1. Challenges and Motivation

Deploying resource-hungry deep learning based generative models in edge computing systems is hard.
- Edge devices have limited working memory.
- Network attached memories need higher bandwidths.
Existing solutions such as distributed model inference and model compression adversely impact service latency and generative capacity respectively.

2. Hypothesis

For a GAN model: “a sufficiently trained discriminator can also indicate how to modify the input to make it resemble more closely to the target distribution” We develop on this hypothesis to generate new samples using only the Discriminator, giving significant memory footprint gains.

3. The GON Model

We propose a new framework called generative optimization network (GON) that is similar to GAN, but does not use the Generator. Our framework is divided into three phases:
- Phase 1: Just like traditional GANs, use the real samples to train the discriminator to maximize its output.
- Phase 2: We start from the randomly generated noise samples and try to maximize the discriminator output by neural network inversion, i.e., fix the weights of the Discriminator and modify the input.
- Phase 3: We update the discriminator weights such that the output for new samples is minimized.

Our second phase may use stochastic gradient ascent or one of its momentum based variants. The working intuition is that the surrogate surface is highly non-convex and optimization over diverse noise samples would lead to distinct local optima. In principle, every optimum corresponds to a local element from the data distribution, allowing us to generate diverse samples.

4. Anomaly Detection at Edge

The GON framework is agnostic to the specific application. In this work we explore the important use case of anomaly detection at the edge.
- Importance of Edge: As applications become more demanding and privacy sensitive, edge devices are more prone to breakdowns, malicious attacks and intrusions.
- Training GONs: We train a GON model (\(D(x)\)) with a dataset of time-series windows of the set of utilization characteristics of each edge node.
- Anomaly Detection: At test time, for each timestamp \(t\), for an input time-series window \(x(t)\), we generate a reconstruction \(\hat{x}(t)\) by ascending the stochastic gradient \(\nabla_x D(x)\). We define the anomaly score as:

\[
S_t = \|x_t - \hat{x}_t\|
\]
Using dynamic thresholding techniques, we generate an anomaly label for each edge node.

5. Comparison with Baselines

- **Baselines**: One-Class SVM (OCSVM), Isolation Forest (IF), Local Outlier Factor (LOF), ONLAD [2], CAE-M [2], MAD-GAN [4] and SlimGAN [3].
- **Datasets**: Public datasets: SMD and MDS. Custom datasets (FTSAT-1/255) using a 10-node Raspberry Pi edge cluster.
- **Evaluation Metrics**: Detection F1 score and the ratio of F1 score with the memory consumption of the model (F1/GB).
- **Testbed**: Raspberry Pi 4B with 8GB RAM and Ubuntu Desktop 20.04 OS.

Results show that having only a single discriminator network allows us to significantly reduce the network's number of parameters and give a performance boost compared to having two such networks. Specifically, GONs gives up to 32% higher detection F1 scores and 58% lower memory consumption, with only 5% higher training overheads compared to baselines.

<table>
<thead>
<tr>
<th>Method</th>
<th>FTSAT-1</th>
<th>FTSAT-25</th>
<th>FTSAT-55</th>
<th>SMD</th>
<th>MDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCSVM</td>
<td>0.796</td>
<td>0.874</td>
<td>0.949</td>
<td>0.811</td>
<td>0.786</td>
</tr>
<tr>
<td>DLOF</td>
<td>0.783</td>
<td>0.872</td>
<td>0.961</td>
<td>0.822</td>
<td>0.845</td>
</tr>
<tr>
<td>CAE-M</td>
<td>0.709</td>
<td>0.866</td>
<td>0.936</td>
<td>0.796</td>
<td>0.776</td>
</tr>
<tr>
<td>MAD-GAN</td>
<td>0.789</td>
<td>0.889</td>
<td>0.941</td>
<td>0.828</td>
<td>0.843</td>
</tr>
<tr>
<td>SlimGAN</td>
<td>0.798</td>
<td>0.897</td>
<td>0.950</td>
<td>0.838</td>
<td>0.855</td>
</tr>
<tr>
<td>GON</td>
<td>0.813</td>
<td>0.899</td>
<td>0.950</td>
<td>0.841</td>
<td>0.866</td>
</tr>
</tbody>
</table>

6. Broader Impact

- **Few-shot handwritten digit generation**: GONs can learn to generate handwritten digits using only 10 samples from the MNIST dataset (60% less memory).
- **Density Estimation**: GONs can also estimate two-dimensional Gaussian distributions (82% less memory).
- **Anomaly Detection**: GONs give higher anomaly detection scores with small training overheads compared to the state-of-the-art.
- **GANs can be applied to other generative modelling tasks.**
- **Code**: https://github.com/imperial-gore/GON

7. Key Takeaways

- GONs have higher anomaly detection scores with small training overheads compared to the state-of-the-art.
- GONs can be applied to other generative modelling tasks.
- Code: https://github.com/imperial-gore/GON

8. References