StreamCast: Fast and Online Mining of Power Grid Time Sequences

Bryan Hooi[*][†] Hyun Ah Song[*] Amritanshu Pandey[‡] Marko Jereminov[‡] Larry Pileggi[‡]
Christos Faloutsos[*]

Abstract

How can we efficiently forecast the power consumption of a location for the next few days? More challengingly, how can we forecast the power consumption if the temperature increases by $10°C$, the number of appliances in the grid increase by 20%, and voltage levels increase by 5%? Such 'what-if scenarios' are crucial for future planning, to ensure that the grid remains reliable even under extreme conditions. Our contributions are as follows: **1) Domain knowledge infusion:** we propose a novel *Temporal BIG* model that extends the physics-based BIG model, allowing it to capture changes over time, trends, and seasonality, and temperature effects. **2) Forecasting:** our StreamCast algorithm forecasts multiple steps ahead and outperforms baselines in accuracy. Our algorithm is online, requiring constant update time per new data point and bounded memory. **3) What-if scenarios and anomaly detection:** our approach can handle scenarios in which the voltage levels, temperature, or number of appliances change. It also spots anomalies in real data, and provides confidence intervals for its forecasts, to assist in planning for various scenarios. Experimental results show that StreamCast has 27% lower forecasting error than baselines on real data, scales linearly, and runs in 4 minutes on a time sequence of 40 million points.

1 Introduction

The smart electrical grid is a set of technologies designed to improve the efficiency and security of power delivery. Estimates [1] suggest that reducing outages in the U.S. grid could save \$49 billion per year, reduce emissions by 12 to 18%, while improving efficiency could save an additional \$20.4 billion per year. A key part of achieving this goal is to use monitoring data to accurately model the behavior of the grid and forecast future load requirements, especially under extreme conditions such as changes in temperature, number of appliances in the grid, or voltage patterns. This allows the grid to remain reliable even under adverse conditions.

A major challenge is scalability - power systems data can be both high-volume and received in real time. This motivates us to develop fast methods that work in this online (or streaming) setting. Rather than operating on an entire dataset at once, online algorithms allow input that arrives over time as a continuous stream of data points. When each new data point is received, the algorithm updates itself - for our algorithm, each update requires constant time, and bounded memory (i.e. it does not need to remember the entire history of the time series).

Hence, our goal is an online algorithm for modelling and forecasting power consumption of a location. The input is a stream over time of real and imaginary voltage and current values:

Informal Problem 1. (Online Model Estimation)

- ***Given:*** *A continuous stream of values of real and imaginary current* $(I_r(t), I_i(t))$*, voltage* $(V_r(t), V_i(t))$*, and temperature* $T(t)$*, for* $t = 1, 2, \cdots$
- ***Estimate:*** *Time-varying parameters of a physics-based model of electrical load behavior that accurately explains the observed values.*

Our model is based on the circuit theoretic BIG model [9], which characterizes load in the electric grid by modeling its voltage sensitivities. Importantly, the use of physics-based models together with current and voltage state variables improves interpretability. For instance, real power consuming loads such as light bulbs can be interpreted as the contribution of the conductance (G) parameter, while susceptance (B), the reactive power component, represents the contributions of motors or capacitors in the grid.

The fitted model is used to forecast future values:

Informal Problem 2. (Multi-step Forecasting)

- ***Given:*** *values of current* $(I_r(t), I_i(t))$*, voltage* $(V_r(t), V_i(t))$*, and temperature* $T(t)$ *for* $t = 1, \cdots, N$*, and given temperature forecasts for the next* N_f *time steps (i.e. for* $t = N+1, \cdots, N+N_f$*),*

[*]School of Computer Science, Carnegie Mellon University
[†]Department of Statistics, Carnegie Mellon University
[‡]Deptartment of Electrical and Computer Engineering, Carnegie Mellon University

- ***Forecast:*** *voltage and current for N_f time steps in the future; i.e. $(V_r(t), V_i(t))$ and $(I_r(t), I_i(t))$ for $t = N+1, \cdots, N+N_f$.*

Due to our physics-based model, our algorithm can handle **what-if scenarios** in which the temperatures or voltages change, which is useful for future planning.

Informal Problem 3. (What-if Scenarios)
- ***Given:*** *current, voltage and temperature data, as above,*
- ***Forecast:*** *future values of voltage and current, under the condition that, e.g., temperature increases by $10°C$, and voltage levels increase by 5%.*

Our contributions are as follows:
1. **Domain knowledge infusion:** we propose a novel, *Temporal BIG* model that extends the physics-based BIG model, allowing it to capture trends, seasonality, and temperature effects.
2. **Forecasting:** our StreamCast algorithm forecasts multiple steps ahead and outperforms baselines in accuracy by 27% or more. StreamCast is online, requiring linear time and bounded memory.
3. **What-if scenarios and anomaly detection:** our approach accurately handles scenarios in which the voltage levels, temperature, or number of appliances change. We also use it to detect anomalies in a real dataset.

Reproducibility: our code is publicly available at `www.andrew.cmu.edu/user/bhooi/power.tar`.

2 Background and Related Work

2.1 Related Work

The BIG model for electrical load The constant power *PQ* model [16] is a common approach for power grid modelling. However, industry experience has shown that it incorrectly characterizes load behavior [14]. Recent advances [3] have shown that load behavior at a given time can be accurately described by a linear relationship between current and voltage. From circuit theory, this can be represented by a parallel or series combination of susceptance (B) and conductance (G). This captures both magnitude and angle, in contrast to existing traditional load models [16].

Time series forecasting Classical time-series forecasting methods include autoregression (AR)-based methods, including ARMA, ARIMA [2], seasonal ARIMA [2], and vector autoregression (VAR) [5]. Exponential smoothing (ETS) models [22], including Holt-Winters [22] capture trends and seasonal patterns. Other methods include Kalman filtering [11], Hidden Markov Models (HMMs) [12], and non-linear dynamical systems [15].

For power grid load modelling, common approaches include AR models [7, 17], ETS [10], and neural networks [6]. [18, 23] use weather data as inputs. [19] uses tensor decomposition to forecast power grid time sequences.

Contrast with existing literature Other than [19], all methods above do not use physics-based electrical models, and do not consider what-if scenarios, while our approach does both. Compared to [19], our approach (which is completely different from their tensor-based approach) additionally allows for weather data, is an online algorithm, and produces confidence intervals.

Table 1: **StreamCast captures the listed properties**. AR++ refers to ARIMA, seasonal ARIMA etc.

Properties	AR++ [2]	Kalman/LDS [11]	Weather-based [18, 23]	HMM++ [12]	PowerCast [19]	StreamCast
Forecasting	✓	✓	✓	✓	✓	✔
Seasonal patterns	✓	?	✓		✓	✔
Physics-based model					✓	✔
Weather-based			✓			✔
Online algorithm		✓				✔
Confidence intervals	✓	✓		✓		✔
What-if scenarios					✓	✔

2.2 Background

BIG model The BIG model [9] models the current as a linear function of the voltage, parameterized by the BIG parameters: susceptance (B), conductance (G), and a current offset (α_r, α_i). G can be interpreted as the component contributing to real power consuming loads (e.g. due to light-bulbs), while B can be interpreted as the contribution of the reactive power component (e.g. due to motors or capacitors):

$$\begin{aligned} I_r(t) &= G \cdot V_r(t) - B \cdot V_i(t) + \alpha_r + \text{noise} \\ I_i(t) &= B \cdot V_r(t) + G \cdot V_i(t) + \alpha_i + \text{noise} \end{aligned} \tag{2.1}$$

Holt-Winters model The Holt-Winters model [22] models a univariate time series $x(t)$ with seasonal structure. The key idea is to model $x(t)$ as the sum of a non-seasonal or *level* component $l(t)$ and a *seasonal* component $s(t)$. The level component changes according to smooth trends, while the seasonal component is approximately periodic with period m (e.g. $m = 24$ for hourly data with daily seasonality). Smooth trends over time are modelled by a linear trend $b(t)$, representing the rate of change of $l(t)$. Full details can be found in [22].

3 Proposed Model

Table 2 shows the symbols used in this paper.

Table 2: Symbols and definitions

Symbols	Definitions
N	Number of time ticks in time sequences
I_r, I_i, V_r, V_i	Real and imaginary current and voltage
B, G, α_r, α_i	BIG parameters (susceptance, conductance, offset to I_r and I_i)
$\theta(t)$	Parameter vector $(B, G, \alpha_r, \alpha_i)$ at time t
$y(t), X(t)$	BIG in linear model form; see Eq. (3.3)
$\theta_L(t)$	Nonseasonal part of $\theta(t)$
$\theta_S(t)$	Seasonal part of $\theta(t)$
$\theta_T(t)$	Trend in $\theta_L(t)$
$\theta_W(t)$	Weather part of $\theta(t)$
w	Weather coefficients
T_0	Temperature threshold
N_{init}	Initialization period

3.1 Proposed Dynamic BIG Model Consider the static BIG model in Eq. (2.1). Its parameters B, G, α_r, α_r can be interpreted as types of load; but in practice, these should change over time as appliances are switched on and off, or usage levels change. How do we add temporal structure to this model? A natural step is to replace B, G, α_r, α_r by time series $B(t), G(t), \alpha_r(t), \alpha_r(t)$. Eq. (2.1) becomes:

$$\begin{aligned} I_r(t) &= G(t) \cdot V_r(t) - B(t) \cdot V_i(t) + \alpha_r(t) + \text{noise} \\ I_i(t) &= B(t) \cdot V_r(t) + G(t) \cdot V_i(t) + \alpha_i(t) + \text{noise} \end{aligned} \tag{3.2}$$

For notational simplicity, rewrite this equivalently as:

$$\underbrace{\begin{pmatrix} I_r(t) \\ I_i(t) \end{pmatrix}}_{y(t)} = \underbrace{\begin{pmatrix} V_r(t) & -V_i(t) & 1 & 0 \\ V_i(t) & V_r(t) & 0 & 1 \end{pmatrix}}_{X(t)} \underbrace{\begin{pmatrix} G(t) \\ B(t) \\ \alpha_r(t) \\ \alpha_i(t) \end{pmatrix}}_{\theta(t)} + \text{noise} \tag{3.3}$$

Now, changes in usage (e.g. appliances switching on and off) correspond to changes in $\theta(t)$. What temporal patterns do we need to capture? Intuitively, some appliances follow daily seasonality (e.g. lights used during working hours), while others follow slow-moving trends, e.g. a gradual increase in load due to population growth. Hence, like in Holt-Winters, we decompose $\theta(t)$ into a nonseasonal **level** part $\theta_L(t)$, and a **seasonal** part $\theta_S(t)$; hence, $\theta(t) = \theta_L(t) + \theta_S(t)$. Here $\theta_L(t)$ changes according to smooth trends, while $\theta_S(t)$ has seasonal patterns, with period m.

To model smooth trends (e.g. population growth), we additionally define a **trend** term $\theta_T(t)$, which approximates the change in $\theta_L(t)$ from one time tick to the next; i.e. $\theta_L(t) \approx \theta_L(t-1) + \theta_T(t)$. Then, our assumptions under this model are that:

A1: $y(t) \approx X(t)(\theta_L(t) + \theta_S(t))$ (Low noise; see (3.3))

A2: $\theta_L(t) \approx \theta_L(t-1) + \theta_T(t)$ (Level moves wrt. trend)

A3: $\theta_T(t) \approx \theta_T(t-1)$ (Trends change smoothly)

A4: $\theta_S(t) \approx \theta_S(t-m)$ (Seasonality)

We will formalize these as soft constraints in our optimization objective in Section 4.

3.2 Dynamic BIG with Temperature Model Let $T(t)$ denote the temperature at time t. When temperature increases above a threshold, electricity demand tends to increase due to the use of air conditioning. To capture this, we introduce **weather coefficients** w and a **temperature threshold** T_0: temperature over the threshold linearly adds to a weather component $\theta_W(t)$. Hence, we replace assumption A1 with:

A1': $y(t) \approx X(t)(\theta_L(t) + \theta_S(t) + \theta_W(t))$, where $\theta_W(t) = w \cdot \max(0, T(t) - T_0)$

The max function ensures that only temperatures above the threshold contribute to the weather component.

4 Proposed Optimization Objective

We now define our optimization objective based on our assumptions A1' to A4. All norms are L2 norms, for later computational simplicity.

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_1 + \lambda L_2 + \mu \mathcal{L}_3 + \nu \mathcal{L}_4, \text{ where:} \\ \mathcal{L}_1 &= \sum_{t=1}^{N} \|y(t) - X(t)(\theta_L(t) + \theta_S(t) + \theta_W(t))\|^2 \\ \mathcal{L}_2 &= \sum_{t=2}^{N} \|\theta_L(t) - \theta_L(t-1) - \theta_T(t)\|^2 \\ \mathcal{L}_3 &= \sum_{t=1}^{N} \|\theta_T(t) - \theta_T(t-1)\|^2 \\ \mathcal{L}_4 &= \sum_{t=m+1}^{N} \|\theta_S(t) - \theta_S(t-m)\|^2 \end{aligned} \tag{4.4}$$

Here $\lambda, \mu, \nu > 0$ are hyperparameters (which we end up tuning indirectly rather than directly; see Section 5).

The problem to solve is then:

$$\underset{\theta_L,\theta_T,\theta_S,w,T_0}{\text{minimize}}\ \mathcal{L} \tag{4.5}$$

5 Proposed StreamCast Algorithm

How do we approximately minimize $\mathcal{L}$ in an efficient and online manner?

5.1 Overview In this section, we outline our proposed StreamCast. StreamCast has two steps: 1) an offline step TempFit, which takes a subset of N_{init} data points, and fits w and T_0; 2) an online 'extension' stage StreamFit: upon receiving each new data point at time t, this updates $\theta_L, \theta_T, \theta_S$ according to Eq. (5.7).

Algorithm 1 StreamCast

```
Require: Streams $V_r, V_i, I_r, I_i$
1: $w, T_0 \leftarrow$ TempFit($V_r(1{:}\ N_{\text{init}}), V_i(1{:}\ N_{\text{init}})$
                             $I_r(1{:}\ N_{\text{init}}), I_i(1{:}\ N_{\text{init}})$)
2: while input at time $t$ is received: do
3:     Update $\theta_L(t), \theta_T(t), \theta_S(t)$ using StreamFit
4: end while
```

We will describe StreamFit in Section 5.2, and TempFit in Section 5.3.

5.2 Streaming Optimization (StreamFit) StreamFit estimates θ_L, θ_T and θ_S for fixed temperature parameters w, T_0, by minimizing our objective, Eq. (4.5). Note that Eq. (4.5) could in theory be minimized using least squares; however, this is not online, and is also much too slow as it contains $O(N)$ unknowns for N time points, which would then take around $O(N^3)$ time. We would like a linear-time algorithm with constant update time per data point.

Our approach is to split up the objective over t, and take gradient update steps with respect to the term for $t = 1, 2, \cdots$ successively. At time t, we take a gradient step with respect to only the terms of $\mathcal{L}$ corresponding to time t. This allows the fitted parameters to 'track' the true values over time as we perform gradient updates. Meanwhile, each update is highly efficient as it only involves a single term of the objective function. Assume that we have fit $\theta_L, \theta_T, \theta_S$ up to time $t-1$ and are fitting them at time t. The component of $\mathcal{L}$ for time t is:

(5.6)
$$\begin{aligned}\mathcal{L}(t) &= \|y(t) - X(t)(\theta_L(t) + \theta_S(t) + \theta_W(t))\|^2\\ &\quad + \lambda\|\theta_L(t) - \theta_L(t-1) - \theta_T(t)\|^2\\ &\quad + \mu\|\theta_T(t) - \theta_T(t-1)\|^2 + \nu\|\theta_S(t) - \theta_S(t-m)\|^2\end{aligned}$$

To do gradient descent at time t, we start from the previously learned values and take a gradient step. Hence, we start by assuming the previous trend ($\theta_T(t) = \theta_T(t-1)$) and seasonality ($\theta_S(t) = \theta_S(t-m)$), while for the level we follow a 'default extrapolation' of starting at the previous level and following the previous trend ($\theta_L(t) = \theta_L(t-1) + \theta_T(t-1)$).

Next, we want to move in the gradient direction with respect to minimizing $\mathcal{L}$. Computing the gradients of (5.6) is straightforward and results in the update equations: letting $\hat{y}(t) = X(t)(\theta_L(t-1) + \theta_T(t-1) + \theta_S(t-m) + \theta_W(t))$,

(5.7)
$$\begin{aligned}\theta_L(t) &= \underbrace{\theta_L(t-1) + \theta_T(t-1)}_{\text{default extrapolation}} + \alpha \underbrace{X(t)^T(y(t) - \hat{y}(t))}_{\text{gradient update}}\\ \theta_T(t) &= \underbrace{\theta_T(t-1)}_{\text{previous trend}} + \beta \underbrace{(\theta_L(t) - \theta_L(t-1) - \theta_T(t-1))}_{\text{gradient update}}\\ \theta_S(t) &= \underbrace{\theta_S(t-m)}_{\text{previous seasonality}} + \gamma \underbrace{X(t)^T(y(t) - \hat{y}(t))}_{\text{gradient update}}\end{aligned}$$

$\alpha, \beta, \gamma > 0$ are 'learning-rate' tuning parameters that replace λ, μ, ν. We will consider how to set these, and how to initialize θ_L, θ_T and θ_S, in Section 5.3.

5.3 Temperature Model Optimization (TempFit) In TempFit, we optimize the temperature coefficient w and threshold T_0 using an alternating approach. First, fixing w and T_0, we solve for θ_L, θ_T and θ_S to minimize $\mathcal{L}$. We then do the reverse (fixing θ_L, θ_T and θ_S and solving for w and T_0), until convergence, as shown in Algorithm 2. The former step of solving for θ_L, θ_T and θ_S given w and T_0 is done using StreamFit.

Algorithm 2 TempFit

```
Require: $V_r, V_i, I_r, I_i$
1: while not converged do
2:     Solve $\theta_L, \theta_T, \theta_S$ using Eq. (5.7) (StreamFit)
3:     Solve $w$, $T_0$ by minimizing Eq. (5.8)
4: end while
```

It remains to solve for w and T_0 for fixed $\theta_L, \theta_T, \theta_S$. Define $r(t) = y(t) - X(t)(\theta_L(t) + \theta_S(t))$. Then, since changing w and T_0 only affects the $\mathcal{L}_1$ loss term, minimizing $\mathcal{L}$ is equivalent to minimizing:

$$\mathcal{L}' = \sum_{t=1}^{N} \|r(t) - X(t) \cdot w \cdot \max(0, T(t) - T_0)\|^2 \tag{5.8}$$

Minimizing over w is a straightforward least squares problem. Minimizing over T_0 is less straightforward since $\max(0, T(t) - T_0)$ is nonlinear in T_0. However, temperature is typically given to 0 or 1 decimal of precision (roughly the precision of most thermometers), and varies within a fairly narrow range, e.g. $0°C$ to

$35°C$. Hence, it suffices to try all thresholds between $\min_t T(t)$ and $\max_t T(t)$ in intervals of $0.1°C$, and select among these to minimize $\mathcal{L}'$.

To complete our algorithm, we need to explain how to choose α, β, γ, and initial values for $\theta_L(t), \theta_T(t), \theta_S(t)$. We select the former using nonlinear optimization (Levenberg-Marquadt algorithm [13]) of $\mathcal{L}$. For the latter, it can be verified that the objective $\mathcal{L}$ is a quadratic in these initial values, so we solve for them using least squares. We do these in Line 2 of TempFit, then fix these values subsequently for the rest of the algorithm (Line 2-4 of StreamCast).

5.4 Forecasting Step (Forecast) Given fitted model parameters up to time N, how do we forecast future currents $\hat{I}_r(t), \hat{I}_i(t)$, for $t = N+1, \cdots, N+N_f$?

We first forecast voltages $\hat{V}_r(t), \hat{V}_i(t),\ t = N+1, \cdots, N+N_f$ using standard univariate Holt-Winters. Then, for $k = 1, 2, \cdots$ we forecast $\theta(N+k)$ as the sum of last estimated level $\theta_L(N)$, trend $\theta_T(N)$, last estimated seasonality at the corresponding position $\theta_S(N-m+1+((k-1) \mod m))$, where $\mod$ is the modulo function, and temperature component $w \cdot \max(0, T(N+k) - T_0)$. Finally, we forecast $\hat{I}_r(t), \hat{I}_i(t)$ using the BIG model, Eq. (3.2).

5.5 Extensions

Anomaly Detection To detect anomalies, we compute an anomaly score at each time. Intuitively, the larger the error between the fitted and actual values, the more anomalous a point is. The fitted values $\hat{I}_r(t), \hat{I}_i(t)$ are obtained by plugging the learned parameters into the BIG equations, Eq. (3.2). The real and imaginary errors are then $E_r(t) = I_r(t) - \hat{I}_r(t)$, and $E_i(t) = I_i(t) - \hat{I}_i(t)$. The anomalousness at time t is then the sum of real and imaginary errors at time t, each in units of inter-quartile ranges (IQRs[1]):

$$\text{anomalousness}(t) = \frac{E_r(t)}{\text{IQR}(E_r(1\colon N))} + \frac{E_i(t)}{\text{IQR}(E_i(1\colon N))}$$

While any threshold may then be used, we can follow common practice of designating deviation of $\geq 2 \times IQR$ as outliers, resulting in a threshold of 4 for our anomalousness score. In Section 6.4 we show a clear anomaly found in the LBNL dataset.

Confidence Intervals Confidence intervals allow us to provide lower and upper bounds that contain future values, with e.g. 95% confidence. We use the past distribution of residuals, $I_r(t) - \hat{I}_r(t)$ and $I_i(t) - \hat{I}_i(t)$, as an estimate of residuals in the future. Thus, we sample a 'possible future' by repeatedly sampling from this distribution of past residuals, and treating the sampled values as residuals at time $N+1$, repeating this process for $N+2$, and so on. We sample 1000 possible futures in this way. To generate $(1-\alpha)$ confidence intervals, we use the empirical $\alpha/2$ and $(1-\alpha/2)$-quantiles of these possible futures. The results are shown in Figure 6.

6 Experiments

We design experiments to answer the questions:

- **Q1. Forecasting accuracy:** how accurately does StreamCast forecast, based on real data?
- **Q2. Scalability:** how does the algorithm scale with the data size?
- **Q3. What-if scenarios:** does StreamCast give accurate results under what-if scenarios, and detect real anomalies in real data?

Our code and links to datasets are publicly available at `www.andrew.cmu.edu/user/bhooi/power.tar`. Experiments were done on a 2.4 GHz Intel Core i5 Macbook Pro, 16 GB RAM running OS X 10.11.2. We set N_{init} to $10m$ (i.e. 10 days) in our experiments.

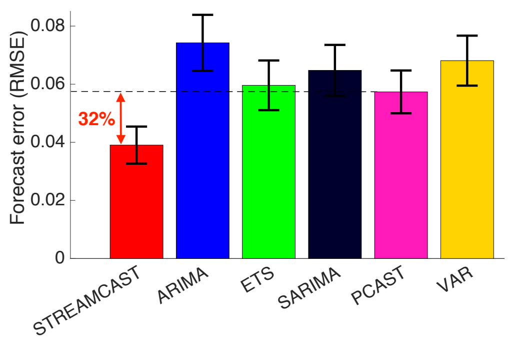
6.1 Data We use the following two datasets:

- **CMU data:** ($N = 648$) hourly voltage and current for the Carnegie Mellon University (CMU) campus for 23 days, from July 29, 2016 to August 20, 2016. Voltage angle is unavailable for this data, so here V_r is the voltage magnitude and $V_i = 0$.
- **LBNL data:** ($N = 3168$) from the Lawrence Berkeley National Laboratory (LBNL) Open μPMU project [20], from October 1, 2015 to October 11, 2015. The data is originally at 120Hz, but we downsample it to one sample every 5 minutes (where each new data point is the mean of the raw data within those 5 minutes).

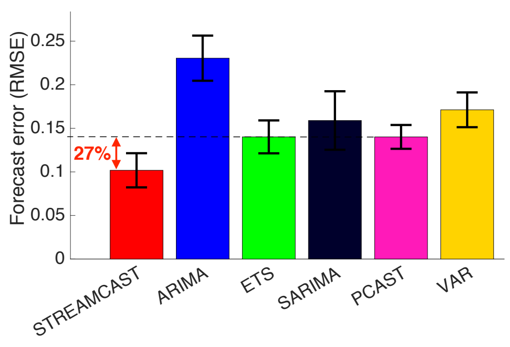
6.2 Q1. Forecasting accuracy

Baselines our baselines are ARIMA [2], Holt-Winters (ETS) [4], seasonal ARIMA [2], PowerCast (PCAST) [19] (a recent tensor-based power grid forecasting approach), and vector autoregression (VAR), which uses temperature data and the voltage time sequences as input. Following standard practice, the ARIMA, SARIMA, and VAR orders are selected using AIC (Akaike information criterion) [8], and the Holt-Winters hyperparameters are selected using nonlinear

[1]The IQR is the difference between 75% and 25% quartiles, used as a more robust measure of spread compared to standard deviation. [21]



(a) CMU data



(b) LBNL data

Figure 1: **StreamCast forecasts accurately:** it has at least 27% lower forecasting error than baselines. Error bars show 1 standard deviation.

optimization. For PowerCast, we follow the original paper in setting $N_w = 5, \sigma = 0.5$.

Experimental setup in each trial, an algorithm is given the data for the first N days and forecasts I_r and I_i for each time point of the $(N + 1)$th day, where we average each algorithm's accuracy over $N = 11, 12, \cdots, 20$ for CMU and $N = 4, 5, \cdots, 8$ for LBNL. The forecasts are compared with the true values using normalized RMSE: $\text{RMSE}(x, \hat{x}) = \sqrt{\|x - \hat{x}\|_2^2 / \|x\|_2^2}$, where $x = [I_r \quad I_i]$ contains I_r and I_i values stacked into a single vector.

Results: Figure 1 shows that StreamCast outperforms the baselines in both datasets, with at least 27% lower RMSE. The tensor-decomposition based PCAST, which is also physics-based, is the second-best performer, suggesting that physics-based models may be beneficial for forecasting.

6.3 Q2. Scalability We run StreamCast on a version of our CMU dataset, duplicated repeatedly so as to produce larger datasets. We run StreamCast on time series of sizes as plotted on the x-axis of Figure 2, from around 1 million to around 40 million. The plot is

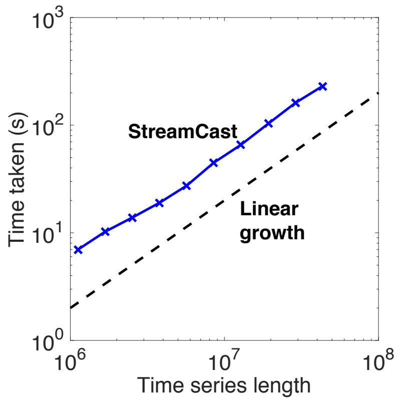


Figure 2: **StreamCast is fast and scales linearly:** growth parallel to the diagonal indicates linear growth.

parallel to the diagonal, indicating linear growth.

StreamCast takes less than 4 minutes for the trial of size 40 million, making it scalable to large datasets.

6.4 Q3. What-if scenarios

Changing temperature and number of appliances: how can we forecast under the scenario that temperature increases by $10°C$, and number of appliances increases by 20%? Such scenarios are useful for future planning, but standard forecasting methods cannot handle them. The BIG parameter G represents the contribution of the conductive load component (e.g. light-bulbs) and B as the contribution of the reactive load component (e.g. motors), so we examine the results of increasing either G, B, or temperature, and plot how the forecasts change under each scenario.

The results are shown in Figure 3. The left plots are for increasing G; the center plots for changing B, and the right plots for increasing temperature. Upper plots are for I_r while lower plots are for I_i. In each plot, the colored lines correspond to different amounts of increase: e.g. $1.1 \times G$ means that the amount of reactive load on campus increased by 10%. The results show that: 1) when G is increased, only I_r increases, but not I_i; 2) when B is increased, only I_i increases, but not I_r; 3) when temperature increases, both I_r and I_i increase. All three results are intuitive, given that in the CMU dataset we have $V_i = 0$; it can be verified from (3.2) that in this case I_r should be influenced by changes in G, but not by changes in B.

Changing voltage An important goal in practice is to ensure that the system can make accurate predictions under changes to voltage levels. To test our model, we use an electrical system simulator, SUGAR [16]. The simulator uses physics-based models for different elec-

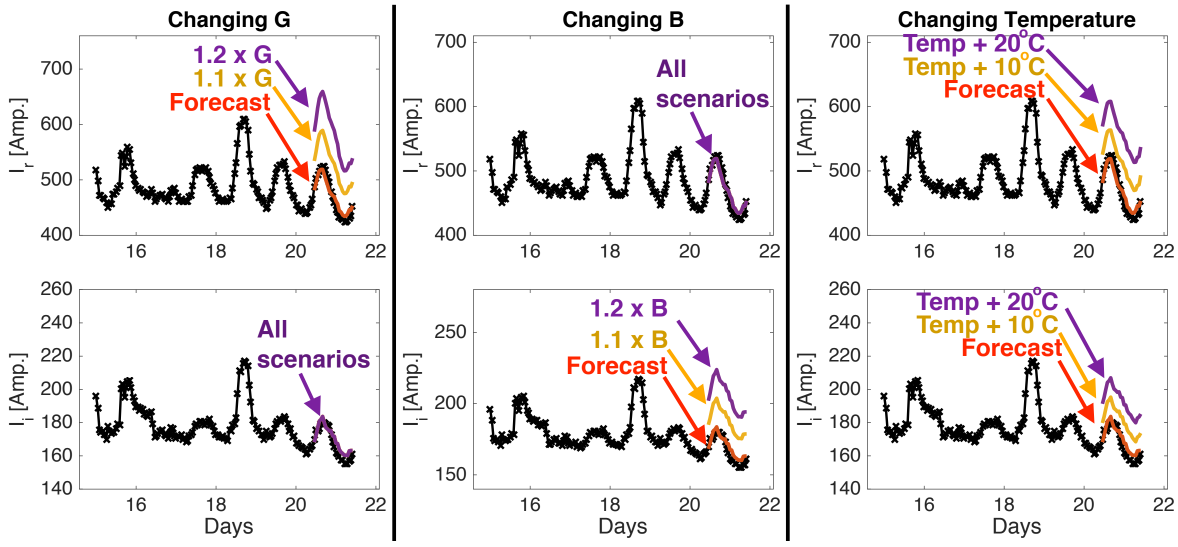


Figure 3: **StreamCast can handle forecasting under what-if scenarios:** the plots show the results of 1) increasing G; 2) increasing B; and 3) increasing temperature.

trical equipment: we specify 10 motors and additional load with maximum resistance 50Ω, varying sinusoidally with daily periodicity. SUGAR generates realistic time series currents for an electrical system given specified input voltages V_r, V_i. In order to have a realistic voltage series, our input voltages are the voltage series from our CMU dataset. We obtain currents (I_r, I_i) as outputs from this simulation.

To test StreamCast, we fit it on (V_r, V_i, I_r, I_i), but then evaluate its RMSE on a different (V'_r, V'_i, I'_r, I'_i) in which V_r and V_i are defined in one of two ways: 1) **Increase:** $V'_r = 1.05 \times V_r$, $V'_i = 1.05 \times V_i$. 2) **Decrease:** $V'_r = 0.95 \times V_r$, $V'_i = 0.95 \times V_i$. We then obtain I'_r and I'_i from the same electrical simulation under voltages V'_r and V'_i.

Standard forecasting methods would not work as baselines, since an electrical model is needed to accurately predict what happens to I when V changes. Hence, we use the following baselines: 1) PQ: this is the same as our StreamCast approach, but substituting the BIG model with the more common PQ electrical model [16]. 2) PowerCast [19] as in the previous section; 3) Window*k*: this fits the *static* BIG model (Section 2.2) to short time windows of size k, where $k = 2, 4, 8, 16$.

Figure 4 shows the fit of StreamCast, PQ and Window4 against the true values (the remaining methods are not plotted for visibility, but their RMSE is in Table 3), and Table 3 shows the RMSE of each method against the true values. StreamCast outperforms the baselines by a clear margin. Because PQ is a constant-power model, increases in voltage magnitude necessarily lead to the model predicting a decrease in current of a similar magnitude. However, in practice, both voltage and current can change in the same direction, in which case the PQ model makes the wrong qualitative predictions. The Window*k* baselines tend to overfit to near-term behavior due to their use of windows; this explains the high variance over time in Figure 4.

Test case	StreamCast	PQ	PowerCast	Window2	Window4	Window8	Window16
Increase	**<u>0.19</u>**	7.13	5.26	11.18	4.94	4.95	4.72
Decrease	**<u>0.27</u>**	7.69	5.66	9.46	5.78	5.56	4.58

Table 3: **StreamCast is accurate even under different voltage levels:** here the methods are tested on what-if scenarios with different voltage levels. Bold underline shows the best performer. Error values are given as **percentage** RMSE (i.e. RMSE ×100).

Anomaly Detection Section 5.5 explains how to detect anomalies under our method; Figure 5 shows the results on the LBNL dataset, where our method outputs a plot ('anomalousness') of the anomaly score over time. There is a large spike in anomalousness around day 2; note that the anomaly is not at all visible from only the current time series. In the voltage time series, however, we find a 2-second period of quick oscillations between negative and positive values exactly at the time of the anomaly. Follow-up analysis reveals that the oscillation is likely an error in the measuring device: the complex angle of voltage is synchronized with the rest of the system via a GPS signal, and in case of loss of this

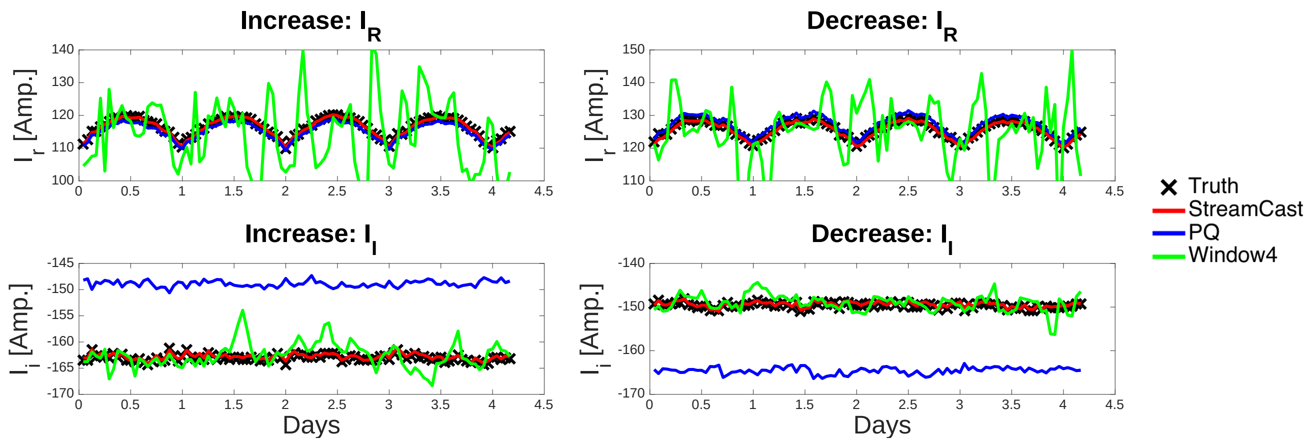


Figure 4: **StreamCast accurately responds to changes in voltage:** forecasts of I_r and I_i by each method vs. true values, when voltage was increased or decreased by 5%. StreamCast fits the data more accurately then baselines. RMSE results (with additional baseline methods) are in Table 3.

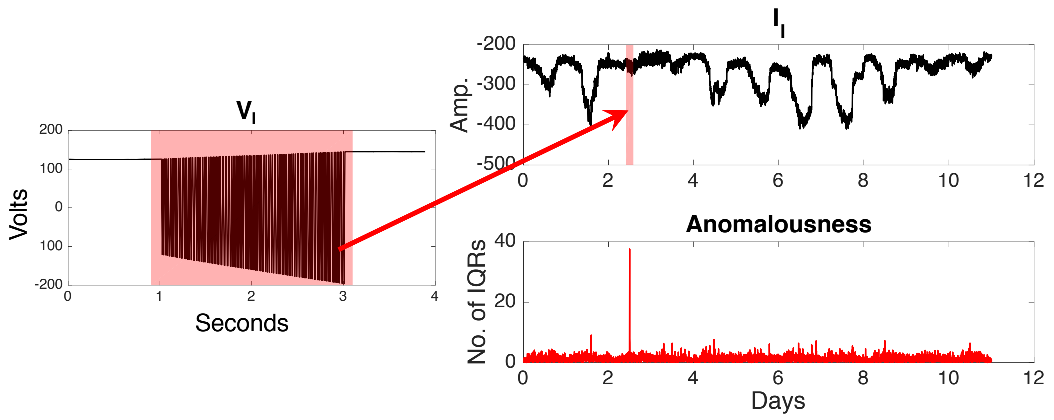


Figure 5: **StreamCast detects an anomaly in the LBNL dataset**. It corresponds to a 2-second period where voltage rapidly oscillates between negative and positive values.

GPS signal, we may observe oscillations such as those in Figure 5, which explains why the voltage magnitude remains stable while the angle oscillates.

Confidence Intervals Confidence intervals are useful for obtaining ranges of predictions: e.g. when monitoring the grid. Section 5.5 explains how we compute them. 95% and 99% confidence intervals on our CMU dataset are shown in Figure 6.

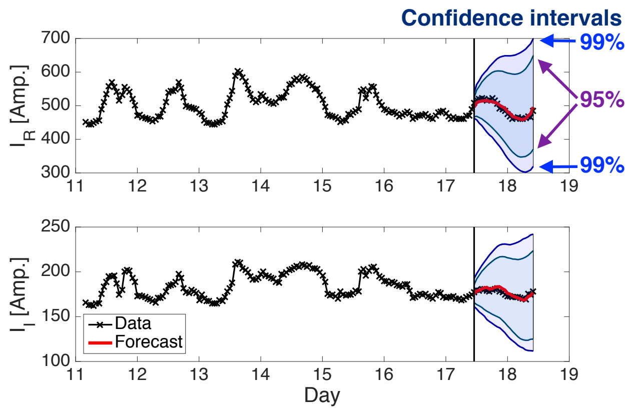


Figure 6: **StreamCast provides confidence intervals.**

7 Conclusion

Our contributions are as follows:

1. **Domain knowledge infusion:** we propose a novel, *Temporal BIG* model that extends the physics-based BIG model, allowing it to capture changes over time, trends, seasonality, and temperature effects.
2. **Forecasting:** our StreamCast algorithm forecasts multiple steps ahead and outperforms baselines in accuracy. StreamCast is online, requiring linear time and bounded memory.
3. **What-if scenarios and anomaly detection:** our approach accurately handles scenarios in which

the voltage levels, temperature, or number of appliances change. We also use it to detect anomalies in a real dataset. Finally, StreamCast provides confidence intervals for its forecasts, to assist in planning for various scenarios.

Reproducibility: our code is publicly available at `www.andrew.cmu.edu/user/bhooi/power.tar`.

8 Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1247489, and by the Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, and in part by the Defense Advanced Research Projects Agency (DARPA) under award no. FA8750-17-1-0059 for the RADICS program. Any opinions, findings, and 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, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] S. M. Amin. Us grid gets less reliable [the data]. *IEEE Spectrum*, 48(1):80–80, 2011.

[2] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control.* John Wiley & Sons, 2015.

[3] D. M. Bromberg, M. Jereminov, X. Li, G. Hug, and L. Pileggi. An equivalent circuit formulation of the power flow problem with current and voltage state variables. In *PowerTech, 2015 IEEE Eindhoven*, pages 1–6. IEEE, 2015.

[4] R. G. Brown. *Statistical forecasting for inventory control.* McGraw/Hill, 1959.

[5] J. D. Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[6] H. S. Hippert, C. E. Pedreira, and R. C. Souza. Neural networks for short-term load forecasting: A review and evaluation. *IEEE Transactions on power systems*, 16(1):44–55, 2001.

[7] S.-J. Huang and K.-R. Shih. Short-term load forecasting via arma model identification including non-gaussian process considerations. *IEEE Transactions on power systems*, 18(2):673–679, 2003.

[8] C. M. Hurvich and C.-L. Tsai. A corrected akaike information criterion for vector autoregressive model selection. *Journal of time series analysis*, 14(3):271–279, 1993.

[9] M. Jereminov, A. Pandey, H. A. Song, B. Hooi, C. Faloutsos, and L. Pileggi. Linear load model for robust power system analysis. In *IEEE PES Innovative Smart Grid Technologies*, page (submitted). IEEE, 2017.

[10] P. Ji, D. Xiong, P. Wang, and J. Chen. A study on exponential smoothing model for load forecasting. In *Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific*, pages 1–4. IEEE, 2012.

[11] R. E. Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[12] J. Letchner, C. Re, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 246–257. IEEE, 2009.

[13] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[14] J. R. Martí, H. Ahmadi, and L. Bashualdo. Linear power-flow formulation based on a voltage-dependent load model. *IEEE Transactions on Power Delivery*, 28(3):1682–1690, 2013.

[15] Y. Matsubara and Y. Sakurai. Regime shifts in streams: Real-time forecasting of co-evolving time sequences. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1045–1054. ACM, 2016.

[16] A. Pandey, M. Jereminov, X. Li, G. Hug, and L. Pileggi. Aggregated load and generation equivalent circuit models with semi-empirical data fitting. In *Green Energy and Systems Conference (IGSEC), 2016 IEEE*, pages 1–6. IEEE, 2016.

[17] J. Park, Y. Park, and K. Lee. Composite modeling for adaptive short-term load forecasting. *IEEE Transactions on Power Systems*, 6(2):450–457, 1991.

[18] A. Selakov, D. Cvijetinović, L. Milović, S. Mellon, and D. Bekut. Hybrid pso–svm method for short-term load forecasting during periods with significant temperature variations in city of burbank. *Applied Soft Computing*, 16:80–88, 2014.

[19] H. A. Song, B. Hooi, M. Jereminov, A. Pandey, L. Pileggi, and C. Faloutsos. Powercast: Mining and forecasting power grid sequences. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2017.

[20] E. M. Stewart, A. Liao, and C. Roberts. Open pmu: A real world reference distribution micro-phasor measurement unit data set for research and application development. 10/2016 2016.

[21] G. Upton and I. Cook. *Understanding statistics.* Oxford University Press, 1996.

[22] P. R. Winters. Forecasting sales by exponentially weighted moving averages. *Management science*, 6(3):324–342, 1960.

[23] P. Zhang, X. Wu, X. Wang, and S. Bi. Short-term load forecasting based on big data technologies. *CSEE Journal of Power and Energy Systems*, 1(3):59–67, 2015.