Anomaly detection using AutoEncoders

By : Mahmoud Gamal and Abdelrahman Ibrahim

Contents

- Problem Recap
- Proposed Approach
- Timeline and progress
- Results so far
- What’s next?

Contents

Anomaly detection

- How to detect abnormal behavior?
  - Ex: Robotics movement
- What technique did we focus on?
  - Using AEs to reconstruct only the normal events, transactions or sequences
- What did we propose to do?
  - Trying this technique to solve existing problems

Are we using an existing model?

No, it's a new problem and we implement the model from **Scratch**

Anomaly detection

- What problem did we choose?
  - Fraud detection on transaction data found on Kaggle
    - IEEE Transaction fraud [3]
    - Credit card fraud detection [4]

Contents

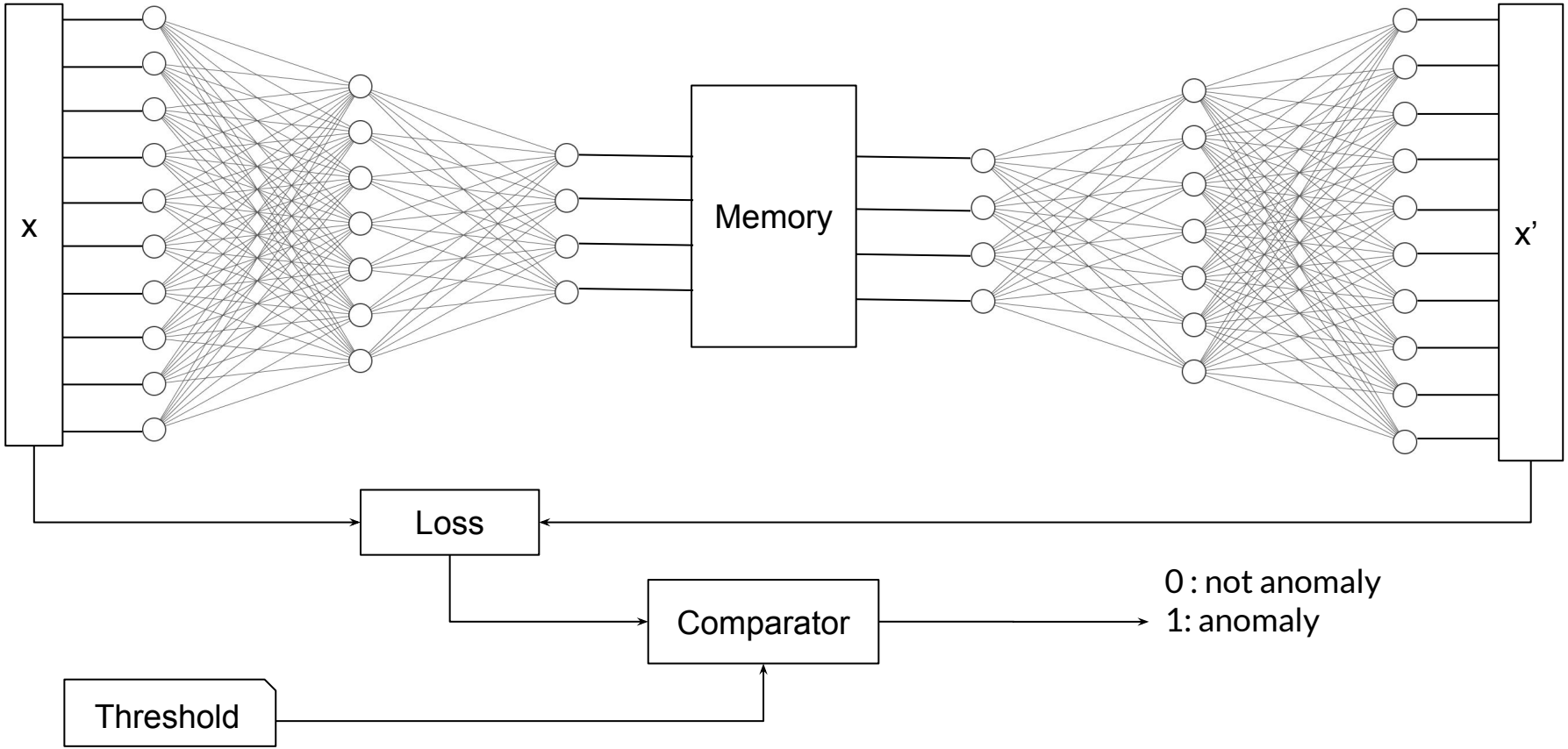
- Problem Recap
- Proposed Approach
- Timeline and progress
- Results so far
- What’s next?

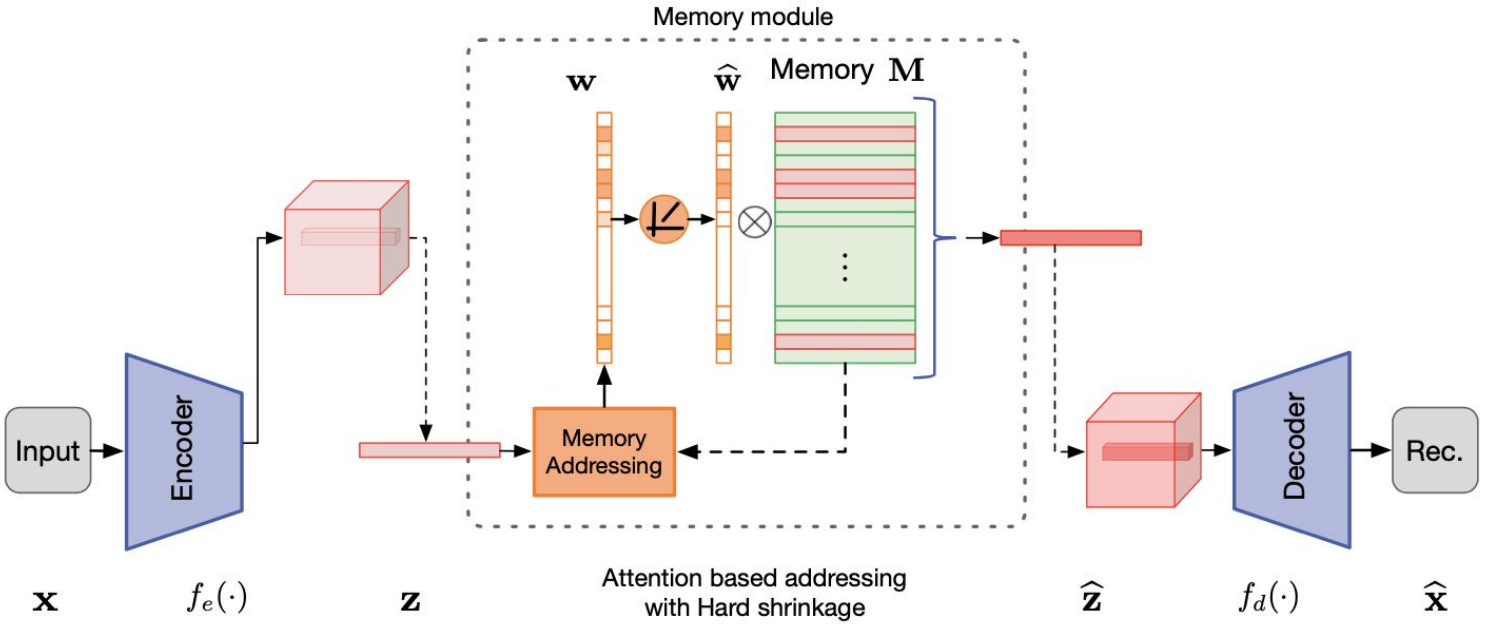
What did we add to the AEs?

- Using a memory element “last milestone”
- Exploring the deviation loss for anomaly detection. “This milestone”

Adding a memory element [2]

- Adding a memory element to remember the most frequent normal patterns in the input data.
  - With limited capacity to force the model to learn the most important patterns

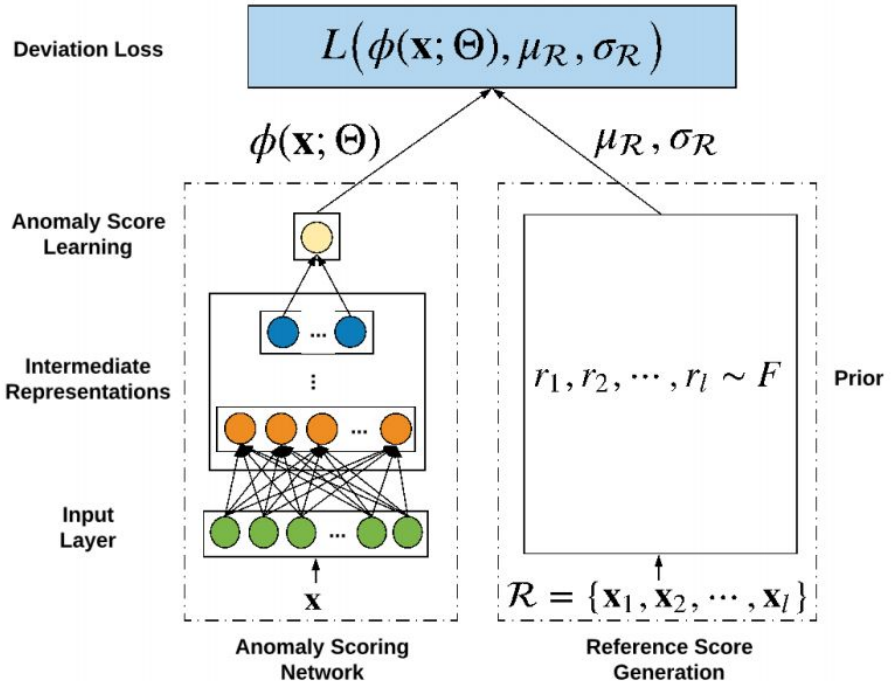

x
Memory
x'
Loss
Comparator
0 : not anomaly
1: anomaly
Threshold



$$\hat{\mathbf{z}} = \mathbf{w}\mathbf{M} = \sum_{i=1}^{N} w_i \mathbf{m}_i, \qquad w_i = \frac{\exp(d(\mathbf{z}, \mathbf{m}_i))}{\sum_{j=1}^{N} \exp(d(\mathbf{z}, \mathbf{m}_j))}, \qquad d(\mathbf{z}, \mathbf{m}_i) = \frac{\mathbf{z}\mathbf{m}_i^{\mathsf{T}}}{\|\mathbf{z}\| \|\mathbf{m}_i\|}.$$

Using Deviation loss [1]

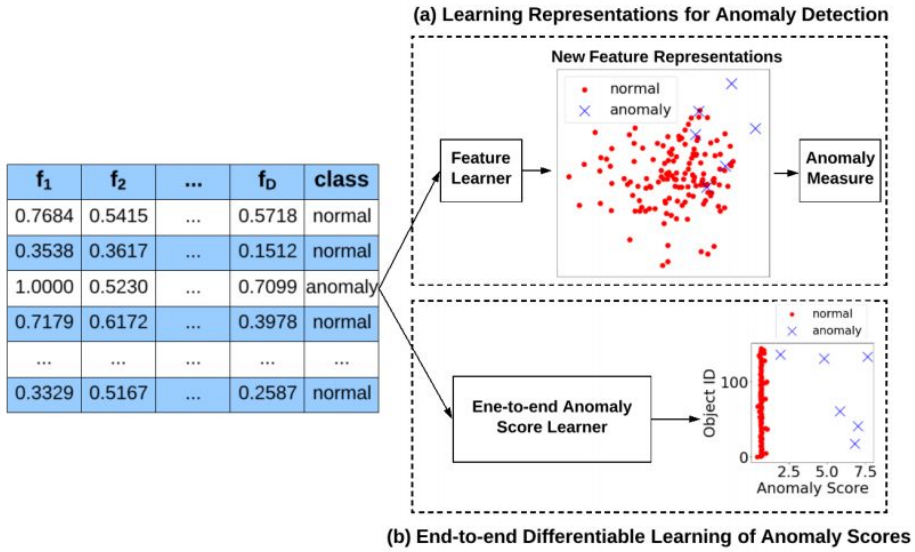
- A classifier with 1 node at the end acting as regression
- We used a pre-trained AE “on normal data only” not an arbitrary network



Using Deviation loss [1]

- Feature extraction by AE
- End to end network pipeline

f_1	f_2	...	f_D	class
0.7684	0.5415	...	0.5718	normal
0.3538	0.3617	...	0.1512	normal
1.0000	0.5230	...	0.7099	anomaly
0.7179	0.6172	...	0.3978	normal
...	...	...	...	...
0.3329	0.5167	...	0.2587	normal



Using Deviation loss [1]

- Loss function

$$dev(\mathbf{x}) = \frac{\phi(\mathbf{x};\Theta) - \mu_{\mathcal{R}}}{\sigma_{\mathcal{R}}},$$

$$L\big(\phi(\mathbf{x};\Theta), \mu_{\mathcal{R}}, \sigma_{\mathcal{R}}\big) = (1-y)|dev(\mathbf{x})| + y \max\big(0, a - dev(\mathbf{x})\big),$$

Contents

- Problem Recap
- Proposed Approach
- Timeline and progress
- Results so far
- What’s next?

Timeline and Progress

Task	Progress
Simple AE Model Building	100%
Data Preprocessing	100%
Building Memory Layer	100%
Training + Tuning Memory to minimize MSE	50%
Investigate Deviation Loss function	100%
Freezing weights and building Deviation Loss Model	100%
Training on Deviation Loss	100%
Running on test data and submit on Kaggle	20%

Contents

- Problem Recap
- Proposed Approach
- Timeline and progress
- Results so far
- What’s next?

Alone the Memory approach **Failed** to output a useful result

Note

The best recall was 0.018 which is not satisfactory no matter what techniques we used.

Deviation loss approach alone Is Promising and is producing a recall of 0.844 but a very bad precision 0.002

Note

The accuracy is still around 99.8% which indicates that the negative class recall is still high

Challenges

- Data skewness
- Curse of dimensionality
- Very subtle differences
- Many classes of anomalies

Contents

- Problem Recap
- Proposed Approach
- Timeline and progress
- Results so far
- What’s next?

What is next?

- Tuning the deviation model more to increase the precision as possible.
- Try to merge the two approaches to see if they help each other.
- Try an analogous method to triplet loss function

References

1. https://arxiv.org/abs/1911.08623
2. https://arxiv.org/abs/1904.02639
3. https://www.kaggle.com/c/ieee-fraud-detection/data
4. https://www.kaggle.com/mlg-ulb/creditcardfraud

Thank you!