeoID with the remaining FIPS codes assigned as destination GeoIDs. A mobility dictionary can be then built to represent total population flows between each region of the United States.

The dataset was created by analyzing anonymous mobile phone users' visit trajectories to places of interest (POIs) provided by SafeGraph. Multiscale aggregation then was used to infer population flows on three geographic scales: census tract to census tract, county to county, and state to state. In this study, SP-CPC uses the county-to-county population flow numbers.

To generate the mobility dictionary, the FIPS codes were first obtained from [7]. A CSV file from one day's worth of data (for simplicity's sake, the same CSV file was used throughout this project) from the GeoDS dataset was then read in, and the visitor and population flow for each origin/destination GeoID pair was added to an empty dictionary. However, these origin/destination pairs were only present if population or visitor flows were recorded between the locations, so the master FIPS codes list was used to fill in any missing origin/destination GeoID papers with 0. Next, the number of visitors from each origin GeoID to that destination GeoID was divided by the total number of individuals traveling to the destination GeoID to generate weights that sum to 1 for each destination. This format makes it easier to weight each time series in the SP-CPC loss function (highlighted in Section 5.2).

5 SPATIAL PROBABILISTIC CONTRASTIVE PREDICTIVE CODING

This section describes the structure of our SP-CPC architecture in detail and also describes the data structures employed in the model itself to load the case and mobility data. Figure 1 gives an overview of the proposed architecture of SP-CPC.

We first describe our approach of including human mobility information inside the traditional CPC model in Section 5.1. We denote the resulting spatial information-aware architecture as Spatially-Aware CPC. Section 5.2 describes how human mobility information is used in the loss function of SP-CPC to give the model an understanding that spatially close regions (having high mobility between them) are expected to have similar infectious disease case trends. Finally, we describe how SP-CPC converts the encoded form of predictions to raw case predictions in Section 5.3 using the Metropolis-Hastings algorithm.

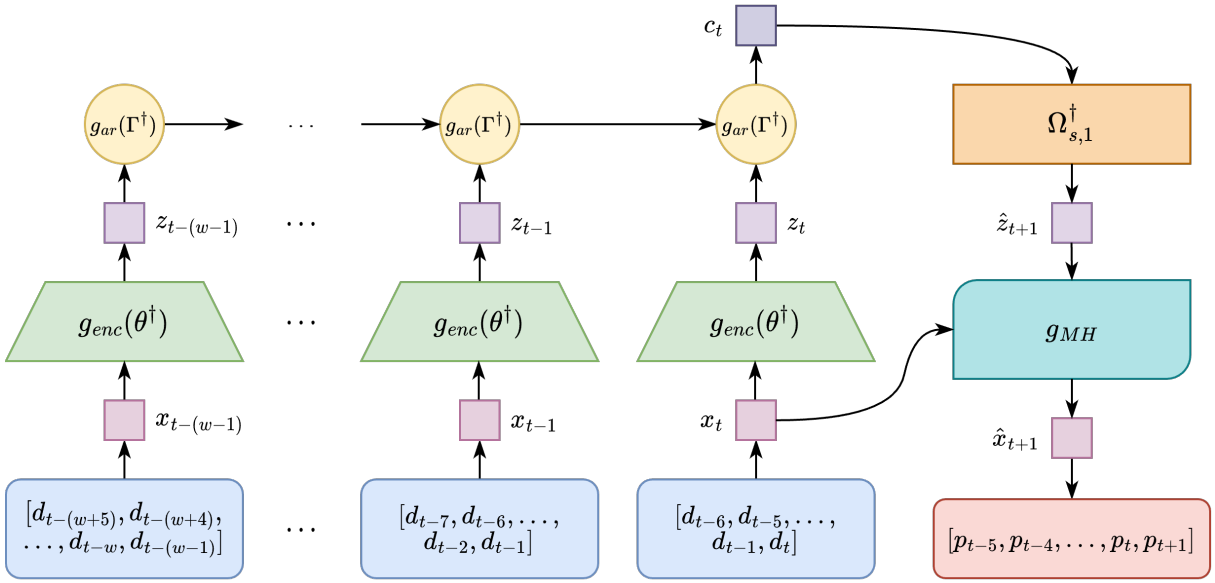


Figure 1: Architecture of Overlapping Spatial Probabilistic Contrastive Predictive Coding for Prediction

5.1 Spatial CPC Model Architecture

The architecture of SP-CPC is heavily influenced by the architecture of CPC presented in [20] and in Section 2.1.

5.1.1 Encoder. An encoder g_{enc} first converts each window of data x_t into its encoded form z_t. In our implementation, a convolutional neural network (CNN) was used as the model for the encoder. Specifically, a 1D convolutional (Conv1D) layer was initialized with a kernel size of 1, an embedding dimension of 1, and an encoding dimension of 60. A ReLU activation function was then applied to the output of the Conv1D layer. The optimal encoder parameters $\theta^\dagger$ are obtained from training the Spatial CPC model.

5.1.2 Autoregressive Layer. An autoregressive layer g_{ar} then takes in the ordered array of the encoded windows of data $z_{t-w+1}, z_{t-w+2}, \ldots, z_t$, where w is the number of input windows, and produces a context vector c_t which summarizes the encodings for the seven weeks. Our implementation uses a GRU with an input size of 60 (the output size of the encoder) and a hidden size of 60 as the autoregressive component. The optimal autoregressive layer parameters $\Gamma^\dagger$ are obtained from training the Spatial CPC model.

5.1.3 Predictive Layer. A predictive layer $\Omega_{s,j}$ is then defined for every county s passed into the model for each future time step $j = 1, 2, ..., k$ to convert the context vector summary to predictions for future encodings. Our implementation uses an SLP as the predictive layer, and the future encoding z_{t+k} is the matrix product $\Omega_{s,j} c_t$.

The $\Omega_{s,j}$s are stored in a PyTorch ModuleDict which maps county FIPS codes to the $\Omega_{s,j}$ for each county. This approach differs slightly from the original CPC paper, which could not differentiate between spatial locations and therefore ended up using the same Ω for all predictions for the same future time step k. Our approach allows us to maximize the mutual information Spatial CPC learns from the past data for each county by being able to differentiate between each county.

The optimal predictive layer parameters $\Omega^\dagger_{s,j}$ are obtained from training the Spatial CPC model.

5.1.4 InfoNCE. The final step of Spatial CPC is to calculate the InfoNCE loss between the predicted value and the target value. For each county, the predicted value is computed as the matrix product of the $\Omega_{s,j}$ for the county and the context vector c_t. The InfoNCE loss is then calculated between the predicted value and the target vector z_{t+j}. This is where the spatial information is incorporated into our model in a process described in detail in Section 5.2.

5.2 Loss Function Modification

To incorporate the mobility dataset mentioned in Section 4.3, it was necessary to modify the InfoNCE loss function and the data structures used in the method.

First, the predicted $\hat{z}_{t+1}$ from the model and the target z_{t+1} for each county were booted into their respective dictionaries. Then, the logits of each county were calculated by multiplying the weight between that origin county and each destination county with the prediction and target matrices. Finally, the cross entropy loss between the logits of the county and the label was calculated, and this entire process was repeated for each county. Then, the total loss of all the counties was calculated and returned with the necessary dimensions.

5.3 Metropolis-Hastings

The Metropolis-Hastings algorithm g_{MH} was utilized to compute the final prediction of the number of cases in each county. The true number of days into the future Metropolis-Hastings would predict would be dependent on if overlapping or non-overlapping windows

Algorithm 1 Metropolis-Hastings for predicting $\hat{x}_{t+1}$

Input: x_t, c_t and the trained parameters $\theta^\dagger$, $\Omega_c^\dagger$
Output: $\hat{x}_{t+1}$
$x_{t+1}^{(0)} \leftarrow \mathcal{N}_7(x_t, \mathrm{Id}_7)$
$\tau \leftarrow 0$
$all_{x_{t+1}} \leftarrow []$
for $\tau < 100$ **do**
  $\tilde{x} \leftarrow \mathcal{N}_7(x_{t+1}^{(\tau)}, \mathrm{Id}_7)$
  $z_{t+1} \leftarrow g_{\mathrm{enc}}(x_{t+1}^{(\tau)}; \theta^\dagger)$
  $\tilde{z} \leftarrow g_{\mathrm{enc}}(\tilde{x}; \theta^\dagger)$
  $\rho \leftarrow \exp\left((\tilde{z} - z_{t+1})^{\mathrm{T}} \Omega_s^\dagger c_t\right)$ ▷ Note that $\hat{z}_{t+1} = \Omega_s^\dagger c_t$
  if $\min(\rho, 1) > rand(0, 1)$ **then**
    $x_{t+1}^{(\tau+1)} \leftarrow \tilde{x}$
  else
    $x_{t+1}^{(\tau+1)} \leftarrow x_{t+1}^{(\tau)}$
  end if
  $all_{x_{t+1}} \leftarrow$ concatenate $\left\{all_{x_{t+1}}, x_{t+1}^{(\tau+1)}\right\}$
end for
$use_{t+1} \leftarrow$ last 50 elems of $all_{x_{t+1}}$
$\hat{x}_{t+1} \leftarrow avg(use_{t+1})$

were used; in Figure 1, only one new day of data is predicted, while in the non-overlapping model, seven days of new data are predicted.

Metropolis-Hastings is an algorithm for obtaining a series of random samples from a proposal distribution when direct sampling is not attainable [24]. The principle for using Metropolis-Hastings is derived from the probability density ratio fundamental to the functioning of CPC. After multiple direct samples with changes in the parameters of the proposal distribution over each iteration, the raw predicted case values will converge towards a single value. In the case of COVID-19 data, data is variable, so an element of randomness is necessary to obtain better predictions.

5.3.1 Parameters. Two parameters are passed into the model to configure the algorithm. The first is the context vector for each county; the autoregressive model is configured to return a context vector c_t vector for each county at the end of every sample. However, as introduced in 5.1.3, the c_t vectors are multiplied by the different $\Omega_{s,k}$ matrices for each county to form a $\hat{z}_{t+1}$ vector, representing the *encoded* form of the prediction for each county; in other words, $\hat{z}_{t+1}$ does not contain any true predicted values for the number of cases in each county. The second parameter is the x_t vector for that county, representing the raw values for the number of COVID-19 cases in that county during the current time step. Metropolis-Hastings is run one time for each county and its set of parameters c_t and x_t.

5.3.2 Proposal Distribution. The probability distribution of the model is instantiated from the MultivariateNormal class of PyTorch. The covariance matrix is set to the identity matrix, and the mean of the distribution is set to the x_t vector because future case values are dependent on previously-seen patterns.

5.3.3 Generating Predictions. SP-CPC's implementation of Metropolis-Hastings to predict future COVID-19 cases is detailed

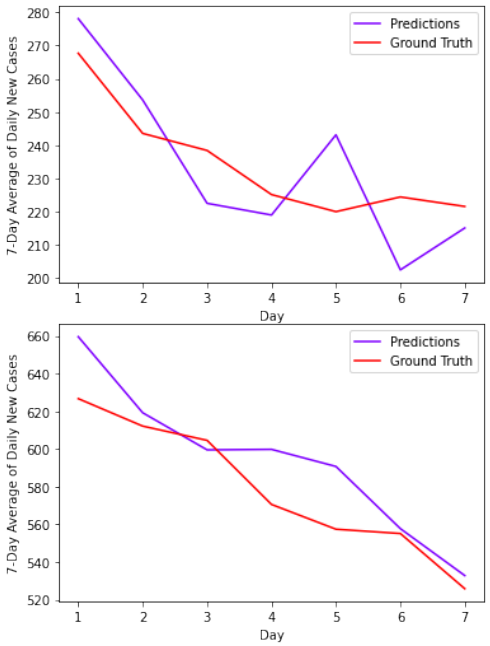


Figure 2: Example Graphs of Predictions from the first experiment (using overlapping intervals to predict the next day only)

in pseudocode in Algorithm 1. The proposal normal distribution detailed in Section 5.3.2 is referred to as $\mathcal{N}$, and the identity matrix is referred to as Id. g_{enc} represents the encoder model described in Section 5.1.1 that generates the encoding for a window, and $\theta^\dagger$ represents the trained parameters for the encoder. $\Omega_s^\dagger$ represents the trained parameters for the predictive layer described in Section 5.1.3. The prediction $\hat{x}_{t+1}$ will be compared with the true value x_{t+1} to generate the percent difference in predictions between the predictions and the ground truth, and these results are highlighted in Section 6.

6 EXPERIMENTAL EVALUATION

This section presents the results of how the SP-CPC algorithm performed in predicting future COVID-19 cases for each of the 30 most populous US counties. In Section 6.1, we describe the method for loading the COVID-19 case data, and in 6.2, we describe the parameters used for training. Next, in Section 6.4, we outline other traditional time series prediction algorithms and provide results of how SP-CPC performs against these algorithms. Finally, in Section 6.5, we evaluate how changes to the different model parameters affect the InfoNCE loss of the Spatial CPC.

Note that when calculating the final MAPE, we reject ground truth values less than 100 to avoid issues with excessively large MAPEs.

Table 1: SP-CPC results compared to traditional models (best results in bold)

Model	Train % Diff (1 day fut)	Valid % Diff (1 day fut)	Train % Diff (7 day fut)	Valid % Diff (7 day fut)
SP-CPC	**8.36%**	**7.89%**	28.11%	28.63%
ARIMA	9.78%	9.34%	**27.35%**	**27.81%**
Constant Interpolation	31.73%	33.74%	31.31%	35.55%

6.1 Data Loading

The first experiment, referred to as Experiment 1, uses $w = 7$ windows to predict $k = 1$ future time steps. The COVID-19 case data is loaded in as an array of batches, where each batch is a dictionary that maps county FIPS codes to a batch of data for a county.

Each batch of data contains a stack of 8×7 tensors, each representing eight overlapping weeks of daily case data. Each new "week" of data adds one new day of data and removes the last day, so in total, the model only contains 14 distinct days worth of data. Each week is a window in the SP-CPC model, and $w + k = 8$ windows are required. This version of the model is depicted in Figure 1. It is important to note that the next time step predicted, $\hat{x}_{t+1}$, will only be the only new element predicted by the model. The MAPE will thus only be calculated between the last value of $\hat{x}_{t+1}$ and the corresponding value of x_{t+1}.

Another version of the dataset was also prepared where the data in each week was distinct and did not overlap. This dataset used $w = 3$ windows to predict $k = 1$ future time steps; this is the equivalent of using 21 days of past data to predict the next seven days. This version is subsequently referred to as Experiment 2, and the MAPE will be computed between the elements of $\hat{x}_{t+1}$ and the elements of x_{t+1}.

The batch size determines the size of the stack sampled from each county, and in a single batch, all of the county-level stacks are collected into a single larger batch. The SP-CPC model then iterates through each county individually.

When training the model in Experiment 1, $w + k = 8$ weeks (but only 14 distinct days) of data are passed in. The first $w = 7$ weeks are used to generate the context vector and the predicted encoding for the eighth week ($k = 1$, so only 1 future time step is generated), which is then compared against the encoded version of the true eighth week reserved in the data and then used to compute loss. When evaluating the model, the 8^{th} week is passed in but is simply not used. A similar procedure for Experiment 2 occurs, except with $w = 3$ and $k = 1$.

6.2 Overall Parameters and Miscellaneous

In SP-CPC, the Spatial CPC component is the only one that requires training (as the Metropolis-Hastings portion of the model uses the learned parameters from it to generate predictions), so Spatial CPC was trained using the Adam optimizer with a learning rate of $2e^{-4}$. All training ran on a Linux environment equipped with one NVIDIA Quadro RTX 8000 GPU. As mentioned in Section 6.1, the overlapped dataset version of our code had hyperparameters $k = 1$, $w = 7$, while the nonoverlapped dataset version had $k = 1$ and $w = 3$. For both experiments, a batch size of 32 (meaning 32 samples would be pulled for each county) was utilized, and the train/validation split was 80% and 20%, respectively for both experiments.

6.3 Experimental Results

For the first experiment, on the training set, the MAPE between the predicted value of cases and the actual value of cases was 8.36%, and on the validation set, it was 7.89%. A graph of some of the example predictions can be found in Figure 2. In the second experiment, the MAPE between the predicted value of cases and the actual value of cases was 28.11% on the training set, and 28.63% on the validation set. While it is somewhat unusual to see the training error higher than that of the validation error (as in Experiment 1), it at least shows us that our model did not overfit.

6.4 Comparison to Competitor Time-Series Prediction Approaches

We compare SP-CPC against two competitor models: 1) a naive constant interpolation, and 2) an ARIMA model. Results on how SP-CPC compares to these baseline models are summarized in Table 1.

6.4.1 Constant Interpolation. Constant interpolation was used as a baseline naive method to compare against. The predictions for the next seven days were simply set to the corresponding values from the past seven days (e.g. seven days from now will be the same as today, and one day from now will be the same as six days prior). The rationale of this seven-day window is to avoid confusion about the model due to periodic patterns of reported cases, which often have a low number of reported cases on Sundays, and a large number of cases reported on Mondays to compensate. Note that since we predict the number of *new* cases on a day, this approach of using the number of new cases seven days ago does not predict zero new cases.

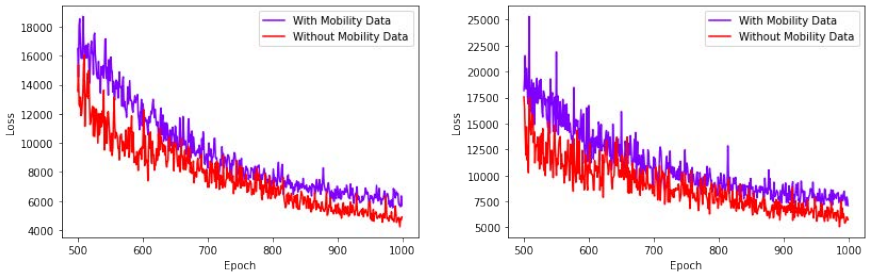
The MAPE between the predicted value of cases and the actual value of cases was calculated, with results shown in Table 1. We observe a MAPE of more than 30% for case prediction for both one and seven days into the future. For this approach, it is expected that a one-day and a seven-day prediction yield similar prediction errors, as for both cases, the value of seven days ago is used for prediction. We observe that this prediction error is much higher than that of our proposed SP-CPC model.

6.4.2 ARIMA. We also compare our proposed SP-CPC approach to an autoregressive integrated moving average (ARIMA) model. We note that this model does not take advantage of relevant covariate data like spatial human foot traffic data.

We compared SP-CPC against its two closest ARIMA equivalents: one that uses 13 days of past data at each county and returns a prediction for the next day (mirroring Experiment 1 as shown in Figure 1) and one that uses 21 days of past data and returns a future prediction for the next 7 days (mirroring Experiment 2). After using the same training and validation samples as used for the SP-CPC model, the mean absolute percent difference between the predicted

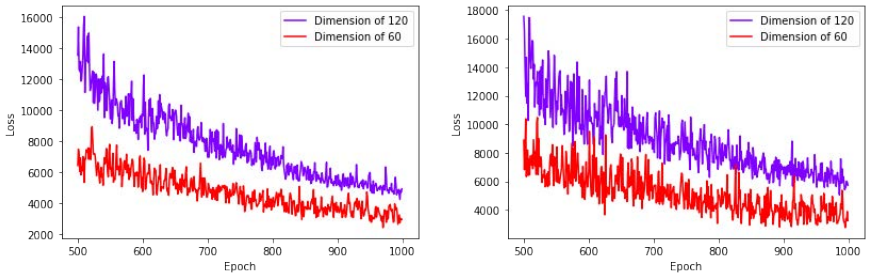
Table 2: Experiment settings (default parameters in bold)

Experiment Type	Parameter	Values
Effect of Model Parameters	Use of Mobility Data in Loss Function	No / **Yes**
	Encoder and Autoregressive Dimension	120 / **60**
	Encoder Activation Function	**ReLU** / LeakyReLU
	Encoder Model	**Conv1D** / Linear
	Autogressive Model	**GRU** / RNN



(a) Train Set Loss **(b) Validation Set Loss**

Figure 3: Effect of Using Mobility Data on InfoNCE loss from Epochs 500-1000



(a) Train Set Loss **(b) Validation Set Loss**

Figure 4: Effect of Using Different Encoder and Autoregressive dimensions on InfoNCE loss from Epochs 500-1000

value of cases and the actual value of cases was calculated, with the results summarized in Table 1. We observe a MAPE of 9.34% for the one-day prediction (Experiment 1), which is substantially higher than the MAPE of our proposed SP-CPC. This is a positive result, showing that our SP-CPC architecture can leverage spatial information to make better predictions than ARIMA models, which only use information from a single region.

However, this result is not entirely surprising, as ARIMA models have been shown to give poor predictions for seasonal time series [27]. Past trends of COVID-19 data have usually shown a peak of cases during the winter months and a comparative lull during the warmer months, giving COVID-19 the widely-accepted characterization of a seasonal disease [17]. For our future work, we will further compare SP-CPC to prediction models based on epidemiological models using both compartmental models [2, 13] and agent-based simulation [14, 21].

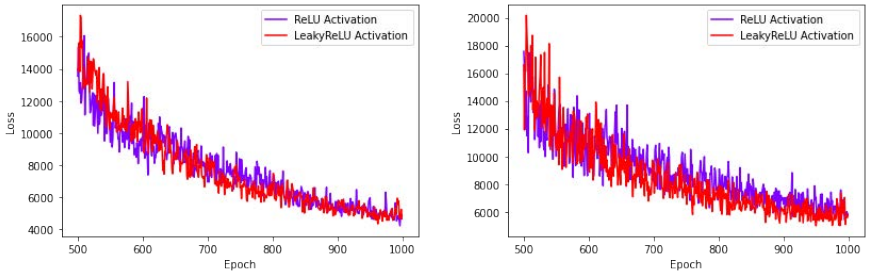
However, for the seven-day future prediction (Experiment 2), we observe that ARIMA yields better results than our proposed SP-CPC model. We contribute this to the non-overlapping approach that loses the context of the days of the week. In our conclusions in Section 8, we describe future research to improve the predictions of our model further into the future.

6.5 Effect of Model Parameters

This section presents experiments testing the impact of different parameters (described in Section 5) of our proposed SP-CPC architecture. Table 2 shows the five parameters that are evaluated in this section. Default

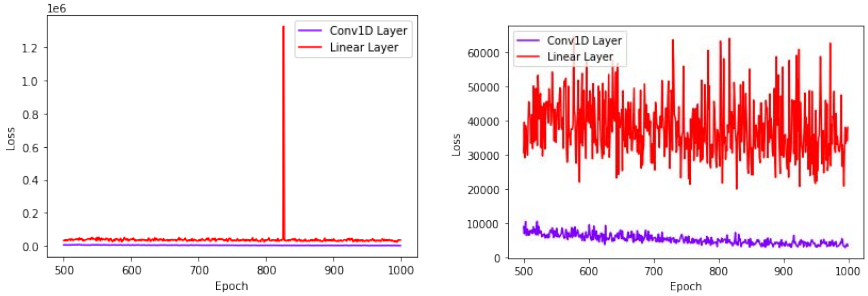
6.5.1 Use of Mobility Data in Loss Function. This experiment tested the impact of including mobility data in the InfoNCE loss function of Spatial CPC. As shown in Figure 3, both train and validation losses were consistently lower when mobility data was incorporated into InfoNCE. However, the average runtime increased from 2.83 seconds per epoch to 7.57 seconds per epoch when the mobility data was included. This can be explained by the use of a nested for loop to compute the logits for each county, increasing the algorithm complexity from $O(n)$ to $O(n^2)$, where n is the number of counties. Due to the lower loss of the model with the mobility data, however, we continued to use the mobility data in the InfoNCE loss function.

6.5.2 Encoder and Autoregressive Dimension. Different dimensions of the output channels of the encoder and the input/output sizes of the autoregressive model were also tested. Dimensions of 120 and 60 were tested, and using a dimension of 60 saw massive reductions in the loss, as shown in Figure 4. We hypothesize this dramatic



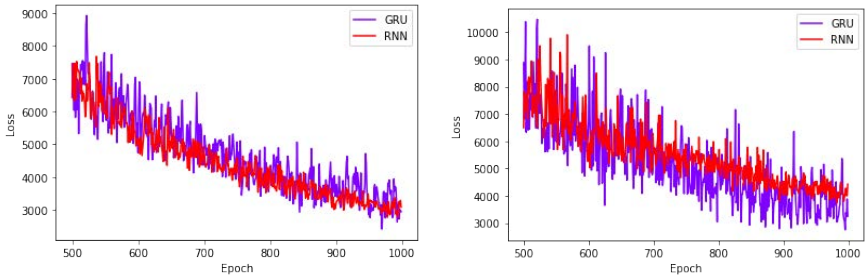
(a) Train Set Loss **(b) Validation Set Loss**

Figure 5: Effect of Using Different Activation Functions in the Encoder on InfoNCE loss from Epochs 500-1000



(a) Train Set Loss **(b) Validation Set Loss**

Figure 6: Effect of Using Different Layer in the Encoder on InfoNCE loss from Epochs 500-1000



(a) Train Set Loss **(b) Validation Set Loss**

Figure 7: Effect of Using Different Autoregressive Model on InfoNCE loss from Epochs 500-1000

reduction in loss due to the encoder choosing the most useful features from the data, rather than including features that may not have any meaning to the number of cases, and future research can include experimenting with different dimensions.

6.5.3 Encoder Activation Function. Two different activation functions after the convolution layer in the encoder were tested: the ReLU function and the Leaky ReLU function. When training with an encoder dimension of 120, there were no statistically significant differences in losses, as seen in Figure 5. However, the average runtime increased from 7.57 seconds per epoch when using the ReLU activation function to 8.35 seconds per epoch when using the LeakyReLU activation function inside the encoder, so the ReLU was used as the activation function moving forward.

6.5.4 Encoder Model. Two different encoders were also tested: the original 1D convolution layer and a single linear layer. As shown in Figure 6, using a linear layer resulted in extremely higher losses, so we continued using a 1D convolutional layer inside the encoder in future experiments.

6.5.5 Autoregressive Model. One final experiment tested the difference in model performance between a recurrent neural network (RNN) and a GRU as the autoregressive component of SP-CPC. Figure 7 shows that losses were comparable for both models, but training time did substantially increase when from 7.78 seconds to 13.02 seconds per epoch when an RNN was used. Due to the increase in training time, a GRU was chosen as the autoregressive model in our implementation.

7 CODE AVAILABILITY

All code used to train the model and format data is available publicly in the GitHub repository that hosts the code at https://github.com/ajzliu/SP-CPC.

8 CONCLUSIONS AND FUTURE WORK

This study introduces Spatial Probabilistic Contrastive Predictive Coding (SP-CPC) applied to COVID-19 case prediction. Built upon Contrastive Predictive Coding, SP-CPC utilizes a mobility dictionary that helps weight each time series against each other in the InfoNCE loss function. Once Spatial CPC returns the context vectors for each county, these vectors are multiplied by a learned parameter individualized for each county to predict an encoding for a future time window. This predicted encoding is then fed into the Metropolis-Hastings algorithm to give future case predictions for each county. Part of this study also included altering model parameters within the Spatial CPC component of the model to see which set of parameters produced the lowest InfoNCE loss, and those parameters were the ones incorporated into the Metropolis-Hastings portion of SP-CPC.

The results of our experiments show that SP-CPC can achieve better short-term prediction than baseline models like constant interpolation and ARIMA, which still are at the forefront of disease modeling [25]. These results were achieved despite SP-CPC receiving limited testing for hyperparameters and with suboptimal procedures for extended predictions on future cases. This demonstrates that SP-CPC is a promising new approach for disease modeling with limited data and spatiotemporal machine learning tasks.

For longer-term predictions, we found that SP-CPC does not outperform ARIMA models. For future work, we want to tackle this problem by investigating ways to leverage overlapping time windows for predictions more than one day into the future. We plan to approach this task by iteratively predicting the next days of COVID-19 cases based on predictions of previous days.

To summarize, there is great potential to apply SP-CPC to model the spread of other contagious diseases beyond COVID-19. An open question is whether the representation that SP-CPC learned using COVID-19 data can be successfully transferred to other emerging infectious diseases which may have different transmission pathways. Future work also includes configuring the model to use different model parameters to achieve a lower InfoNCE loss. In addition, because Metropolis-Hastings applies an approach of sampling on a probability distribution, the process of computing a confidence or model uncertainty for a given estimate becomes relatively trivial. This can be explored in future work.

Finally, there are many future directions for the exploration of improving the accuracy and performance of SP-CPC. Replacing and optimizing the proposal distributions, hyperparameters, and preprocessing of data are all areas for significant improvement of the model, and the model also does not take advantage of a key feature of the original CPC model yet. In these experiments, k has remained at 1, but increasing k may provide much better predictions for extended predictions in the future.

In summary, despite its novelty and lack of optimization, SP-CPC delivers performance that can beat models strongly used in the field of epidemiology. The algorithm shows great promise, and there is a multitude of future directions to further improve the accuracy and robustness of the model.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. DEB-2109647 for "Data-Driven Modeling to Improve Understanding of Human Behavior, Mobility, and Disease Spread." Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This research was additionally supported by the Aspiring Scientists Summer Internship Program (ASSIP) at George Mason University.

REFERENCES

[1] V. Araujo. Vgaraujov/cpc-nlp-pytorch: Implementation of contrastive predictive coding for natural language https://github.com/vgaraujov/CPC-NLP-PyTorch.
[2] F. Brauer. Compartmental models in epidemiology. In *Mathematical epidemiology*, pages 19–79. Springer, 2008.
[3] CDC. Coronavirus Disease 2019 (COVID-19), Feb. 2020.
[4] S. Deldari, D. V. Smith, H. Xue, and F. D. Salim. Time series change point detection with self-supervised contrastive predictive coding. In *Proceedings of the Web Conference 2021*, pages 3124–3135, 2021.
[5] J. Elarde, J.-S. Kim, H. Kavak, A. Züfle, and T. Anderson. Change of human mobility during covid-19: A united states case study. *PLOS ONE*, 16(11), 2021.
[6] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Artificial Intelligence and Statistics*, pages 297–304, Mar. 2010.
[7] K. Healy. County and state fips codes https://github.com/kjhealy/fips-codes.
[8] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019.
[9] R. J. Hyndman, A. V. Kostenko, et al. Minimum sample size requirements for seasonal forecasting models. *foresight*, 6(Spring):12–15, 2007.
[10] Y. Kang, S. Gao, Y. Liang, M. Li, J. Rao, and J. Kruse. Multiscale dynamic human mobility flow dataset in the us during the covid-19 epidemic. *Scientific data*, 7(1):1–13, 2020.
[11] F. Kato, Y. Cao, and M. Yoshikawa. Pct-tee: Trajectory-based private contact tracing system with trusted execution environment. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(2):1–35, 2021.
[12] R. Ke, E. Romero-Severson, S. Sanche, and N. Hengartner. Estimating the reproductive number R0 of SARS-CoV-2 in the United States and eight European countries and implications for vaccination. *Journal of Theoretical Biology*, 517:110621, 2021.
[13] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.
[14] C. C. Kerr, R. M. Stuart, D. Mistry, R. G. Abeysuriya, et al. Covasim: an agent-based model of covid-19 dynamics and interventions. *PLOS Computational Biology*, 17(7):e1009149, 2021.
[15] B. Millidge, A. Seth, and C. L. Buckley. Predictive Coding: A Theoretical and Experimental Review, July 2022.
[16] M. Mokbel, M. Sakr, L. Xiong, A. Züfle, et al. Dagstuhl reports, vol. 12, issue 1 issn 2192-5283. 2022.
[17] C. J. Murray and P. Piot. The potential future of the covid-19 pandemic: will sars-cov-2 become a recurrent seasonal infection? *Jama*, 325(13):1249–1250, 2021.
[18] B. Nikparvar, M. Rahman, F. Hatami, J.-C. Thill, et al. Spatio-temporal prediction of the covid-19 pandemic in us counties: modeling with a deep lstm neural network. *Scientific reports*, 11(1):1–12, 2021.
[19] Nytimes. Nytimes/covid-19-data: An ongoing repository of data on coronavirus cases and deaths in the u.s.
[20] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
[21] J. Pesavento, A. Chen, R. Yu, J.-S. Kim, H. Kavak, T. Anderson, and A. Züfle. Data-driven mobility models for covid-19 simulation. In *ACM SIGSPATIAL International Workshop on Advances in Resilient and Intelligent Cities*, pages 29–38, 2020.
[22] S. Rambhatla, S. Zeighami, K. Shahabi, C. Shahabi, and Y. Liu. Toward accurate spatiotemporal covid-19 risk scores using high-resolution real-world mobility data. *ACM TSAS*, 8(2):1–30, 2022.
[23] M. Rivière, A. Joulin, P.-E. Mazaré, and E. Dupoux. Unsupervised pretraining transfers well across languages, 2020.
[24] C. P. Robert and G. Casella. The metropolis—hastings algorithm. In *Monte Carlo statistical methods*, pages 231–283. Springer, 1999.
[25] R. K. Singh, M. Rani, A. S. Bhagavathula, et al. Prediction of the covid-19 pandemic for the top 15 affected countries: Advanced autoregressive integrated moving average (arima) model. *JMIR Public Health Surveill*, 6(2):e19115, May 2020.
[26] J. Tolles and T. Luong. Modeling epidemics with compartmental models. *Jama*, 323(24):2515–2516, 2020.
[27] F.-M. Tseng and G.-H. Tzeng. A fuzzy seasonal arima model for forecasting. *Fuzzy Sets and Systems*, 126(3):367–376, 2002.