6 Supplementary Material

6.1 Related Work

6.1.1 Generating synthetic data using known invariances

Making deep learning models robust when they are trained on limited data is critical. One approach is to encode known invariances into the model directly [18]. For example a classifier trained to detect cat images should be invariant to rotated, cropped or blurred images of cat. However, it is non-trivial to encode all invariances directly into the model. A simpler way is to encode those invariances in the training data by generating additional data using transformations. Traditional DA techniques used for this purpose employ a limited set of invariances that are static, known a priori, and easy to implement [13, 43, 40, 19]. Although considered best practices for a long time [30], this class of methods are shown to have limitations such as incapability of exploring a large invariance space [1] and requiring domain knowledge to design augmentation schemes [18].

6.1.2 Generating synthetic data by learning invariances

Recently a lot of work has been done in vision domain to infer invariances directly from training data [28, 9, 10, 1, 18] using generative models like GANs/VAEs that have been successfully adapted to TS domain [8, 39, 14, 34, 15], to generate synthetic TS samples that show effectiveness primarily on the task of TS classification. [39] supplements a GAN based generative model with a supervised autoregressive objective in the latent space and shows that it leads to more realistic TS that improve performance on a classification task. [8] use a GAN based approach and learn a modal-agnostic generative framework by jointly training a classifier in the latent space. Although the intuition behind GANs is quite elegant as exhibited in these papers, they are difficult to train and often experience mode collapse (which prevents from generalizing), vanishing gradients, and/or unstable updates. In [11] authors use a deterministic approach and train a denoising auto encoder [36] to generate financial TS data. Although impressive, there is no evidence that these complex GAN-based or compression-based auto encoding methods would work equally well on anomalous TS where its much harder to learn to generate sparse signals like point anomalies and level-shifts, in the synthetic data. In [6, 7] authors show anomalous TS can synthetically be generated by sampling from a learned latent space. Although impressive, there are two key limitations- (a) they still use a limited set of known invariances (eg: jittering, scaling, permutation) to handle class imbalance to train a VAE; (b) they pose AD task as a classification task to alleviate the problem of dealing with temporal labels which limits this approach to be of use in a traditional TS AD setting [37].

Conditional Generative Models. Explicitly conditioning the generative models on class labels has shown to learn a sharper and class-dependent data distribution. This idea has been adopted to modify both VAEs [32] and GANs [31, 27, 14] for TS generation. In [31] authors train two Wasserstein GANs [2] in a sequential manner where the spectograms generated from first WGAN are used to condition the second WGAN, which is trained to generate synthetic TS data. In [27] authors apply conditioning on timestamp information to handle irregular sampling while [14] applies conditioning on class labels in a temporal fashion to both the generator and discriminator to learn a class-dependent generative model.

VAEs for data imputation. Another interesting line of work involving VAEs for TS generation is data imputation. In [26] authors show how VAEs can be used to fill missing data in a high-dimensional heterogenous setting which was extend to TS by [16] by building a sequential latent variable model and using a prior that exploits temporal dynamics in latent space. [42] is another recent work where VAEs are proven to be useful for imputing missing TS data.

Inspired by these recent successes of VAEs we show how `C-GATS` adopts a factorized training procedure and uses ideas from data imputation to train an unconditional VAE on just the normal samples that acts as a foundation model [4] which is then used to fine-tune a conditional generative model that learns to generate synthetic anomalous TS with labels and addresses the limitations of previous approaches.

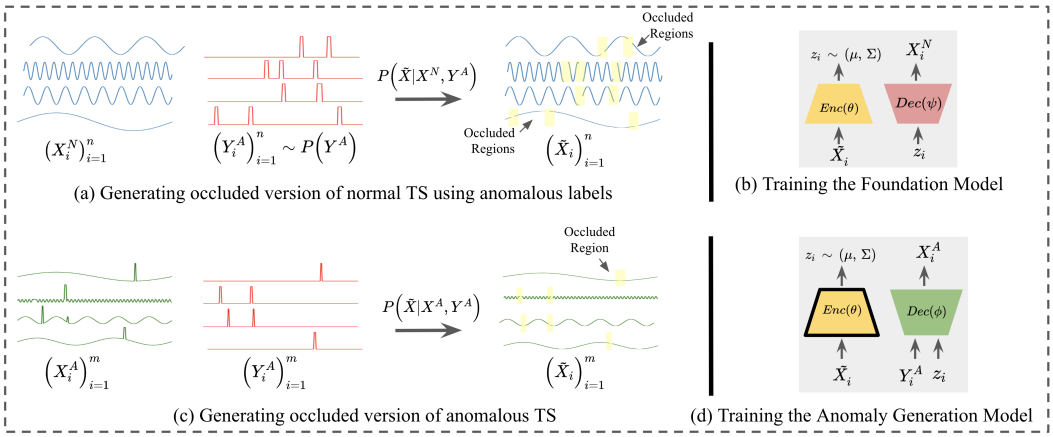

Figure 4: C-GATS Architecture. (a) Step 1: *sample occlusion model* is used to obtain $\tilde{X}$ from normal TS; (b) Step 2: Paired samples $(\tilde{X}, X^N)$ are then used to train the foundation model θ; (c) Step 3: *sample occlusion model* is used to obtain $\tilde{X}$ from anomalous TS; (d) Step 4: Anomaly generation model ϕ is trained in a conditional manner using $(\tilde{X}_i, X_i^A, Y_i^A)_{i=1}^m$. Solid lines around θ in (d) denote frozen weights.

6.2 Dataset Description

Synthetic Sines– We simulate univariate sinusoidal sequences of different frequencies η and phases θ, providing continuous-valued periodic samples each of fixed length, $T = 240$ timestamps; $x_i(t) = sin(2\pi\eta t + \theta)$, where $\eta \sim \mathcal{U}[0, 1]$ and $\theta \sim \mathcal{U}[-\pi, \pi]$. We simulate a total of 64000 TS. Then we insert point anomalies in the data. We sample 3 attributes randomly at uniform– (1) whether to insert an anomaly or not $s \sim \{1, 0\}$; (2) the position where to insert the anomaly, $p \sim range(240)$; (3) how long will the anomaly be, $l \sim range(10)$; (4) direction of anomaly, i.e, either positive of negative spike $d \sim \{+1, -1\}$. Thus, for sequence x_i in simulated data, we corrupt it by inserting anomaly using the process- $x_i^A = f(x_i; s, p, l, d)$.
KPI dataset[1]– A univariate TS dataset consisting of KPI curves from different internet companies in 1 minute interval. We use a sliding window of 240 to downsample the long TS in the data and obtain a fixed size datasets of 30000 TS each of length 240 timesteps.
NAB[2]– A public anomaly detection benchmark dataset [23] that contain streaming data from different domains. We select 4 different datasets from this benchmark- **(a) RealTweets**, **(b) RealTraffic**, **(c) RealAWSCloudWatch**, **(d) ArtificialWithAnomaly**. Each dataset was resampled to sequences of fixed length. Statistics of each dataset are provided in Table 4.

Table 4: Statistics of datasets

Dataset	Total Series	Total Points	Anomaly Points
Synthetic Sines	64000	15360000	171071
KPI	30000	7200000	328527
RealTweets	4000	960000	159740
RealTraffic	1200	288000	14364
RealAWSCloudWatch	121	29040	3070
ArtificialWithAnomaly	180	43200	5040

6.3 Evaluation Mechanisms

- **TRTR**: Train on the 80% real train set and evaluate on the 20% held-out real test set.

[1] http://iops.ai/competition_detail/?competition_id=5&flag=1
[2] https://github.com/numenta/NAB/tree/master/data

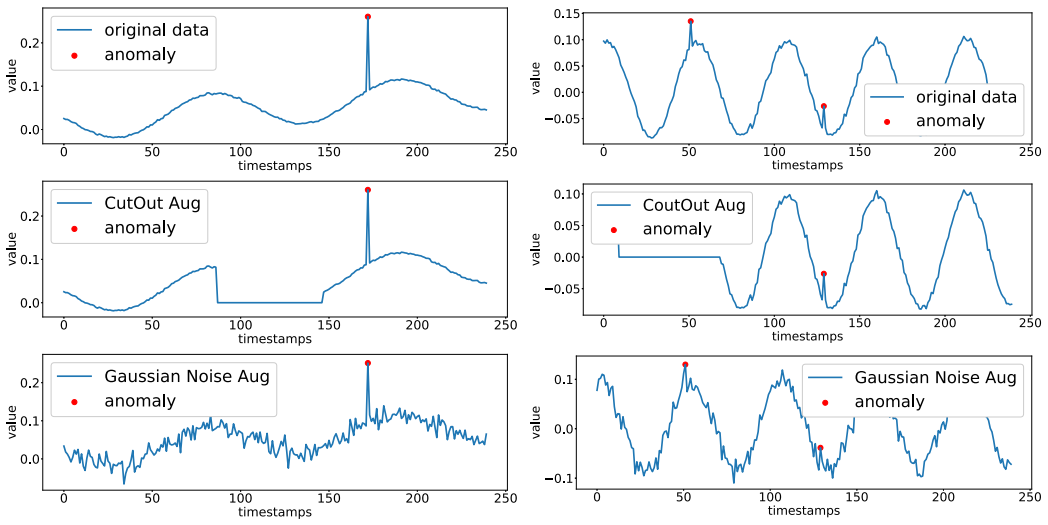

Figure 5: We augment 2 different sine curves with 2 popular DA methods from vision domain that have been applied to TS [38]- first, CutOut [13] where we randomly replace a small section of TS with 0; second, we add random noise sampled from a normal distribution. In both cases DA fails– either Standard DA techniques either introduce new anomalous behavior in the data or corrupt the anomalous timestamps leading to incorrect ground-truth labels.

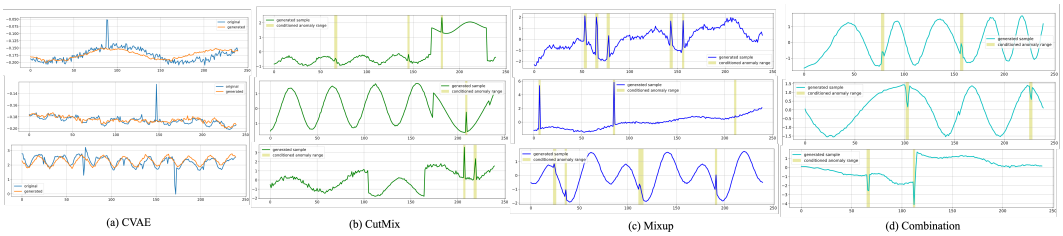

Figure 6: Samples generated by four different augmentation methods. Column (a) shows 3 different samples generated by a CVAE; Blue curve represents the input sample and orange curve represents the generated sample. Column (b), (c) and (d) show samples generated by CutMix [40], Mixup [43] and Combination (i.e jittering, scaling, permute and timewarp) augmentation methods.

- **T(R+S)TR**: Train on the combined data (i.e 80% train set and all of the generated synthetic data) and test on the held-out 20% real test set.
- **TSTR**: Train on the all of the generated synthetic data and test on the held-out 20% real test set.
- **TRTS**: Train on all the real data $(80\% + 20\%)$ and test on a 20% random split of synthetic data.

6.4 Evaluation Metrics

We observed that most of these AD methods [17, 29, 5] report performance scores using either a point-adjusted scoring function [3] or a relaxed version of F1 [17], which leads to an overestimation of detection performance [20]. We therefore use point-based evaluation metrics as recommended by [20] and also use the baselines by [37] and [20]. We report the average point-based precision, recall and F1 scores for each of the AD evaluation experiment performed. A series of AD experiments are run on each of the 6 datasets where we evaluate all the 8 different AD algorithms with 6 different DA strategies. Each experiment was run 5 times with different splits and we report the mean and standard deviation of the F1 score in Table 3, 9, 10 and 11.

6.5 DA Baselines

We pick 7 different DA techniques to compare against C−GATS. We also include an eighth baseline which is simply not using any augmentation method. Mixup [43] creates new training examples out of original samples by using a convex combinations of the features and their labels (controlled by α) resulting into plausible new TS (e.g. see Figure 6.c). We choose $\alpha = 0.2$ for our experiments as it yielded the best results across different values of α. CutMix [40] is another strategy popular in vision domain which involves randomly cutting a patch from one sequence and pasting it into another sequence. The new sequences generated by this method are not very ideal for anomaly detection task as abruptly changing patterns in a sequence could introduce new anomalies that labels won't account for (see Figure 6.b). As third baseline we pick RCGAN [14] as it is a neural-network based approach that applies conditionals in a temporal fashion like C−GATS. Out fourth baseline is a combination of traditional TS augmentation methods [3] including Scaling, Jittering, Permutation and TimeWarping. In this method, for each sequence in a given dataset, we randomly sample one DA strategy from the mix and apply it to the sequence. (e.g. see Figure 6.d).

6.6 AD Baselines

We pick 3 supervised neural-network based AD algorithms– RobustTAD [17], SR-CNN [29] and NCAD [5] as each claim to utilize DA techniques for better detection performance making them an ideal choice for our experiments. Apart from these, we select 5 other baselines that are much simpler but have been shown to be effective on the TS AD benchmark datasets [37, 20]. We refer to them as One-liner A, One-liner B, Case 1, 2 and 3. **Case 1** [20] baseline randomly assigns an anomaly score for every timestamp in a given sequence, i.e $\mathcal{A}(x_t) \sim \mathcal{U}(0, 1)$. **Case 2** [20] baseline assigns a score proportional to the value at each timestamp, i.e $\mathcal{A}(x_t) = ||x_t - \eta||_2$ where $\eta = 0$. **Case 3** [20] baseline is same as Case 2 but η is obtained from an untrained 2-layer LSTM neural network whose parameters are fixed after being initialized from a Gaussian distribution $\mathcal{N}(0, 0.02)$. The scores are then converted into anomaly labels using a threshold $0 \leq \delta \leq 1$. We do a grid search for the value of δ and report the metrics for the best value. Another set of baselines obtained from [37] are one-liners. There are 2 main types of one-liners proposed in [37] one with abs and without. We test both the categories and pick the one that obtain best performance. We pick **One-liner A** as —

```
abs(diff(TS)) > b
```

and **One-liner B** as —

```
abs(diff(TS)) > movmean(diff(TS), k)
              + c * movstd(diff(TS), k) + b
```

As recommended by [37], we adopt a similar brute-force strategy to compute individual **k**, **c**, **b** for all the datasets.

6.7 Ablation Study

To better understand the advantage brought by different components of our method, we perform an ablation study on 6 different datasets and 3 different AD algorithms. We consider variations of the framework by (a) varying the training procedure; and (b) altering the model architecture and train the model in each configuration thrice. We report the average performance metrics of these runs.

6.7.1 Dual-step training strategy

In this experiment we remove the training phase-I from C−GATS and simply train a standard CVAE with temporal conditionals with anomalous samples. Table 10 summarizes the performance of C−GATS in these two different settings. Our study shows that the use of dual-training step contributes to decoupling complex processes of representation learning and anomaly insertion which leads to generation of more controlled and sharper anomalies and contributes to additional performance gain. We further demonstrate this in Figure 7 where the left hand column represents sampled generated

[3] https://tsaug.readthedocs.io/en/stable/

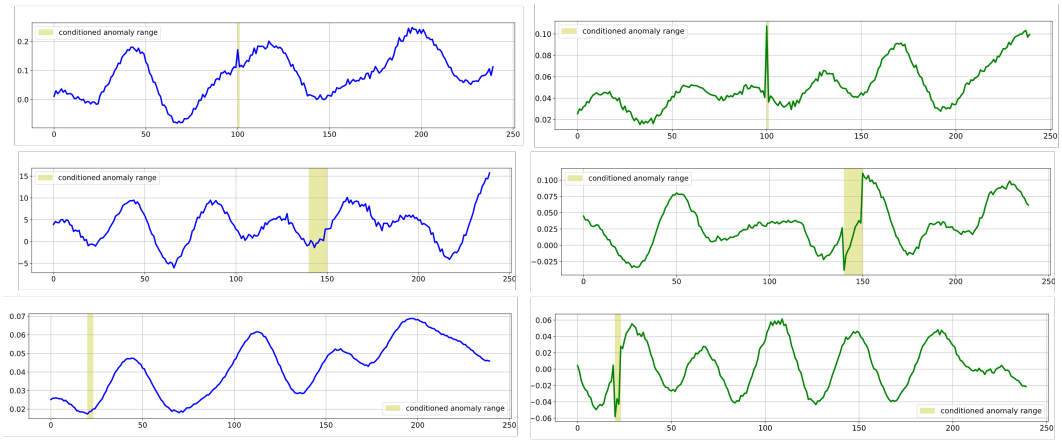

Figure 7: For a fixed z and Y^A in each row; Blue: samples generated by C−GATS in 1-phase training; Green: samples generated by C−GATS in a 2-phase training.

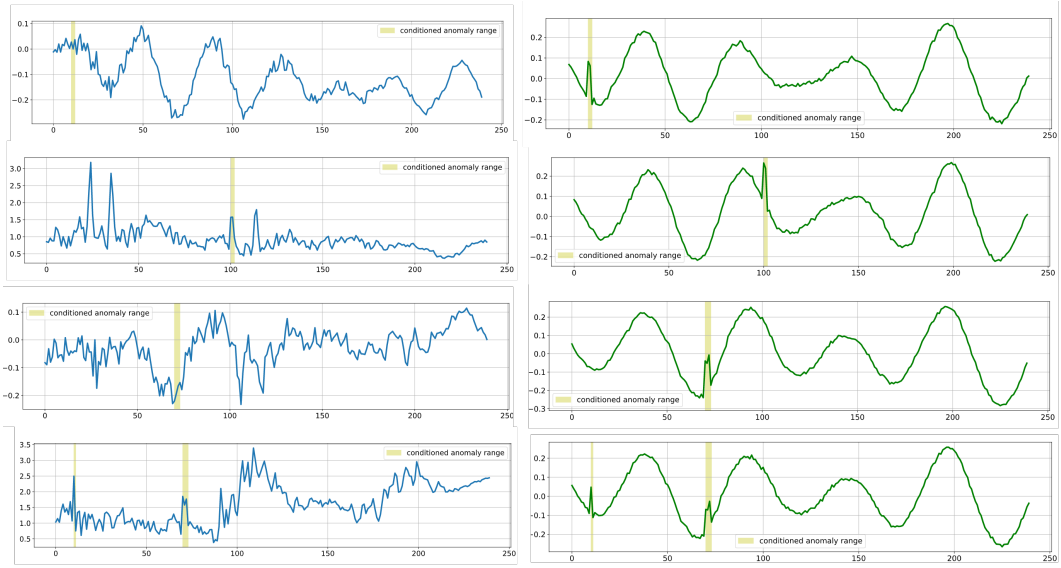

Figure 8: For a fix latent vector z, we vary Y^A in each row. Blue plots: represent samples generated by C−GATS when temporal conditioning is applied at the very first layer of the generator. Green plots: samples generated when conditioning is applied at the last layer of generator.

via one-step training procedure and right-hand column represents the dual-step training of C−GATS on KPI dataset. For each row, we use a fixed noise vector z sampled from a normal distribution along with a fixed condition vector Y^A. The samples generated by on-step training fail to learn sharp distinguishable anomalies in data which can be seen in the samples generated by the two-step training strategy.

6.7.2 Positioning of Conditionals

We study the effectiveness of our proposed approach of applying temporal conditionals for anomaly generation. By varying the position of the temporal conditionals we study its impact on the generation process. Changing the position of applying conditionals from final-most layer to the very first layer of the anomaly generator leads to a decline in AD performances by upto 12% in some cases, see Table 11. Neural networks are known to learn more fundamental and primitive features at the initial layers while more advanced and developed ones at the later layers [41]. Similar phenomena is observed when we assess the quality of generated anomalous samples in Figure 8. The study reveals that applying conditionals at the first layer of the generator interferes with the basic reconstruction of TS

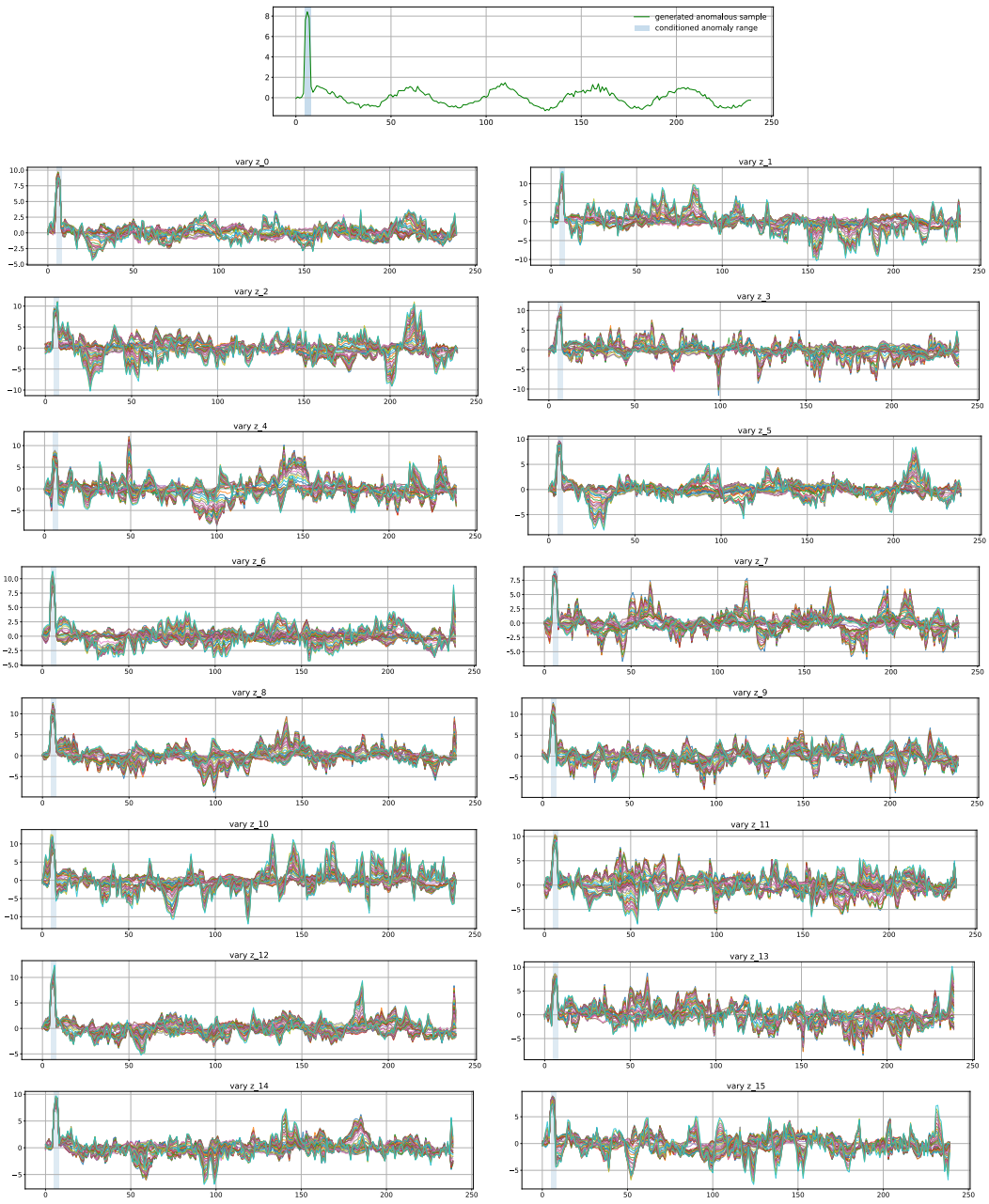

Figure 9: Top center: For a given conditional vector Y^A and a latent vector z sampled from a standard Gaussian, we generate an anomalous sequence. Then we vary z across each of its 16 dimensions one by one keeping the rest constant. For a given conditional vector and a latent z C−GATS can generate multiple reasonable and interesting synthetic anomalous signals.

signal and leads to ambiguous and inconsistent anomalous samples. Whereas when applied at the final-most layer, the generated samples are consistent and better in quality and hence contribute to better AD performance.

6.7.3 Diversity in generation of anomalies across a fixed input

Here we assess the generative power of C−GATS qualitatively in Figure 9. For a fixed condition vector Y^A we take a random noise z which is a 16-dimensional vector and generate an anomalous

Algorithm 1 Training the foundation model

Input: an anomaly label model $p(Y^A)$; a sample occlusion model $p(\tilde{X}|X^N, Y^A)$; set of normal sequences $\{X_i^N\}_{i=1}^n$; batch size B.
Parameter: θ and ψ initialized randomly.
Output: A learned posterior model $p(z|\tilde{X}; \theta^*)$.

while SGD *not converged* **do**
  Sample a batch of B normal sequences $(X_i^N)_{i=1}^B$;
  Initialize an empty buffer $\mathcal{M}$ of size B;
  for $i = 1, \ldots, B$ **do**
    Sample Y^A from *label model* $Y^A \sim P(Y^A)$;
    Feed (X_i^N, Y^A) to *sample occlusion model* and obtain $\tilde{X}_i \sim p(\tilde{X}|X_i^N, Y^A)$;
    Collect samples $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\tilde{X}, X^N)_i\}$;
  end for
  $\epsilon \sim p(\epsilon)$; (Random noise for every datapoint in $\mathcal{M}$)
  Compute $\mathcal{L}_{\theta,\psi}(\mathcal{M}, \epsilon)$ via Eq.(7) and its corresponding gradients $\nabla_{\theta,\psi}\mathcal{L}_{\theta,\psi}(\mathcal{M}, \epsilon)$;
  Update θ and ψ using SGD optimizer;
end while

Algorithm 2 Training the anomaly generation model

Input: a trained variational encoder $p(z|\tilde{X}; \theta^*)$; a sample occlusion model $p(\tilde{X}|X^A, Y^A)$; set of anomalous sequences and labels $\{X_i^A, Y_i^A\}_{i=1}^m$; batch size B.
Parameter: ϕ initialized randomly.
Output: A learned generative model $p(X^A|z, Y^A; \phi^*)$.

while SGD *not converged* **do**
  Sample a batch of B anomalous sequences pairs $(X_i^A, Y_i^A)_{i=1}^B$;
  Initialize an empty buffer $\mathcal{M}$ of size B;
  for $i = 1, \ldots, B$ **do**
    Feed (X_i^A, Y_i^A) to *sample occlusion model* and obtain $\tilde{X}_i \sim p(\tilde{X}|X_i^A, Y^A)$;
    Feed $\tilde{X}_i$ to the trained encoder and obtain the latent representation $z_i \sim p(z|\tilde{X}_i; \theta^*)$
    Collect samples $\mathcal{M} \leftarrow \mathcal{M} \cup \{(z_i, X_i^A, Y_i^A)\}$;
  end for
  Compute $\mathcal{L}_\phi(\mathcal{M})$ (8) and gradients $\nabla_\phi \mathcal{L}_\phi(\mathcal{M})$;
  Update ϕ using SGD optimizer;
end while

sample as shown in the top-center of the figure. We then vary each of 16 dimensions of z one-by-one between values $(-1, 1)$ keeping the rest constant. The figure shows that on varying each dimension of z, we control different attributes of the generated sample such as frequency, noise, etc. while retaining the anomaly in the same desired location. This displays the generative power of C-GATS in producing a diverse range of synthetic anomalous samples.

6.8 Future Work

In the future, we plan to further improve C-GATS's generation quality by forcing it to generate not just realistic but hard examples that can further improve the quality of AD models. Similar ideas have been used in approaches like MODALS [8] where sampling in latent space is advised by a classifier to pick samples that can potentially fool the classifier. We believe adapting a similar strategy in C-GATS where we use a group of different anomaly detectors to aid the sampling process in latent space could force C-GATS to generate samples in feature space that are not only anomalous but can simultaneously fool these detectors, and hence improve downstream AD methods substantially. Another area of improvement is to model the latent distribution using a prior that captures temporal dynamics like [16].

Table 5: Performance of different AD algorithms in different augmentation settings on *RealTweets Dataset*

AD Algorithm	TRTR			T(R+S)TR			TRTS			TSTR		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
RobustTAD	0.53	0.57	0.54	0.59	0.59	0.57	0.47	0.54	0.49	0.57	0.57	0.56
SR-CNN	0.49	0.51	0.46	0.55	0.54	0.53	0.45	0.50	0.47	0.52	0.53	0.50
NCAD	0.55	0.60	0.57	0.61	0.64	0.61	0.49	0.55	0.53	0.58	0.63	0.59

Table 6: Performance of different AD algorithms in different augmentation settings on *RealTraffic Dataset*

AD Algorithm	TRTR			T(R+S)TR			TRTS			TSTR		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
RobustTAD	0.74	0.65	0.67	0.78	0.70	0.73	0.64	0.63	0.62	0.76	0.69	0.69
SR-CNN	0.69	0.60	0.64	0.71	0.63	0.66	0.68	0.55	0.58	0.70	0.61	0.65
NCAD	0.75	0.66	0.69	0.78	0.69	0.71	0.75	0.65	0.67	0.77	0.67	0.69

Table 7: Performance of different AD algorithms in different augmentation settings on *RealAWSCloud-Watch Dataset*

AD Algorithm	TRTR			T(R+S)TR			TRTS			TSTR		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
RobustTAD	0.30	0.48	0.34	0.31	0.48	0.34	0.25	0.42	0.29	0.30	0.48	0.34
SR-CNN	0.27	0.42	0.25	0.29	0.43	0.27	0.22	0.37	0.25	0.26	0.42	0.25
NCAD	0.29	0.49	0.35	0.31	0.49	0.35	0.24	0.45	0.26	0.28	0.49	0.35

Table 8: Performance of different AD algorithms in different augmentation settings on *ArtificialWith-Anomaly Dataset*

AD Algorithm	TRTR			T(R+S)TR			TRTS			TSTR		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
RobustTAD	0.44	0.32	0.36	0.50	0.37	0.41	0.41	0.30	0.34	0.47	0.35	0.37
SR-CNN	0.39	0.17	0.28	0.43	0.26	0.34	0.37	0.15	0.25	0.41	0.21	0.29
NCAD	0.45	0.36	0.39	0.50	0.42	0.45	0.43	0.35	0.37	0.48	0.40	0.41

Table 9: Baseline TS AD algorithms performances across different datasets

Algorithm	**Dataset**	**w/o Aug.**	**Mixup** *(alpha=0.2)*	**CutMix**	**Combination** *(Scaling, Jitter, Permute,TimeWarp)*	**RCGAN**	**C-GATS**
One-liner A	Synthetic Sines	0.70 $\pm$0.000	0.70 $\pm$0.000	0.70 $\pm$0.000	0.70 $\pm$0.000	0.70 $\pm$0.000	0.70 $\pm$0.000
	KPI	0.61 $\pm$0.000	0.61 $\pm$0.000	0.61 $\pm$0.000	0.61 $\pm$0.000	0.61 $\pm$0.000	0.61 $\pm$0.000
	RealTweets	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000
	RealTraffic	0.51 $\pm$0.000	0.51 $\pm$0.000	0.51 $\pm$0.000	0.51 $\pm$0.000	0.51 $\pm$0.000	0.51 $\pm$0.000
	RealAWSCloudWatch	0.38 $\pm$0.000	0.38 $\pm$0.000	0.38 $\pm$0.000	0.38 $\pm$0.000	0.38 $\pm$0.000	0.38 $\pm$0.000
	ArtificialWithAnomaly	0.19 $\pm$0.000	0.19 $\pm$0.000	0.19 $\pm$0.000	0.19 $\pm$0.000	0.19 $\pm$0.000	0.19 $\pm$0.000
One-liner B	Synthetic Sines	0.69 $\pm$0.000	0.69 $\pm$0.000	0.69 $\pm$0.000	0.69 $\pm$0.000	0.69 $\pm$0.000	0.69 $\pm$0.000
	KPI	0.56 $\pm$0.000	0.56 $\pm$0.000	0.56 $\pm$0.000	0.56 $\pm$0.000	0.56 $\pm$0.000	0.56 $\pm$0.000
	RealTweets	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000	0.49 $\pm$0.000
	RealTraffic	0.52 $\pm$0.000	0.52 $\pm$0.000	0.52 $\pm$0.000	0.52 $\pm$0.000	0.52 $\pm$0.000	0.52 $\pm$0.000
	RealAWSCloudWatch	0.36 $\pm$0.000	0.36 $\pm$0.000	0.36 $\pm$0.000	0.36 $\pm$0.000	0.36 $\pm$0.000	0.36 $\pm$0.000
	ArtificialWithAnomaly	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000
Case 1	Synthetic Sines	0.09 $\pm$0.004	0.09 $\pm$0.008	0.09 $\pm$0.003	0.09 $\pm$0.002	0.09 $\pm$0.003	0.09 $\pm$0.001
	KPI	0.08 $\pm$0.006	0.09 $\pm$0.003	0.08 $\pm$0.009	0.08 $\pm$0.010	0.09 $\pm$0.004	0.08 $\pm$0.001
	RealTweets	0.12 $\pm$0.009	0.11 $\pm$0.004	0.12 $\pm$0.001	0.12 $\pm$0.010	0.11 $\pm$0.008	0.12 $\pm$0.003
	RealTraffic	0.07 $\pm$0.002	0.06 $\pm$0.010	0.07 $\pm$0.005	0.07 $\pm$0.008	0.06 $\pm$0.001	0.07 $\pm$0.006
	RealAWSCloudWatch	0.12 $\pm$0.002	0.12 $\pm$0.007	0.11 $\pm$0.001	0.11 $\pm$0.011	0.12 $\pm$0.005	0.11 $\pm$0.004
	ArtificialWithAnomaly	0.14 $\pm$0.000	0.13 $\pm$0.003	0.14 $\pm$0.003	0.14 $\pm$0.001	0.14 $\pm$0.003	0.14 $\pm$0.001
Case 2	Synthetic Sines	0.43 $\pm$0.000	0.43 $\pm$0.000	0.43 $\pm$0.000	0.43 $\pm$0.000	0.43 $\pm$0.000	0.43 $\pm$0.000
	KPI	0.33 $\pm$0.000	0.33 $\pm$0.000	0.33 $\pm$0.000	0.33 $\pm$0.000	0.33 $\pm$0.000	0.33 $\pm$0.000
	RealTweets	0.21 $\pm$0.000	0.21 $\pm$0.000	0.21 $\pm$0.000	0.21 $\pm$0.000	0.21 $\pm$0.000	0.21 $\pm$0.000
	RealTraffic	0.14 $\pm$0.000	0.14 $\pm$0.000	0.14 $\pm$0.000	0.14 $\pm$0.000	0.14 $\pm$0.000	0.14 $\pm$0.000
	RealAWSCloudWatch	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000
	ArtificialWithAnomaly	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000	0.18 $\pm$0.000
Case 3	Synthetic Sines	0.43 $\pm$0.002	0.43 $\pm$0.001	0.42 $\pm$0.001	0.43 $\pm$0.003	0.43 $\pm$0.005	0.43 $\pm$0.004
	KPI	0.33 $\pm$0.003	0.33 $\pm$0.004	0.32 $\pm$0.001	0.32 $\pm$0.009	0.32 $\pm$0.006	0.32 $\pm$0.005
	RealTweets	0.20 $\pm$0.003	0.21 $\pm$0.005	0.21 $\pm$0.001	0.20 $\pm$0.001	0.20 $\pm$0.002	0.20 $\pm$0.000
	RealTraffic	0.18 $\pm$0.009	0.18 $\pm$0.010	0.18 $\pm$0.007	0.18 $\pm$0.003	0.18 $\pm$0.005	0.18 $\pm$0.011
	RealAWSCloudWatch	0.19 $\pm$0.001	0.19 $\pm$0.002	0.19 $\pm$0.002	0.19 $\pm$0.000	0.19 $\pm$0.001	0.19 $\pm$0.003
	ArtificialWithAnomaly	0.19 $\pm$0.004	0.19 $\pm$0.001	0.19 $\pm$0.008	0.19 $\pm$0.002	0.19 $\pm$0.003	0.19 $\pm$0.001

Table 10: Comparison of change in detection performance under different training strategies.

AD Algorithm	Dataset	w/o Aug.	C-GATS w/ end-to-end training	C-GATS w/ decoupled training
RobustTAD	Synthetic Sines	0.74 $\pm$0.005	0.74 $\pm$0.002	**0.75 $\pm$0.004**
	KPI	0.55 $\pm$0.011	0.63 $\pm$0.009	**0.67 $\pm$0.003**
	RealTweets	0.54 $\pm$0.021	0.56 $\pm$0.011	**0.57 $\pm$0.010**
	RealTraffic	0.67 $\pm$0.026	0.70 $\pm$0.021	**0.73 $\pm$0.015**
	RealAWSCloudWatch	0.34 $\pm$0.262	0.33 $\pm$0.281	**0.34 $\pm$0.192**
	ArtificialWithAnomaly	0.36 $\pm$0.275	0.39 $\pm$0.151	**0.41 $\pm$0.107**
SR-CNN	Synthetic Sines	0.73 $\pm$0.009	0.73 $\pm$0.001	**0.74 $\pm$0.003**
	KPI	0.54 $\pm$0.008	0.57 $\pm$0.004	**0.58 $\pm$0.007**
	RealTweets	0.46 $\pm$0.027	0.52 $\pm$0.018	**0.53 $\pm$0.009**
	RealTraffic	0.64 $\pm$0.021	0.65 $\pm$0.009	**0.66 $\pm$0.020**
	RealAWSCloudWatch	0.25 $\pm$0.319	0.25 $\pm$0.220	**0.27 $\pm$0.199**
	ArtificialWithAnomaly	0.28 $\pm$0.019	0.33 $\pm$0.015	**0.34 $\pm$0.011**
NCAD	Synthetic Sines	0.77 $\pm$0.008	0.77 $\pm$0.007	**0.79 $\pm$0.001**
	KPI	0.64 $\pm$0.006	0.67 $\pm$0.011	**0.69 $\pm$0.001**
	RealTweets	0.57 $\pm$0.019	0.60 $\pm$0.021	**0.61 $\pm$0.009**
	RealTraffic	0.69 $\pm$0.029	0.71 $\pm$0.012	**0.71 $\pm$0.019**
	RealAWSCloudWatch	0.35 $\pm$0.219	0.35 $\pm$0.117	**0.35 $\pm$0.111**
	ArtificialWithAnomaly	0.39 $\pm$0.101	0.44 $\pm$0.082	**0.45 $\pm$0.091**

Table 11: Comparison of change in detection performance under different architecture choices.

AD Algorithm	Dataset	w/o Aug.	C-GATS w/ conditioning at first layer	C-GATS w/ conditioning at last layer
RobustTAD	Synthetic Sines	0.74 $\pm$0.005	0.74 $\pm$0.003	**0.75 $\pm$0.004**
	KPI	0.55 $\pm$0.011	0.60 $\pm$0.007	**0.67 $\pm$0.003**
	RealTweets	0.54 $\pm$0.021	0.55 $\pm$0.017	**0.57 $\pm$0.010**
	RealTraffic	0.67 $\pm$0.026	0.67 $\pm$0.019	**0.73 $\pm$0.015**
	RealAWSCloudWatch	0.34 $\pm$0.262	0.32 $\pm$0.198	**0.34 $\pm$0.192**
	ArtificialWithAnomaly	0.36 $\pm$0.275	0.36 $\pm$0.119	**0.41 $\pm$0.107**
SR-CNN	Synthetic Sines	0.73 $\pm$0.009	0.74 $\pm$0.002	**0.74 $\pm$0.003**
	KPI	0.54 $\pm$0.008	0.56 $\pm$0.005	**0.58 $\pm$0.007**
	RealTweets	0.46 $\pm$0.027	0.48 $\pm$0.011	**0.53 $\pm$0.009**
	RealTraffic	0.64 $\pm$0.021	0.65 $\pm$0.019	**0.66 $\pm$0.020**
	RealAWSCloudWatch	0.25 $\pm$0.319	0.25 $\pm$0.222	**0.27 $\pm$0.199**
	ArtificialWithAnomaly	0.28 $\pm$0.019	0.29 $\pm$0.018	**0.34 $\pm$0.011**
NCAD	Synthetic Sines	0.77 $\pm$0.008	0.77 $\pm$0.003	**0.79 $\pm$0.001**
	KPI	0.64 $\pm$0.006	0.66 $\pm$0.010	**0.69 $\pm$0.001**
	RealTweets	0.57 $\pm$0.019	0.58 $\pm$0.015	**0.61 $\pm$0.009**
	RealTraffic	0.69 $\pm$0.029	0.69 $\pm$0.021	**0.71 $\pm$0.019**
	RealAWSCloudWatch	0.35 $\pm$0.219	0.35 $\pm$0.109	**0.35 $\pm$0.111**
	ArtificialWithAnomaly	0.39 $\pm$0.101	0.41 $\pm$0.159	**0.45 $\pm$0.091**