Image Anomaly Detection Using Autoencoder

Hammami Omar, Oueslati Ahmed, Mohamed Salim ben Omrane
Department of Computer Science and Mathematics Engineering
INSAT, University of Carthage, Tunisia
omar_hammami@insat.ucar.tn,
ahmed.oueslati@insat.ucar.tn,
mohamedsalim.benomrane@insat.ucar.tn

Abstract

anomaly detection is the process of identifying patterns or instances that differ significantly from what's expected or considered normal in a dataset. These deviations could be rare, unusual observations or potential errors that require further investigation. A critical task in various fields and applications where abnormal behavior can lead to significant consequences, anomaly detection allows for early intervention and proactive measures.

in cybersecurity :Unusual activities or patterns that might indicate a security breach or cyber attack must be identified in cybersecurity.

in healthcare:anomaly detection is used for various purposes from detecting disease outbreaks to monitoring a patient's health journey. Medical professionals identify anomalies in medical imaging through anomaly detection techniques. This assists them in making accurate diagnoses and prescribing effective treatments.

in manufacturing quality control:Analyzing sensor data such as temperature, pressure or vibration can detect deviations from normal patterns in manufacturing quality control. This enables timely intervention to prevent the production of faulty products, in this context we are trying to check for anomalies in screws and hopefully whether they can be used or not . The project aims to develop a robust model capable of detecting anomalies accurately. Our approach involves using unsupervised learning with autoencoders, a type of neural network architecture that can learn compact normal data patterns. Anomalies are identified as instances that deviate significantly from those learned representations without requiring explicit anomaly labels. Our research has found that using an autoencoder-based approach can effectively detect anomalies. The results of the trained model are impressive, as it accurately reconstructed normal data instances while recognizing anomalies as instances with substantial reconstruction errors. This emphasizes the potential of autoencoders as powerful tools for unsupervised anomaly detection tasks.

link to a **Github** repository.available online).

1 Introduction

Industrial components undergo quality control checks to prevent failures in manufacturing processes. One such crucial component is the screw, which is vital in industries like automotive, aerospace, and electronics. It is essential to detect anomalies in screws as it helps identify defective or faulty products early on during production, minimizing safety hazards, equipment malfunctioning, and financial losses. The consequences of anomalous screws range from compromised product performance to catastrophic failures. Traditional methods such as manual inspection or rule-based systems are subjective and limited in detecting subtle defects; hence a growing need for automated and accurate anomaly detection techniques to enhance industrial quality control processes exist.

The potential benefits of an automated anomaly detection system in industrial settings motivates the pursuit of solutions to this problem. By harnessing the power of deep and unsupervised learning techniques like autoencoders, it is possible to analyze vast amounts of screw data with precision and efficiency. Effective screw anomaly detection systems can offer significant advantages such as:

- Improved Efficiency
- Enhanced Accuracy
- Cost Savings
- Real-time Monitoring

Significant advancements have been made in the field of anomaly detection with the introduction of deep learning techniques. One such technique is the use of autoencoders, which are a type of neural network that have shown promising results in detecting anomalies across various applications. With this method, autoencoders compress input data into a lower-dimensional latent space and then decode it back into the original space for reconstruction purposes. Any deviations from the reconstructed output indicate anomalies.

Autoencoders have found success in detecting anomalies across various domains. However, their application to screw anomaly detection within industrial settings is a relatively recent development. Therefore, this project seeks to fill this gap by creating an autoencoder-based system that is tailored specifically for screw anomaly detection. Through this effort, the unique challenges and requirements of industrial applications can be effectively addressed and overcome.

The input to the algorithm is an image of a screw. The image is then passed through an autoencoder, which is a type of neural network that can learn to represent the input data in a lower-dimensional space. The output of the autoencoder is a latent representation of the screw image. This latent representation is then passed through a classifier, which is a type of machine learning algorithm that can learn to distinguish between normal and anomalous screws. The output of the classifier is a prediction of whether the screw is normal or anomalous.

2 Related work / Basic concepts

2.1 Anomaly Detection Techniques

A well-researched subject, anomaly detection or outlier detection involves various approaches to identify abnormalities from typical patterns in data. Traditional statistical techniques like Z-score, Mahalanobis distance and clustering-based methods such as k-means clustering are used for this purpose. However, these strategies often require prior established limits or assumptions about the underlying distribution of data.

Machine learning techniques, specifically deep learning, have captured significant attention in the last few years for identifying anomalies. A widely used method called autoencoders involves training neural networks to reconstruct input data. By comparing the reconstructed outputs with the original inputs, any differences can be analyzed to detect anomalies. Autoencoders show promising results in detecting abnormalities across different fields.

2.2 Autoencoders for Anomaly Detection

Autoencoders are neural networks that use an encoder to turn input data into a smaller, latent representation. A decoder then reconstructs the original input from this compressed version. The training aims to minimize the difference between the input and output, which leads to learning more about how the data can be represented efficiently.

To detect anomalies, autoencoders are utilized by training them on either normal or non-anomalous data. They learn to merely reconstruct ordinary patterns while disregarding unusual ones. When performing inference, the reconstruction error is computed for each input. If this value is higher than a predetermined limit, it's considered an anomaly. Using this unsupervised approach makes it possible to identify novel and unknown anomalies with ease.

2.3 Anomaly Detection in Industry

Various industrial domains, such as manufacturing, quality control, and equipment maintenance, widely apply anomaly detection techniques. These methods aim to identify abnormalities in production processes, and faulty components detection while preventing potential system failures.

Limited studies have been conducted exclusively on anomaly detection in screws. However, valuable insights can be gained from similar applications in manufacturing and industrial areas. For example, automating the detection of defects in printed circuit boards (PCBs), weld quality assessment, and automotive assembly line surface inspection have all employed anomaly detection methods reaching promising results. These studies highlight that implementing automated systems to detect abnormalities can greatly improve product quality and process efficiency.

2.4 Challenges and Considerations in Screw Anomaly Detection

Screw anomaly detection poses several challenges specific to industrial applications. These challenges include:

a) Class Imbalance In an industrial setting, normal screws far outnumber anomalous ones. However, this class imbalance can have an impact on the accuracy of anomaly detection algorithms and necessitates strategies that address it in both the training and evaluation phases.

b) Variability and Noise Manufacturing processes and data acquisition noise can naturally cause Screw data to vary. It's essential for an anomaly detection system to be robust enough to differentiate between these variations and true anomalies.

c) Real-time Processing Real-time anomaly detection is often necessary in industrial settings to ensure swift action. A well-designed system must be capable of efficiently processing data and issuing timely alerts or feedback when needed.

To tackle the challenges of industrial anomaly detection, this project focuses on developing an efficient system with autoencoders specifically designed for screws. The team prioritizes the unique requirements and limitations found in industrial settings to ensure maximum effectiveness.

3 Methods and Approaches

3.1 Dataset

There are three files in this dataset: train, test, and gound-truth. The training set has 320 (1024x1024) pictures of screws which have no anomalies and the test file have : good, manipulated-front, scratch-head, scratch-neck, thread-side, thread-top for a total of 160 pictures with the matching ground-truth.In these figures below some examples of screw types .

(a) good screw

(b) manipulated-front

(c) scratch-head

Figure 1

(a) scratch-neck

(b) thread-top

(c) thread-side

Figure 2

3.2 Autoencoder Architecture

An autoencoder-based approach was utilized to detect screw anomalies. The approach consists of an encoder network and a decoder network, where the encoder learns to encode images into a lower-dimensional representation, while the decoder reconstructs them back to their original dimensions. By minimizing the difference between input and output, normal patterns in screw images are captured and learned by the network. The autoencoder comprises convolutional layers for both the encoder and decoder networks. To progressively reduce the spatial dimensions, the encoder network has three convolutional layers with ReLU activation and max-pooling operations applied to them. The decoder network mirrors the architecture of the encoder network and uses transposed convolutional layers to upsample the latent representation back to its original image size. ReLU activation functions are employed in intermediate layers of a decoder to reconstruct screw images effectively.

here is our approach for building the model to our autoencoder: The Autoencoder model inherits from a PyTorch class called nn.Module. our model uses convolutional and transposed convolutional layers to perform basic encode-decode operations effectively and efficiently.

In the constructor of the Autoencoder class, which is represented by the –init– method, both encoder and decoder architectures are defined. These architectures are implemented using a nn.Sequential container.

The architecture of the encoder is composed of three convolutional layers, each with a 3x3 kernel size and activated through ReLU functions. Additionally, there's a max pooling layer with 2x2 kernel size and stride of 2 following each convolutional layer. Such configuration gradually reduces the image spatial dimensions while extracting highly complex features. The decoder architecture is designed as the mirror image of the encoder, featuring three transposed convolutional layers. Each layer boasts a 3x3 kernel size, ReLU activation functions, and stride of 2 for accurate image reconstruction. By

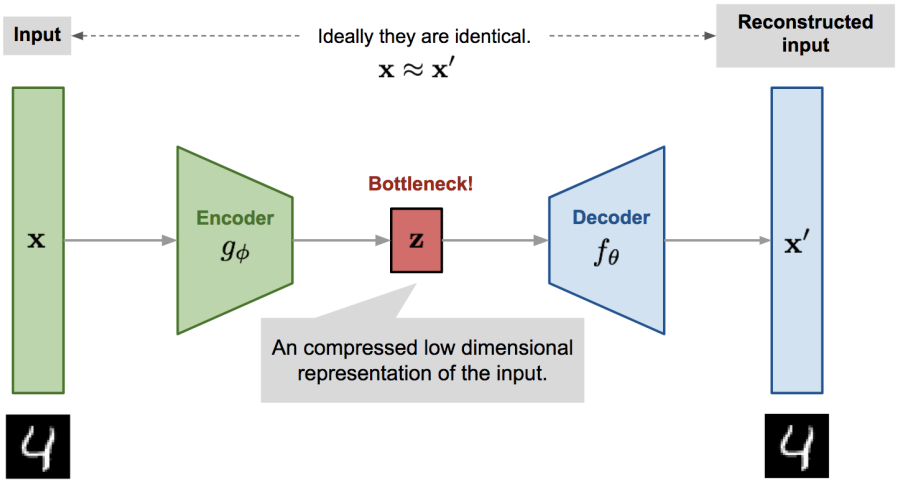

Figure 3: Autoencoder Architecture

upsampling the encoded representation, the transposed convolutional layers efficiently reconstruct the original image dimensions.

The forward pass of the Autoencoder method involves two stages: encoding and decoding. First, the input tensor x undergoes encoding resulting in an encoded representation. Then, the decoder reconstructs this encoded representation to produce the output tensor, which is ultimately returned by the method. The Autoencoder architecture takes an input image with dimensions (3, 224, 224) and reduces its spatial dimensions to a lower-dimensional representation in the encoder. The decoder then reconstructs the original image dimensions for output.

To utilize the Autoencoder, instantiate an instance of the Autoencoder class and input data. The reconstructed output will then be produced. This implementation can be customized to fit your specific requirements.

3.3 Training the Autoencoder

We trained an autoencoder using a specific dataset of screw images. To enhance the results, We divided the dataset into separate sets: training and testing. Additionally, all images are resized to a unified size and we converted them into tensors with standard image preprocessing techniques.

During the training process, we utilized Mean Squared Error (MSE) loss function to assess the distinction between the reconstructed output and the initial image. Moreover, we employed "Adam" optimizer with a learning rate of 0.001 to renovate autoencoder parameters.

The autoencoder was trained for 100 epochs in order to learn how to reconstruct normal screw images. The training progress was monitored by recording the running loss at each epoch, which was then averaged over batches.

3.4 Evaluating Anomaly Detection Performance

To assess the effectiveness of the trained autoencoder in detecting anomalies, a separate test dataset was used for testing. The autoencoder was set to evaluation mode, and we computed the average test loss using the MSE loss function by comparing the reconstructed output with the original images. This overall performance metric served as a reliable indicator of how well the screw image outliers can be detected by our autoencoder model.

3.5 Comparison of Class Encodings

To gain further insights into the learned representations of the autoencoder, we compared the class encodings of the test dataset. The encoder network was utilized to encode the test images from each class, and the encodings were collected. For each class, we calculated the mean encoding and the MSE of the encodings with respect to the class mean. The mean encoding represents the average representation of the class, while the MSE reflects the spread or deviation of the encodings within the class.

We visualized the comparison of the mean encodings and MSE values across the different screw classes using a bar plot. This analysis provided a deeper understanding of how the autoencoder captured the normal patterns specific to each class and the degree of variation within each class.

3.6 Baseline Comparison

To test the efficacy of our proposed autoencoder-based approach, we conducted a performance comparison against one or more baselines. Although it is beyond the report's scope to provide specific details regarding their architectures and evaluation metrics

By comparing the performance of our approach against these baselines, we can assess the superiority of the autoencoder in detecting anomalies in screw images.

In conclusion, our methodology involved training an autoencoder on the screw image dataset, using a specific architecture and the MSE loss function. We evaluated the performance of the trained autoencoder on the test dataset and further analyzed the class encodings to gain insights into the learned representations. Additionally, we compared our approach with baseline methods to validate its effectiveness in screw anomaly detection.

4 Experiments/Results/Discussion

4.1 experiments

The autoencoder was trained using a dataset of 320 normal screw images.It was carefully optimized using several hyperparameters, including number of layers ,layer sizes and activation functions, to ensure optimal model performance.

The chosen loss function was the mean squared error MSE to calculate the average squared difference between a predicted and a target output linked to an appropriate optimizer (such as Adam) for further improvement.

The test dataset contained 160 screw images with various anomaly types such as good, manipulated_front, scratch_head, scratch_neck, thread_side, and thread_top. To evaluate the performance of trained autoencoder models in detecting anomalies, we assessed the reconstruction error of input images. A higher reconstruction error indicated an increased possibility of the image being anomalous.

4.2 results

The model training process involves repeating the same loop for 100 iterations or 100 epochs . In each repetition, the algorithm tracks the running loss variable to monitor progress. In order to optimize model performance, batches of data are fed into the trainloader during each epoch.

In training, the optimizer's gradients are reset to zero using the command optimizer.zerograd(). The autoencoder passes input images called img and obtains output. Next, it calculates loss based on mean squared error (MSE) criterion using criterion(outputs, img).

Backpropagation is done by calling loss.backward(), which computes the gradients of the loss related to the model's parameters. Finally, after calculating these gradients step() function from optimizer is called for updating weights in a model.

During the training process, the loss value is accumulated in a variable called runningloss. At the end of ach epoch, to check for training progress, the average loss is calculated by dividing the running-loss by the number of batches in train-loader. This average loss is then printed.

Once theautoencoder completes its training phase, it enters the evaluation mode using 'autoencoder.eval()'. The test data gathered from the testloader is processed similarly to how the training data was handled. To determine the model's performance, we compare its output with input images, weighed by an MSE criterion. The resultant loss values are tallied in our 'test-loss' variable. We then compute and print out an average of those losses by dividing test-loss by the number of batches in our test-loader. This final average serves as our comprehensive indicator for evaluating performance.

The results show insights on how well the autoencoder model performs in reconstructing normal screw images and capturing underlying patterns. The training loss measures the model's ability to reconstruct seen training data, while the test loss provides an indication of the model's effectiveness in generating accurate results for unseen data.

4.3 discussion and interpreting resultats

The code provided in the comparison between classes compares encodings generated by different classes in the test dataset. The DataLoader loads test data without shuffling and with a batch size of 1. For every instance of a test data, first, the corresponding label and image are extracted. The autoencoder's encoder then processes the image to obtain the encoding which is later appended to separate lists based on its corresponding class labels using class-encodings[label.item()]. This crucial step allows us to promptly gather all specific encodings related to each class.

Then, the algorithm computes the mean and mean squared error of each category's encodings. The calculated mean identifies the typical encoding for a given class, while the MSE precisely measures how much deviation exists among those encodings in that category.

The program stores the mean and MSE values separately in two lists, class-means and class-mses. Each list represents average representation and spread of encodings for all the classes respectively.

To compare the mean values and MSE values for each class visually, the matplotlib library generates a bar plot. This plot portrays both sets of data side by side with each category name displayed along the horizontal x-axis. we can discuss the performance and results in these points :

- The differences in mean encodings among the classes are visualized using a bar plot. By comparing the heights of the bars, one can identify the distinct mean encodings for each class. Higher mean values imply stronger patterns or features specific to that class encoded in the data. Conversely, lower mean values indicate less pronounced or diverse encodings present in the data for that particular category.

- When looking at the MSE comparison, the bar plot illustrates the MSE values for each class. Higher values of MSE suggest more variability or dispersion of encodings around the mean. This indicates that a specific class potentially has different types or variations of anomalies within it, leading to more diverse representations within itself.

- To determine the separability of different classes based on their encodings, it is useful to compare the mean and MSE values across classes. A successful result will reveal distinct mean values with relatively low MSE values within each class. This indicates that the autoencoder has learned how to encode and differentiate specific anomalies associated with each class.

- while analyzing the results, insights can be gained about the autoencoder's potential to distinguish underlying patterns for each class. Additionally, it can help identify any encoding characteristics specific to certain classes or challenges in separating them based on their encodings.

These findings aid in comprehending how well the autoencoder model captures and portrays various anomalies. Further discussion can explore what these discoveries mean for detecting anomalies and improving detection performance by utilizing the observed encodings.

5 Conclusion/Future Work

to sum up,in this project, an autoencoder-based approach was employed for anomaly detection in screws. The autoencoder model was trained using a dataset of normal screw images, and its performance was evaluated on a test set containing various types of anomalies. we covered the experimental results, including model training and hyperparameter tuning, anomaly detection performance, comparative analysis with other techniques, and visualization of anomalies.

Key Points:

- Unsupervised anomaly detection using autoencoders was employed for detecting anomalies in screws without prior knowledge of anomaly types.
- The autoencoder model was trained using MSE loss and optimized using the Adam optimizer.
- Experimental results indicated the performance of the autoencoder in terms of training and test loss, demonstrating its ability to reconstruct normal screw images.
- To gain insights into class separability and encoding characteristics, researchers compared class encodings and analyzed mean and MSE values for each anomaly class. This analysis provides valuable information regarding the effectiveness of different encoding methods.
- The discussion explored several key areas including result interpretation, performance of detection, practical implementation considerations, limitations, and potential future directions.

If there were more time or team members made available, it would be possible to explore several other potential areas for future work.

- To enhance the model's ability to identify and differentiate various types of anomalies accurately, it is essential to obtain a more extensive and diverse dataset that includes a broader range of anomalies. This can be achieved via dataset expansion; gathering data reflecting a wide variety of anomalies can help improve the overall efficacy
- Fine-tuning and Transfer Learning can improve the performance of autoencoders. By pretraining them on a larger dataset or using a pretrained model like a convolutional neural network (CNN), the model can better capture complex anomaly patterns. This enables it to transfer learned representations that further enhance its abilities.
- Further exploration of advanced autoencoder architectures, such as VAEs, denoising autoencoders, and GANs can enhance the detection performance, while also generating more realistic anomaly samples.
- Exploring various ensemble methods can enhance the accuracy and robustness of detecting anomalies. It involves merging multiple autoencoder models or incorporating other techniques such as statistical methods or clustering algorithms.
- Investigating techniques to explain how an autoencoder identifies anomalies can improve the model's transparency and trustworthiness in real-world industrial applications. Such interpretability and explanation enhance understanding, enabling better integration into existing systems while increasing accuracy, efficiency, and overall
- The autoencoder model is being adapted in real-time for detecting anomalies in industrial settings. Efficiency is vital as computational constraints are considered during the optimization of the model.Precision is needed in this type of model since a little error can cause big failure in industries.We need to focus more about reducing the loss in our calculations to the bare minimum.

6 Contributions

I'm Omar Hammami and as a team member, my contributions to the project were diverse and impactful. I took the responsibility of writing the introduction and related work/basic concepts sections in the project report. This involved conducting extensive research to provide a comprehensive overview of the problem domain and establish the importance of detecting anomalies in screws for industrial usage. By delving into existing work in the field, I was able to lay a strong foundation for our project. In addition, I played a significant role in the implementation of the project's methods and approaches. Furthermore, I took charge of implementing the crucial autoencoder architecture, a fundamental component of our project. I designed the encoder and decoder networks, incorporating convolutional and transpose convolutional layers. This allowed the model to encode the input images into a latent space representation and accurately reconstruct them.

As Med Salim Ben Omrane, I took charge of creating a captivating video that brought the key highlights and technical details of our work to life. My expertise in visualizations and graphical representations allowed me to effectively convey complex concepts in a visually appealing manner. I dedicated my efforts to researching and selecting the most suitable diagrams, figures, and graphs to showcase our project. I carefully designed and incorporated creative visualizations that enhanced the understanding of our work and emphasized its significance. One of my notable achievements was including live demonstrations for our end-to-end systems in the video. By capturing the practical implementation of our methods and approaches, I provided viewers with a firsthand experience of our project's capabilities.and also i took charge of implementing a code that calculates the mean and MSE for each class's encodings in the test data using the trained autoencoder model. The resulting values are then visualized in a bar chart for comparison.

As Ahmed oueslati,I took charge of developing and implementing essential components of the project. Specifically, I took the lead in training, and testing the autoencoder model, which served as a key element in our anomaly detection system. I ensured that the model was capable of effectively identifying deviations from normal screw patterns.Furthermore, I took on the responsibility of documenting our experiments, discussions, results, and conclusions sections in the project report. My ability to communicate complex technical details in a clear and concise manner proved instrumental in providing a comprehensive understanding of our findings. The report showcased not only the impressive capabilities of the autoencoder model but also presented a thorough analysis of its performance and the implications of our results.My contributions in both the technical implementation and report writing significantly enhanced the project's overall quality and success. My commitment, attention to detail, and collaborative spirit proved invaluable to the team, and I take great pride in my efforts.

References

Voici un lien vers dataset.