Using Deep Generative Models to Boost Forecasting: A Phishing Prediction Case Study

Syed Hasan Amin Mahmood
Department of Electrical Engineering
Lahore University of Management Sciences
Lahore, Pakistan
syed.mahmood@lums.edu.pk

Ahmed Abbasi
Department of IT, Analytics, and Operations
University of Notre Dame
Notre Dame, IN, USA
aabbasi@nd.edu

***Abstract*—Time series predictions are important for various application domains. However, effective forecasting can be challenging in noisy contexts devoid of time series data encompassing stationarity, cyclicality, completeness, and non-sparseness. Cybersecurity is a good example of such context. In organizational security settings, predicting time series related to emerging attacks could enhance cyber threat intelligence, resulting in timely and actionable insights at the operational, tactical, and strategic levels. In order to explore this gap, we propose a deep generative model-based framework for time series forecasting in noisy data environments. The proposed framework incorporates a novel ensembling strategy where generative adversarial networks and recurrent variational autoencoders are leveraged in unison with base predictors for enhanced regularization of time series predictive models. The framework is extensible, supporting different model combinations and analytical or iterative model fusion strategies. Using a test bed encompassing 10 years of weekly phishing attack volume data from 5 organizations in the technology, financial services, and social networking sectors, we show that the framework can boost predictive power for various standard time series models. Additional results reveal that the framework outperforms generative data augmentation approaches designed to enrich the input time series data matrices. Collectively, our findings suggest that utilizing generative models in more robust end-to-end setup can improve prediction in cyber threat intelligence contexts, as well as related problems involving challenging time series data.**

***Index Terms*—Time series modeling, generative adversarial networks, variational autoencoders, phishing, cyber threat intelligence, cybersecurity, deep learning, predictive analytics.**

I. Introduction

Time series modeling is important for proactive forecasting and situational awareness in many contexts ranging from sales and finance to health policy, environmental planning, economics, and cybersecurity. However, whereas time series data in many contexts exhibits characteristics such as seasonality, cyclicality, completeness, stationarity, and non-sparseness, the stochastic processes that underpin time series data in certain application domains are often devoid of these properties [1]. In these domains, classic statistical and machine learning methods for time series modeling are often less effective. Cybersecurity is a good example of this; the quantity and severity of attacks experienced by an organization over time might look very different in terms of shape and structure relative to its primary performance metrics such as sales, expenses, inventory, growth, equity, etc.

In organizational security contexts, cyber threat intelligence (CTI) has emerged as an analytics-driven approach to developing timely and actionable insights about emerging threats and/or key actors [2], [3]. Predictive analytics applications of CTI include detection of hacker assets [4], [5], system and device vulnerabilities [6], [7], phishing threats [8], and susceptible users [9]. From a CTI perspective, the ability to forecast near-term threats — such as phishing attacks and intrusion detection attempts in the coming days and weeks — could complement this existing body of work by affording opportunities for proactive mitigation strategies at the operational, tactical, and strategic levels.

In order to explore this gap in time series modeling, we propose the use of generative models as a mechanism for enhanced forecasting in noisy, sparse data environments. Our proposed deep generative model (DGM) framework incorporates a novel ensembling strategy where generative adversarial networks (TimeGAN) and recurrent variational autoencoders (RVAE) are leveraged in unison with base predictors for enhanced regularization of time series predictive models. The framework, which is designed to boost performance for many/most baseline time series models, also allows fusion of multiple predictions through least-squares solution (LSS) or MergeNet, thereby supporting use of analytical or iterative prediction aggregation.

Using a test bed encompassing 10 years of weekly phishing attack volume data from 5 organizations in the technology, financial services, and social networking sectors, we show that the framework can improve prediction for various models such as ARIMA, linear and ridge regression, random forest, multilayer perceptrons (MLPs), and long short-term memory (LSTM) recurrent neural networks. Additional results show that the framework outperforms generative data augmentation approaches designed to enrich columns or rows of the input time series data matrix — suggesting that use of generative models in more robust end-to-end learning frameworks can further improve forecasting in challenging time series contexts. Our results have important implications for phishing prediction, future work examining proactive uses of predictive analytics for cyber threat intelligence, as well as research in broader application domains involving complex, noisy time series data.

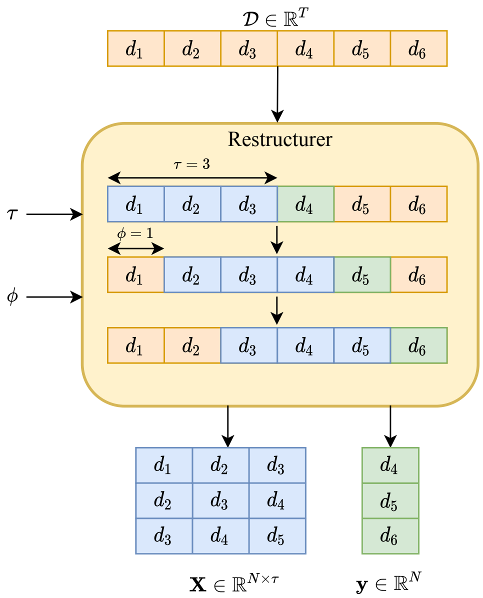

Fig. 1: Illustration of Restructurer block, used to restructure univariate time series of length T to obtain N lag features and target values, where $N = \frac{(T \quad)}{\quad}$.

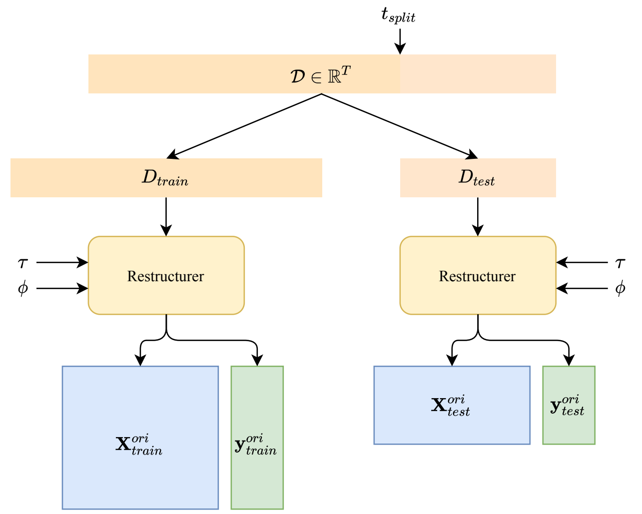

Fig. 2: Illustration of train-test split for time series data.

II. Related Work and Problem Formulation

Prior CTI work in the predictive analytics space has focused on detecting emerging threats and/or identifying threat actors. Previous threat detection studies include ones on identification of hacker assets [5], [10], [11], detecting vulnerable cyber-physical systems [12], uncovering attack vectors in malicious source code [13], identifying and categorizing phishing web-sites [14]–[16], detecting attack servers [17], and creating digital traces of attacks [18], [19]. Threat actor-related work has examined how to predict or detect key actors [20]–[22], identifying and understanding users most susceptible to attacks [23], and predicting vulnerable employees [9].

While significant research has focused on the detection problem in CTI contexts, work concerned with time series forecasting in the cybersecurity domain remains limited. Some prior studies used social media data and activity levels as a predictor for future distributed denial of service and other cyberattacks [24], [25]. The standard approaches used are methods such as ARIMA, regression methods, feature-based machine learning classifiers, and recurrent neural networks such as LSTMs [26]–[28]. The lack of stationarity or cyclicality, incompleteness, and sparseness, however, have posed challenges for effective forecasting in CTI contexts. Some of these challenges are also prevalent in health and policy domains, making solutions for more robust time series modeling in complex, noisy contexts an important research gap [1].

The time series forecasting problem can be formulated as a windowed training-testing split with temporally earlier data instances used to predict future outcomes. In addition to the windowed training and testing, restructurer blocks are used to generate "windowed" feature columns, for instance, attack volumes in each of the past   weeks being used as the features to forecast attack volume in the next week. Fig. 1 shows how a restructurer block can be used to construct such a windowed feature data matrix. Fig. 2 shows the restructurer block within the broader windowed time series training-testing split context.

To address gaps in time series modeling and forecasting in challenging contexts, we believe generative models provide one promising avenue. Generative models are concerned with approximating an underlying data distribution given access to a finite set of samples from it. The end goal is often to have instances or columns that are characteristically similar to the real ones. Deep generative models utilize deep learning towards attaining this goal. While multiple DGMs have been proposed including Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Deep Boltzmann Machines and Normalizing Flows, we limit our discussion to the former two, which are also the most popular ones. Through this selection, we remain concise in our analysis while gaining benefits associated with the said models' popularity (known efficacy on numerous problems, vast reach/impact, active area of research, etc.). Still, our proposed framework is generic and extensible, and can easily accommodate other models too.

There has been limited recent work on generative models for forecasting, with most of those studies using GANs. The two most common tactics have been data augmentation via instance generation and feature generation. Instance generation methods use GANs to demonstrate their efficacy for row-generation based augmentation [29]–[31]. For example, [31] utilized synthetic, generated training data applied to real test instances to illustrate usefulness of the augmentation-based instances. Other work has focused on use of data augmentation strategies to construct new columns of data, or generated multivariate time series features [32]–[34]. This approach entails simulating a prediction horizon and having a generator

(a) Instance Augmentation (b) Feature Augmentation (c) Generative Ensembling

Fig. 3: Three possible utilizations of generative models in time series contexts. Generative models have been used for instance augmentation (a) or feature augmentation (b). We propose a DGM generative ensembling framework (c) that emphasizes inclusion of ensembles of predictors trained on original data and one, as shown in (c), or more sets of generated data synthesized using deep generative models, while also leveraging some aspects of instance and feature augmentation.

attempt to construct realistic feature columns using actual data as input (as opposed to noise) or actual data in conjunction with noise in the case of conditional GANs [32], [33]. In many cases, these generated feature columns are coupled with the original input to improve predictive power [35], [36].

Fig. 3 highlights these two data augmentation use cases for generative models, namely use of GANs to create new instance rows (Fig. 3a) or new feature columns (Fig. 3b). Fig. 3c depicts the main intuition behind our proposed deep generative modeling framework. We build on the prior GAN data augmentation work by incorporating instance and feature level augmention in conjunction with ensembling strategies. Furthermore, we employ both GAN and recurrent variational autoencoder (RVAE) and show that use of GAN and RVAE in conjunction with base predictors can boost forecasting performance. Finally, our framework is extensible with respect to the choice of ensemble fusion strategies, allowing use of analytical and iterative methods such as LSS and MergeNet. Before describing our framework in greater detail, we provide important background on the two types of generative models employed in our DGM framework — GANs and VAEs.

A. Generative Adversarial Network

Generative Adversarial Networks estimate generative models through a process corresponding to a minimax two-player game. The setup involves simultaneously training two models with adversarial objectives: a generator G that captures a data distribution to generate real-like data and a discriminator D that distinguishes between real and generated data.

The generator $G(z;\theta_g)$ can be modelled as a differentiable function that maps input noise variable $z \in \mathcal{Z}$ to data space $\mathcal{X}$, i.e., $G: \mathcal{Z} \mapsto \mathcal{X}$. On the other hand, the discriminator $D(x;\theta_d)$ is a function that maps input from data space to the probability that the input data is real, i.e., $D: \mathcal{X} \mapsto [0,1]$. D is trained to maximize the probability of assigning correct labels to both real and generated data. G is simultaneously trained to minimize $\log(1 - D(G(z)))$, i.e., the probability of the discriminator correctly identifying generated samples as not real.

The zero-sum game can be framed as the following optimization problem, which in practice is implemented using an iterative, numerical approach:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}(x)} \log[D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Our framework, described later in Section III, uses GANs designed specifically for time series contexts.

B. Variational Autoencoder

Variational autoencoders have their roots in latent variable models. We first introduce a latent variable $z \in \mathcal{Z}$ that we believe somehow encodes meaningful information about data variable $x \in \mathcal{X}$. Given a set of observations $\mathcal{D}$, our objective is to select $p \in \mathcal{P}_{x,z}$ that best explains the observed data.

$$\mathcal{P}_{x,z} = \{p(x,z) \mid p(z) \in \mathcal{P}_z, p(x|z) \in \mathcal{P}_{x|z}\}$$

One way to do this is by minimizing the Kullback-Leibler (KL) divergence between the data distribution and the model's marginal distribution, which is equivalent to maximizing the marginal log-likelihood over $\mathcal{D}$.

$$\min_{p \in \mathcal{P}_{x,z}} D_{KL}(p_{data}(x) \| p(x)) = \max_{p \in \mathcal{P}_{x,z}} \sum_{x \in \mathcal{D}} \log p(x)$$
$$= \max_{p \in \mathcal{P}_{x,z}} \sum_{x \in \mathcal{D}} \log \int p(x,z)\, dz$$

Fig. 4: Illustration of how DGM, TimeGAN or RVAE for instance, is used to generate time series using training data.

The problem, however, is intractable for high dimensional z. An alternative, then, is to find a lower bound that is easier to optimize than maximizing the log-likelihood directly. We thus introduce a variational family Q of distributions that approximate the true posterior $p(z|x)$. We further assume a parametric setting where any distribution in the model family $P_{x;z}$ is parameterized by $\theta \in \Theta$ and distributions in variational family Q are parameterized by $\phi \in \Phi$.

$$
\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x; z)\, dz \\
&= \log \int \frac{q_\phi(z|x)}{q_\phi(z|x)} p_\theta(x; z)\, dz \\
&\geq \int q_\phi(z|x) \log \frac{p_\theta(x; z)}{q_\phi(z|x)}\, dz \\
&= \int q_\phi(z|x) \left(\log p_\theta(x|z) + \log \frac{p(z)}{q_\phi(z|x)} \right) dz \\
&= E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z))
\end{aligned} \tag{1}
$$

We can now learn a latent variable model by maximizing the evidence lower bound (ELBO) derived in (1), i.e., by performing the following optimization:

$$
\max_\theta \sum_{x \in D} \max_\phi E_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) || p(z))
$$

In variational autoencoders, the parameters θ and ϕ are learnt using neural networks. The variational posterior $q_\phi(z|x)$ is often called the encoder while the generative model $p_\theta(x|z)$ is referred to as the decoder. Having provided an overview of GANs and VAEs, in the subsequent section we describe how our proposed framework utilizes variations of these two types of generative models as part of the ensembling approach alluded to in Fig. 3.

III. Proposed Framework

As noted in the prior section and highlighted earlier in Fig. 3, our deep generative model framework is geared towards ensembling of base and generative models for time series forecasting. Fig. 4 summarizes the data generation pipeline. Having already discussed restructurer blocks (left side of the figure) and training matrix construction (right side of the figure), we focus on the center portion of Fig. 4 in this section. We specifically discuss the time series GAN and VAE models, and how these models' outputs are used with base predictors for fusion later using analytical or iterative aggregation techniques.

A. The Time Series Generative Modeling Process

As discussed in Section II, different generative models have unique properties. We can learn to generate data points by transforming noise (i.e., using GANs) or by learning a latent distribution to sample from (i.e., through VAEs). VAEs are distinct in that they can also be used to generate time series more closely aligned with the original time series by inputting the original data to the encoder. As we later show empirically, and with visual illustrations, these differences in the underlying intuitions and mechanisms for GANs versus VAEs are a crucial driver for the success of the proposed DGM ensembling framework in our phishing attack CTI context.

1) GANs for Time Series Modeling: For effective time series generation, recent advancements in GANs for time series data have been proposed [37]. For instance, TimeGAN combines the flexibility of the unsupervised GAN framework with control over conditional temporal dynamics afforded by supervised autoregressive models. An embedding, and associated recovery network, is introduced to provide reversible mapping between features and latent representations while reducing the high dimensionality of the adversarial learning space, capitalizing on the fact that temporal dynamics are often driven by factors aligned with lower dimensional variation.

Relying solely on the discriminator's binary adversarial feedback, as captured by the unsupervised Loss L_S described earlier in II-A, may not be sufficient incentive for the generator to capture temporal dynamics in the data. Effectual embedding and recovery functions are therefore learnt using reconstruction loss L_R in TimeGAN. The generator is trained to first generate latent representation using noise coupled with a temporally earlier latent representation, and an additional supervised loss L_S is introduced to further discipline learning. Let $_e; \; _r; \; _g; \; _d$ denote parameters of the embedding, recovery, generator and discriminator networks, respectively, and and be two hyperparameters. The optimization procedure is as follows:

$$\min_{e;\,r}(\; L_S + L_R)$$

$$\min_{g}(\; L_S + \max_{d} L_U)$$

The losses are more formally described below. $x, \hat{x}$ denote actual and recovered features, y and $\hat{y}$ correspond to whether a sample is real or synthetic, h denotes latent representation, z denotes noise, and g is the generator function implemented using a recurrent neural network.

$$L_R = E_{x_{1:T} \; p} \sum_t k x_t \; \hat{x}_t k_2$$

$$L_U = E_{x_{1:T} \; p} \sum_t \log y_t + E_{x_{1:T} \; \hat{p}} \sum_t \log(1 \; \hat{y}_t)$$

$$L_S = E_{x_{1:T} \; p} \sum_t k h_t \; g(h_{t \; 1}; z_t) k_2$$

2) Recurrent Variational Autoencoder: To adapt VAE, as outlined in II-B, for the time series context, a recurrent version of VAE called RVAE is employed using principles from [38]. Parameters for the encoder and decoder networks are learnt using recurrent neural networks, LSTM units more specifically. In essence, RVAE can be thought of as encompassing a VAE at every time step, and the objective function in (1) is updated as such to obtain the following optimization problem:

$$\max \sum_{x \, 2 D} \max E_{q \; (z \; _T j x \; _T)} \sum_{t=1}^{T} h \log p \; (x_t j z \; _t; x_{<t}) \; D_{KL} \; q \; (z_t j x \; _t; z_{<t}) jj p \; (z_t j x_{<t}; z_{<t}) \; i$$

It is worth noting that aside from using GANs and VAEs in unison to leverage their varying strengths via ensembling, there has been prior work on combining VAE and GAN models in a single, parsimonious model often referred to as VAE-GAN. The idea is to collapse the VAE decoder and the GAN generator into one (and jointly training them) by letting them share parameters [39]. As we later show, this approach does not work as well for our time series context — though future work could explore this idea in greater depth.

B. Ensembling Strategies

Once an ensemble of models run across real and/or generated data has output predictions, the next step in the framework is to fuse or aggregate those predictions. In addition to simple averaging, analytical and iterative methods can have varying advantages and disadvantages for how ensembling or fusion of predictions takes place. Accordingly, we discuss least-squares solution (LSS) and MergeNet-based strategies.

Given forecasts by m models of length n each, we want to find the weighted average of all forecasts that most closely resembles the target time series, as expressed in (2). $\hat{y}_0$ = 1 (vector of ones) always, and is introduced to include a bias term. The weights are decided at training time. All predicted ($\hat{y}_i$) and target (y) values discussed here thus refer to those obtained from training data, although this is not explicitly stated in the ensuing notation, for convenience and readability.

$$w_0\hat{y}_0 + w_1\hat{y}_1 + w_2\hat{y}_2 + \quad + w_m\hat{y}_m \quad y$$
$$\hat{Y}w \quad y$$
$$\hat{y} \quad y \qquad (2)$$

where w_i 2 R, $w = [w_0; w_1; :::; w_m]^|$ 2 R^{m+1}, $\hat{y}_i$ 2 R^n, $\hat{Y} = [\hat{y}_0; \hat{y}_1; :::; \hat{y}_m]$ 2 $R^{n \; (m+1)}$ and y 2 R^n.

1) Analytical Method (LSS): We more formally express what close resemblance to target time series, as indicated in (2), entails through our objective in (3).

$$w \; = \arg\min_{w} k y \; \hat{y} k \qquad (3)$$

While the norm in (3) can be any valid norm, we select it to be l_2-norm to conveniently obtain optimal analytical solution. We can see that the optimal $\hat{y}$ would be the orthogonal projection of y onto $(m+1)$-dimensional space spanned by columns of $\hat{Y}$. With optimal weights w, the residual error $(y \; \hat{Y}w \;)$ would consequently be orthogonal to every column of $\hat{Y}$, i.e., $(y \; \hat{Y}w \;)\hat{Y} = 0$, leading to

$$w \; = (\hat{Y}^| \hat{Y}) \; ^1 \hat{Y}^| y$$

The same is formally derived using matrix calculus in Eqs. (4)-(10). We observe that this is really the least-squares solution (LSS), which can be very efficiently computed using modern algorithms. We assume $\hat{Y}$ to be full rank, for $\hat{Y}^| \hat{Y}$ to be invertible. We further assume $n > m$, for $\hat{Y}$ to be left invertible, which is typically the case — the length of time series is often much greater than the number of models. If this is not the case, it can be shown that $w \; = \hat{Y}^| (\hat{Y} \hat{Y}^|) \; ^1 y$ is an optimal solution instead.

We thus use $w \; = \hat{Y}^y y$, where ^{y} represents the (left or right, as appropriate) pseudo-inverse, as the analytical solution, and refer to it as LSS without loss of generality. It is important to note that while we use a linear combination of predictions, this scheme is applicable much more widely, for example we can also perform polynomial combination with degree 2 using $\hat{Y}^0 = [\hat{Y}; \; \hat{Y} \; \hat{Y}]$ instead of $\hat{Y}$, where denotes the Hadamard product.

The major advantage of such an analytical, LSS-based fusion strategy is that it provides the optimal weights for predictions on the training data. That is, the root mean square error (RMSE) between $\hat{y}_{train}$ and y_{train} is guaranteed to be less than or equal to the RMSE obtained using any other weights. Further, it can be computed efficiently if n or m are not extremely large. Conversely, as is often the case in data mining, over-fitting is a possible challenge — the optimal weights found using training data may not translate into ideal weights for prediction on test data (which can be the case when dealing with data that has limited stationarity). Further, $\hat{Y}^y$ can be very expensive to compute for very high dimensional $\hat{Y}$, since inverse computation is typically a $O(n^3)$ operation.

(a) Original (b) RVAE (c) TimeGAN (d) VAE-GAN

Fig. 5: Training data, comprising original time series and those generated by RVAE, TimeGAN and VAE-GAN.

$$w = \operatorname*{argmin}_{w} \|y - \hat{Y}w\|_2^2 \tag{4}$$

$$\nabla_w \|y - \hat{Y}w\|_2^2 = \nabla_w (y - \hat{Y}w)^\intercal (y - \hat{Y}w) \tag{5}$$

$$= \nabla_w (y^\intercal y - 2w^\intercal \hat{Y}^\intercal y + w^\intercal \hat{Y}^\intercal \hat{Y} w) \tag{6}$$

$$= -2\hat{Y}^\intercal y + 2\hat{Y}^\intercal \hat{Y} w \tag{7}$$

$$\nabla_w \|y - \hat{Y}w\|_2^2 = 0 \tag{8}$$

$$-2\hat{Y}^\intercal y + 2\hat{Y}^\intercal \hat{Y} w = 0 \tag{9}$$

$$w = (\hat{Y}^\intercal \hat{Y})^{-1} \hat{Y}^\intercal y \tag{10}$$

2) Iterative Method (MergeNet): Iterative methods offer complementary pros and cons to analytical techniques such as LSS. We use gradient descent for iterative optimization, which can be implemented as a simple neural network. First, all chunks of size c are extracted, where $c \leq n$, from $\hat{y}_i$ and y, keeping time order intact (because the k^{th} chunk of each $\hat{y}_i$, for example, would be weighted to closely resemble the k^{th} chunk of y). We aim to minimize error in (11); for convenience and readability, y refers to a associated chunk of y and $\hat{Y}$ contains the particular chunk of every $\hat{y}_i$. Weights are updated iteratively using (13). η is the learning rate.

$$E = \|y - \hat{Y}w\|_2^2 \tag{11}$$

$$w = w - \eta \nabla_w E \tag{12}$$

$$w = w - \eta(-2\hat{Y}^\intercal y + 2\hat{Y}^\intercal \hat{Y} w) \quad * (7) \tag{13}$$

This can be implemented as a single neuron neural network with $\hat{y}_i$s as inputs (number of data points equal to number of chunks) with mean square error loss. Through this equivalence, we can perform prediction aggregation through more complex neural networks and choose other loss functions as needed.

In contrast to LSS, iterative methods such as MergeNet can be efficient when n and m are larger. By using $c < n$, and through iteration, the scheme is more likely to generalize to test data and is capable of modeling many complex relations. However, it also faces many of the classic neural network limitations (run times, hyperparameters, etc.). Fig. 9 illustrates this contrast between LSS and MergeNet performance, where the latter is more effective when the train-test distributions are less similar. We include both in our framework to allow the DGM ensemble framework to be extensible for a myriad of noisy, complex time series contexts, where one ensembling strategy may be more suitable than the other.

IV. Experiments

We evaluated our proposed DGM framework in a relevant CTI context — predicting phishing attack volume. Our time series data test bed was taken from the PhishMonger project repository [19]. The PhishMonger time series repository includes over 1.5 million unique verified phishing attacks related to over 100 targeted organizations' websites across a 10 year period from 2006 through 2015. We focus on five of the most highly targeted organizations in the social media, financial services, and internet sectors: AOL, Bank of America, eBay, Facebook, and Wells Fargo. For each organization, weekly phishing attack volume data is modeled as a time series, with the first seven years used for training and the last three for testing.

In order to provide a small visual illustration of what the actual and generated data for these types of time series may look like, Fig. 5 provides an example. The figure shows training data for AOL and Facebook. These two were chosen for this particular figure since they represent different industry sectors. As noted, the TimeGAN generates samples from noise that are not necessarily paired with equivalent real world tuples. Hence, the structure of the generated data only captures certain cyclical patterns and also appears noisy. In contrast, the RVAE does not generate data simply by explicitly sampling the latent distribution ($\mathcal{N}(0;1)$ in our case). Instead, the original data is input to the encoder, and the resulting decoder output forms the RVAE generated data. As shown in the figure, this allows the VAE samples to be more closely aligned with the real data in terms of distribution and structure, within a Gaussian distributional framework. Further, we also include

Fig. 6: Boxplots for original and generated training data.

VAE-GANs to show that the RVAE might be more effective than a simple end-to-end hybridization of VAEs and GANs.

Fig. 6 provides a more aggregated visual view of attack distributions for all ve organizations in our data set, excluding the outlier spikes in the time series. From the gure, we can see that the shape/structure alignments observed earlier in Fig. 5 are also prevalent at the distribution level. Namely, RVAE produces distributions that are more in sync with the original data, whereas TimeGAN's output has more limited diversity and seems less aligned with the original data. Interestingly, it might seem that simply using the RVAE in conjunction with the original data would be better than also including TimeGAN in the mix. However, the inclusion of TimeGAN adds robustness in results for many organizations, as we show in the subsequent results.

A. Impact of DGM Framework on Performance

Table I and Table II show the overall results for the DGM framework when boosting performance for six different prediction methods (ARIMA, linear and ridge regression, random forest, MLP, and LSTM) across the ve organizational test beds. In order to allow easier readability, we have aggregated the 3-tuple, that is, three dimensional predictor-testbed-DGM results, into two 2-tuples by predictor and organization. Looking at the mean MAE and RMSE results by predictor (Table I), we can see that the use of DGM boosts performance for ve of the six base predictors examined. The one exception is random forest, where the base predictor not utilizing the DGM framework outperformed the DGM-boosted predictions. For methods such as ARIMA and LSTM, DGM reduced prediction error markedly, by 10 to 15 percentage points. Similarly, looking at the organization-level aggregated results (Table II), we can see that DGM improves performance across all ve organizational test beds. The improvements are most pronounced on the AOL and Bank of America phishing attack time series, where RMSE is reduced by 5 to 50 percent. The results on random forest might be explained by the inclusion of an inherently ensemble-driven base predictor into another ensemble, although further work is needed to examine the underlying causes. Nevertheless, the results across all ve organizations and ve out of six base predictors suggest that the DGM framework concepts of ensemble-oriented usage of generative models might have potential to boost time series prediction in noisy, complex environments such as attack prediction for CTI.

TABLE I: Impact of DGM aggregated to base predictors.

	Without DGM		With DGM	
	MAE	RMSE	MAE	RMSE
ARIMA	61.8	92.5	53.4	82.9
Linear Regression	50.9	80.5	48.8	78.6
Ridge Regression	51.0	80.6	48.6	78.4
Random Forest	46.7	78.3	48.7	82.0
MLP	49.5	79.8	47.9	78.5
LSTM	65.8	95.2	48.3	80.5

TABLE II: Impact of DGM aggregated to organizations.

	Without DGM		With DGM	
	MAE	RMSE	MAE	RMSE
AOL	77.0	95.9	73.3	91.5
BofA	36.0	37.8	18.3	20.9
eBay	120.5	236.1	119.8	235.7
Facebook	26.3	33.0	21.8	31.7
Wells Fargo	12.5	20.1	12.1	19.7

The results described in the prior paragraph used a xed prediction horizon. In order to examine the impact of prediction horizon on effectiveness of predictors boosted by DGM (relative to no use of DGM), we plotted MAE (y-axis) for different horizon sizes in weeks (see Fig. 7) on the AOL, Bank of America, and eBay data. As shown in the gure, on all three test beds depicted, DGM reduces forecasting error across an array of prediction horizons. The results are most pronounced when the horizons are shortest. That is, for 20-30 weeks or less. However, this represents a very large prediction horizon range. The results suggest that the DGM framework might be of practical importance for near-term (e.g., 1-4 week) as well as mid-term forecasting as far as a few months out, since it can better anticipate time series structure than the base predictors.

B. Ablation Analysis

Ablation analysis was performed to shed light on the impact of different components of the proposed DGM framework on overall performance. In particular, three types of ablation analyses were performed. In the rst, we explored the impact of using the TimeGAN and RVAE in conjunction with the base predictors, as opposed to using different subset combinations. In the second, we examined the effectiveness of fusing

(a) AOL (b) BofA (c) eBay

Fig. 7: Prediction horizon analysis, illustrating how performance of models on held-out test set changes with expanding prediction horizon (in weeks) with and without deep generative models.

(a) Base Predictor (BP) (b) TimeGAN (c) RVAE (d) BP+TimeGAN+RVAE

Fig. 8: Actual held-out test data and predictions by multiple predictors and their ensemble when using MLP as a base predictor.

with LSS versus MergeNet. Lastly, we compared our DGM framework with alternative data augmentation-based uses for generative models that have been proposed in recent years. Details are as follows.

1) Impact of TimeGAN, RVAE, and Base Combinations: In the main results, our DGM approach ensembled the base predictor trained on original time series with predictors trained using TimeGAN-generated and RVAE-generated data. In order to examine the impact of different combinations, we explored all seven combinations (e.g., Base Predictor + TimeGAN, Base Predictor + RVAE, Base Predictor + RVAE, all three, and each predictor alone). Table III and Fig. 8 show the results. As depicted in the table, for Facebook attack data, ensembling the base predictor, TimeGAN, and RVAE together resulted in the best performance for four of the six base predictors. Fig. 8 illustrates how the base predictor was enhanced by the RVAE and TimeGAN. This example illustrates the effect of using a MLP base predictor on AOL and Facebook attack data. As depicted in the gure, the base predictors tended to overreact to the most recent time series movements, resulting in a relatively more turbulent set of forecasts. Conversely, the TimeGAN produced a much smoother forecast and the RVAE was also somewhat smoother than the base predictor. Collectively, the ensemble with all three models fused together offered the smoothest and most accurate forecasts. The results lend credence to the design of the proposed DGM ensembling framework.

TABLE III: Impact of different combinations of predictions, based on MAE. Underline depicts better than Base Predictor. Bold depicts best. BP, TG and RV denote Base Predictor, TimeGAN and RVAE respectively.

	BP	TG	RV	TG, RV	BP, TG	BP, RV	BP, TG, RV
ARIMA	29.1	36.3	26.3	28.2	24.8	30.7	23.6
Linear Regression	19.9	21.1	21.9	24.8	19.4	19.4	19.2
Ridge Regression	19.9	21.0	21.9	24.8	19.3	19.4	19.2
Random Forest	22.4	35.9	26.8	30.4	22.7	23.3	23.4
MLP	26.1	22.4	28.4	29.4	22.2	22.4	22.1
LSTM	40.4	39.8	40.6	25.9	23.0	22.9	23.0

(a) (b)

Fig. 9: Comparison of LSS and MergeNet for fusing ensemble predictions on Facebook data using linear regression.

2) Impact of LSS Versus MergeNet on Performance: As noted earlier, the analytical LSS approach and the iterative MergeNet method (relying on a neural network) have varying pros and cons when used for fusion. In order to explore these trade-offs, we compared the results for our DGM framework when using LSS versus MergeNet to fuse ensemble predictions (Table IV). Interestingly, while LSS outperformed MergeNet on the ARIMA and LSTM methods, MergeNet was slightly better for the linear and ridge regressions and MLP. Fig. 9 depicts the RMSE results for LSS and MergeNet on the Facebook data using linear regression as the base predictor. The results include two intentional data manipulations (x-axes on the two charts) applied to the training data: (i) the original training data was scaled, and (ii) bias was introduced to the original standardized training data. The purpose was to see how the inclusion of scale distortion and bias in the training data impacts performance as the training data further deviates from test cases. As shown in the gure, LSS outperforms MergeNet when the training data distributions are relatively unperturbed (i.e, the scale is around 1 and bias around 0), while MergeNet is better when scale or bias are fairly high. These results suggest that whereas LSS performs better on the more stationary, less noisy time series, MergeNet has the potential to offer robust learning in noisier environments.

TABLE IV: Impact of prediction aggregation methods, based on MAE. Underline depicts better than base predictor, i.e., without prediction aggregation. Bold depicts best.

	Base Predictor	Simple Average	MergeNet	LSS
ARIMA	29.0	30.4	26.3	23.6
Linear Regression	19.9	20.3	19.0	19.2
Ridge Regression	19.9	20.3	19.0	19.2
Random Forest	22.4	25.5	24.4	23.4
MLP	26.1	23.0	21.9	22.1
LSTM	40.4	40.3	23.0	23.0

3) Comparing DGM Versus Instance and Feature Data Augmentation: DGM presents a slight departure from the standard data augmentation use cases associated with many generative models. Accordingly, we ran experiments to show that using generative models in novel regularization configurations — such as ensembling in our DGM framework — is more effective than routine instance/feature augmentation that creates more rows or columns of data. The results appear in Table V. The ensembling approach employed in DGM improves results on all ve organizational test beds — by 20 to 50 percent on four of the ve data sets. These results further underscore the ef cacy of the DGM framework as a viable alternative to traditional generative model-based data augmentation techniques.

TABLE V: Impact of generated data utilization methods, based on MAE. All approaches use GANs for data generation. Underline depicts better than base predictor. Bold depicts best.

	Base Predictor	Instance Augmentation	Feature Augmentation	DGM Ensembling
AOL	87.9	89.9	87.5	72.3
BofA	60.1	60.1	60.1	26.0
eBay	119.8	121.4	118.3	117.0
Facebook	40.4	40.3	40.4	23.0
Wells Fargo	12.9	15.6	16.1	12.5

V. Conclusion and Future Work

In this work, we proposed a novel deep generative model (DGM) framework for enhanced time series forecasting in contexts involving noisy, complex data. We focused our analysis on an important problem with potential for proactive CTI — forecasting website phishing attack volume for organizations in various industry sectors. Our results suggest that fusing predictions based on original data and that generated by GANs and RVAEs has the potential to boost forecasting power. Ablation analysis revealed that the use of GANs and RVAEs offers complementary bene ts. Further, while LSS worked very well on time series where the testing data patterns were closely aligned with the training data, MergeNet provided bene ts on noisier time series data. Finally, we also show that DGM offers more robust bene ts than standard data augmentation-based regularization approaches using generative models.

The results have important practical implications for CTI. Forecasting of phishing attacks has received limited attention in the literature. As shown earlier in Fig. 7, DGM enables better predictive power over longer prediction horizons — as far out as four to six months. By allowing identi cation of emerging threat levels at varying time intervals, cybersecurity managers can anticipate threat levels/volumes and plan accordingly at the operational, tactical, and strategic levels. Collectively, the results suggest that extensible frameworks such as DGM are well-suited to adapting to the distinct aforementioned characteristics of noisy time series (nonstationarity, lack of cyclicality, sparesness, incompleteness, and so on). By boosting many different base predictors on time series data from ve different organizations' phishing attacks, with two complementary ensemble fusion options, the DGM framework affords exciting future possibilities. For instance, ensembling could be replaced with an end-to-end learning mechanism involving more advanced fusion approaches and cross-pollination between the different ensemble members. While VAE-GANs did not work well here, such architectures may provide the building blocks and intuitions for more powerful hybridization strategies. Generative models clearly have exciting potential for time series forecasting. We believe this study constitutes an important step that other work can build upon.

REFERENCES

[1] M. Tadayon and Y. Iwashita, "Comprehensive analysis of time series forecasting using neural networks," 2020.

[2] S. Samtani, M. Abate, V. Benjamin, and W. Li, "Cybersecurity as an industry: A cyber threat intelligence perspective," *The Palgrave Handbook of International Cybercrime and Cyberdeviance*, pp. 135–154, 2020.

[3] R. Williams, S. Samtani, M. Patton, and H. Chen, "Incremental hacker forum exploit collection and classification for proactive cyber threat intelligence: An exploratory study," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2018, pp. 94–99.

[4] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *Journal of Management Information Systems*, vol. 34, no. 4, pp. 1023–1053, 2017.

[5] V. Benjamin, W. Li, T. Holt, and H. Chen, "Exploring threats and vulnerabilities in hacker web: Forums, irc and carding shops," in *2015 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2015, pp. 85–90.

[6] R. Williams, E. McMahon, S. Samtani, M. Patton, and H. Chen, "Identifying vulnerabilities of consumer internet of things (iot) devices: A scalable approach," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 179–181.

[7] S. Samtani, S. Yu, H. Zhu, M. Patton, and H. Chen, "Identifying scada vulnerabilities using passive and active vulnerability assessment techniques," in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2016, pp. 25–30.

[8] A. Abbasi, F. M. Zahedi, D. Zeng, Y. Chen, H. Chen, and J. F. Nunamaker Jr, "Enhancing predictive analytics for anti-phishing by exploiting website genre information," *Journal of MIS*, vol. 31, no. 4, pp. 109–157, 2015.

[9] A. Abbasi, D. Dobolyi, A. Vance, and F. M. Zahedi, "The phishing funnel model: A design artifact to predict user susceptibility to phishing attacks," *Information Systems Research*, pp. 1–25, 2021.

[10] S. Samtani, R. Chinn, and H. Chen, "Exploring hacker assets in underground forums," in *2015 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2015, pp. 31–36.

[11] S. Samtani, K. Chinn, C. Larson, and H. Chen, "Azsecure hacker assets portal: Cyber threat intelligence and malware analysis," in *2016 IEEE conference on intelligence and security informatics (ISI)*. Ieee, 2016, pp. 19–24.

[12] S. Samtani, S. Yu, H. Zhu, M. Patton, J. Matherly, and H. Chen, "Identifying scada systems and their vulnerabilities on the internet of things: A text-mining approach," *IEEE Intelligent Systems*, vol. 33, no. 2, pp. 63–73, 2018.

[13] V. A. Benjamin and H. Chen, "Machine learning for attack vector identification in malicious source code," in *2013 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2013, pp. 21–23.

[14] A. Abbasi, F. M. Zahedi, and S. Kaza, "Detecting fake medical web sites using recursive trust labeling," *ACM Transactions on Information Systems (TOIS)*, vol. 30, no. 4, pp. 1–36, 2012.

[15] A. Abbasi and H. Chen, "A comparison of tools for detecting fake websites," *Computer*, vol. 42, no. 10, pp. 78–86, 2009.

[16] A. Abbasi, Z. Zhang, D. Zimbra, H. Chen, and J. F. Nunamaker Jr, "Detecting fake websites: the contribution of statistical learning theory," *Mis Quarterly*, pp. 435–461, 2010.

[17] J. Koepke, S. Kaza, and A. Abbasi, "Exploratory experiments to identify fake websites by using features from the network stack," in *2012 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2012, pp. 126–128.

[18] V. Benjamin, J. S. Valacich, and H. Chen, "Dice-e: A framework for conducting darknet identification, collection, evaluation with ethics." *MIS Quarterly*, vol. 43, no. 1, 2019.

[19] D. G. Dobolyi and A. Abbasi, "Phishmonger: A free and open source public archive of real-world phishing websites," in *2016 IEEE Conference on Intelligence and Security Informatics*. IEEE, 2016, pp. 31–36.

[20] V. Benjamin and H. Chen, "Securing cyberspace: Identifying key actors in hacker communities," in *2012 IEEE International Conference on Intelligence and Security Informatics*. IEEE, 2012, pp. 24–29.

[21] V. Benjamin, B. Zhang, J. F. Nunamaker Jr, and H. Chen, "Examining hacker participation length in cybercriminal internet-relay-chat communities," *Journal of Management Information Systems*, vol. 33, no. 2, pp. 482–510, 2016.

[22] V. Benjamin and H. Chen, "Time-to-event modeling for predicting hacker irc community participant trajectory," in *2014 IEEE Joint Intelligence and Security Informatics Conference*. IEEE, 2014, pp. 25–32.

[23] A. Abbasi, F. M. Zahedi, and Y. Chen, "Phishing susceptibility: The good, the bad, and the ugly," in *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2016, pp. 169–174.

[24] A. Okutan, G. Werner, S. J. Yang, and K. McConky, "Forecasting cyberattacks with incomplete, imbalanced, and insignificant data," *Cybersecurity*, vol. 1, no. 1, pp. 1–15, Dec 2018.

[25] Z. Wang and Y. Zhang, "Ddos event forecasting using twitter data," in *Proc. of the IJCAI*, 2017, pp. 4151–4157.

[26] I. Perry, L. Li, and A. Okutan, "Differentiating and predicting cyberattack behaviors using lstm," in *IEEE Conf, on Dependable and Secure Computing*, 2018, pp. 1–8.

[27] G. Werner, S. Yang, and K. McConky, "Leveraging intra-day temporal variations to predict daily cyberattack activity," in *IEEE Intl. Conf. Intelligence and Security Informatics*, 2018, pp. 58–63.

[28] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *NIPS Time Series Workshop*, 12 2016.

[29] F. Wang, Z. Zhang, C. Liu, Y. Yu, S. Pang, N. Duic, M. Shafie-khah, and J. Catalão, "Generative adversarial networks and convolutional neural networks based weather classification model for day ahead short-term photovoltaic power forecasting," *Energy Conversion and Management*, vol. 181, pp. 443–462, 02 2019.

[30] C. Tian, C. Li, G. Zhang, and Y. Lv, "Data driven parallel prediction of building energy consumption using generative adversarial nets," *Energy and Buildings*, vol. 186, pp. 230 – 243, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778818322965

[31] G. Ramponi, P. Protopapas, M. Brambilla, and R. Janssen, "T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling," 2018.

[32] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Computer Science*, vol. 147, pp. 400 – 406, 2019, 2018 International Conference on Identification, Information and Knowledge in the Internet of Things. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050919302789

[33] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic forecasting of sensory data with generative adversarial networks – forgan," *IEEE Access*, vol. 7, pp. 63 868–63 880, 2019.

[34] A. Koochali, A. Dengel, and S. Ahmed, "If you like it, gan it. probabilistic multivariate times series forecast with gan," 2020.

[35] S. Liu and M. Motani, "Long-range prediction of vital signs using generative boosting via lstm networks," 2019.

[36] N. D. Truong, L. Kuhlmann, M. R. Bonyadi, D. Querlioz, L. Zhou, and O. Kavehei, "Epileptic seizure forecasting with generative adversarial networks," *IEEE Access*, vol. 7, pp. 143 999–144 009, 2019.

[37] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 5508–5518.

[38] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 2980–2988.

[39] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 1558–1566.