Anomaly Detection Using Autoencoders with Adversarial Training

Arash Omidi Hosseinabadi, Professor Di Cataldo
Politecnico di Torino

CONTENTS

LIST OF FIGURES

Anomaly Detection Using Autoencoders with Adversarial Training

***Abstract*—This project explores anomaly detection in the KukaVelocityDataset using autoencoders, enhanced with adversarial training to improve reconstruction performance. The focus is on developing a reliable methodology for identifying anomalies through reconstruction error and evaluating the impact of adding adversarial loss. The dataset consists of signals collected from a Kuka robotic arm, and the goal is to effectively detect anomalies by training an autoencoder and using adversarial techniques to enhance detection.**

I. Introduction

Anomaly detection is a crucial task in many industrial applications, including robotics, where identifying unusual patterns can prevent operational failures. This project aims to implement an autoencoder-based approach to detect anomalies in the KukaVelocityDataset, using an additional adversarial training step to evaluate its effectiveness.

II. Background and Motivation

The use of autoencoders for anomaly detection is well established, leveraging their capability to reconstruct normal patterns while generating high reconstruction errors for anomalous data. The dataset used in this project consists of signals from a Kuka robot, including normal operational data and data with anomalies introduced by slowing down certain movements. The motivation for this project came from lab sessions and consultations with Professor Di Cataldo and Mr. Alessio Mascolini, where we experimented with basic autoencoder models. To further enhance the approach, an adversarial loss inspired by GAN architecture was introduced to evaluate whether anomaly detection performance could be improved.

III. Methodology

A. Data Preprocessing

The dataset used for this project was the **KukaVelocityDataset**, containing files such as 'KukaColumnNames.npy', 'KukaNormal.npy', and 'KukaSlow.npy'. The data was preprocessed by cutting it into windows of size 5 (as determined through cross-validation) to ensure key temporal features were retained. Various normalization methods were also applied to the dataset, including:

- **MinMaxScaler**: Scaling the data to a range of [0, 1], which was found to provide the best results.
- **StandardScaler**: Standardizing data to have zero mean and unit variance.
- **RobustScaler**: Scaling data based on the interquartile range, making it robust to outliers.
- **MaxAbsScaler**: Scales each feature by its maximum absolute value.
- **PowerTransformer**: Applies a power transformation to make data more Gaussian-like.
- **QuantileTransformer**: Transforms features to follow a uniform or normal distribution.

B. Feature Reduction Techniques

To improve the performance of the models and potentially increase the accuracy of the discriminator, two different feature reduction methods were implemented:

- **Principal Component Analysis (PCA)**: PCA was applied to reduce the dimensionality of the data while retaining 95% of its variance. The reduced features were used to train the autoencoders and discriminator.
- **Feature Selection Using Mutual Information**: The most important features were selected based on their mutual information with the labels, aiming to focus the model on the most relevant attributes.

While these methods were implemented with the goal of improving model accuracy, the results showed no significant improvement in the discriminator's accuracy. Additionally, the anomaly detection capability weakened due to the reduced feature set. As a result, these methods were not adopted in the final implementation.

C. Autoencoder Design

Two autoencoder models were used in this project to determine which could achieve better anomaly detection performance.

Autoencoder 1: Basic Encoder-Decoder Structure

- **Encoder**: Consisted of 2 fully connected layers with 64 and 32 neurons respectively, using ReLU activation to introduce non-linearity.
- **Decoder**: Consisted of 2 fully connected layers with 64 neurons, followed by an output layer with sigmoid activation to reconstruct the input.
- This basic model was chosen as a starting point to understand how a simple encoder-decoder structure could perform in anomaly detection.

Autoencoder 2: Deep Encoder-Decoder Structure

- **Encoder**: Featured two modular blocks with BatchNormalization and ReLU activation, reducing the input to 64 and then 32 features.
- **Decoder**: Mirrored the encoder with two modular blocks, followed by an output layer with sigmoid activation.
- This model was selected because BatchNormalization helps stabilize training. It was inspired by lab courses on Generative AI, and experimentation showed it provided the best balance between complexity and performance.

D. Adversarial Training

An additional adversarial component was introduced to further improve the autoencoder's reconstruction capabilities. Following the concept of Generative Adversarial Networks (GANs), a discriminator was trained alongside the autoencoder to differentiate between reconstructed and real samples.

Discriminator Structure

- The discriminator consisted of 4 dense layers:
  1) **First Layer**: 256 neurons with Leaky ReLU activation (alpha=0.2), followed by layer normalization.
  2) **Second Layer**: 128 neurons with Leaky ReLU activation, layer normalization, and a dropout rate of 0.4.
  3) **Third Layer**: 64 neurons with Leaky ReLU activation, layer normalization, and an additional dropout of 0.3.
  4) **Output Layer**: A single neuron with sigmoid activation for binary classification.

The discriminator aimed to push the autoencoder to produce more realistic reconstructions.

E. Challenges and Current Issues

During training, the discriminator's accuracy plateaued at approximately 50%, indicating a potential issue in the training process. Several approaches were attempted, such as tuning the learning rate and making the discriminator deeper, but the problem persisted. Suggested future approaches include balancing training between the generator and discriminator, adding a gradient penalty, and implementing batch normalization.

IV. Results and Discussion

The autoencoders were evaluated using **precision**, **F1 score**, **recall**, and **ROC-AUC** as metrics.

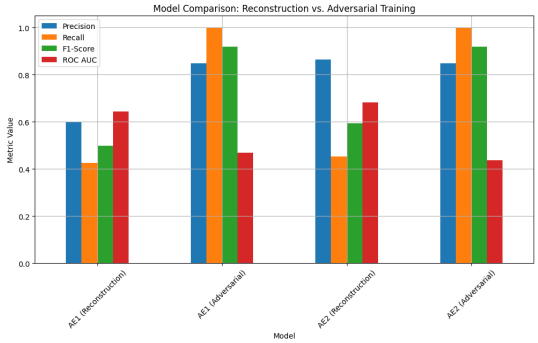

Fig. 1: Comparison of Autoencoder Models with and without Adversarial Training

The figure above compares the performance of Autoencoder 1 and Autoencoder 2 in reconstruction-only and adversarial training scenarios. Metrics such as precision, recall, F1 score, and ROC-AUC are visualized, demonstrating the impact of adversarial training on anomaly detection.

After incorporating the adversarial component, precision, F1 score, and recall showed noticeable improvement, indicating that adding the discriminator positively impacted the anomaly detection capability. However, the **ROC-AUC** metric experienced a slight decrease after implementing the adversarial model. This decrease suggests that while the adversarial component enhanced the model's ability to detect anomalies overall, it introduced challenges in effectively distinguishing between true positives and false positives, which could indicate potential issues in anomaly detection reliability.

Autoencoder 1 exhibited lower precision compared to Autoencoder 2 in reconstruction-only scenarios. However, after applying adversarial training, both autoencoders achieved nearly identical results across all metrics. This finding indicates that a simpler autoencoder, such as Autoencoder 1, can be used to achieve comparable outcomes while saving significant computational resources and processing time. This efficiency makes Autoencoder 1 a more practical choice for real-world applications where resources may be limited.

V. Conclusion

In conclusion, this project demonstrated the application of autoencoders for anomaly detection using the KukaVelocityDataset, with adversarial training used to enhance performance. While significant improvements were noted in recall, precision, and F1 score, the discriminator's performance remained limited. Future work should focus on stabilizing the adversarial component and refining the training dynamics to achieve better results.

VI. Future Work

- **Stabilizing Adversarial Training**: Investigate different architectures or learning rate schedules to stabilize the training process.
- **Extended Evaluation**: Apply the model to additional datasets to validate the generalizability of the approach.
- **Comparison with Other Techniques**: Compare the autoencoder with other state-of-the-art anomaly detection methods to evaluate its relative performance.
- **Explore Different Feature Reduction Methods**: Investigate various techniques to determine if they can enhance the discriminator's performance.

VII. References

- Lab works in the Machine Learning in Applications course.
- Lessons provided during the Machine Learning in Applications class.

VIII. Acknowledgments

I would like to thank Professor Di Cataldo and Mr. Alessio Mascolini for their valuable guidance throughout the project.