Systematic Comparison of Auto Encoders in Industrial application

Kipras Klimkevičius
k.klimkevicius@student.utwente.nl
University of Twente
Enschede, Netherlands

Abstract

Anomaly detection is a field of AI that has significant benefits to many fields, from medical tests to industrial cases. However, anomaly detection models could become expensive to train, due to it being very expensive and difficult to gather enough data to train supervised models. Unsupervised models do not have this issue, since they can be trained on an imbalanced dataset, which as the name of the issue implies, is mostly the case in anomaly detection. Here we systematically compare Auto Encoder (AE) based approaches to unsupervised anomaly detection. For this purpose, we use an industrial dataset and find that Variational AEs perform more consistently and with better results than Convolutional AEs.

Keywords: auto encoder, variational autoencoder, comparison, industrial anomaly detection, transfer learning

ACM Reference Format:
Kipras Klimkevičius. 2018. Systematic Comparison of Auto Encoders in Industrial application. In . ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In this day and age Artificial Intelligence has greatly improved in many different fields. One field of importance is anomaly detection, even within this field, it finds extensive use in a number of contexts, from fraud detection to health care to military surveillance [3], or even industrial applications. For instance, detecting cracks in cement blocks or classifying possibly broken products. Detecting such defects is important since they are heavily linked with direct and indirect costs [8]. A way to intercept these defects would be to hire people to either do recurrent inspections or introduce someone into the production pipeline for product inspection. This however costs a lot of money and is not sustainable, in the sense that a person needs to constantly be present and check. However, Computer Vision can be of help here. We can classify images as damaged or not. There are different methods to do so, however, I will focus on Anomaly Detection to do said job.

Anomaly detection can be done in a multitude of ways. There are supervised and unsupervised learning methods. Supervised methods have led to significant improvements over the years, nevertheless, a problem with supervised methods is that it is extremely expensive to label data [11] and gather a balanced dataset. Unsupervised learning methods do not require the same exhausting work, since they do not require labelling of the data. For this reason, I will focus on unsupervised learning methods. There are numerous different unsupervised algorithms, however, I will focus on Auto Encoders (AE) in particular. AEs are a great option if we work with an imbalanced dataset. The reason is that they are trained on normal (non-anomalous) data, meaning that even if we have a dataset full of normal data, we can still train the model.

Fig 1 depicts the basic structure of an AE. The model will take the input and encode it multiple times, reducing the dimensions of the data to keep the most important features of the input. After encoding and reducing the input, the model will then learn to reconstruct the image by decoding the encoded input. As [4] put it so nicely, we can think of it as encoding $z = f(x)$ and decoding $y = g(z)$. We would then compare the input x and output y with some loss function $L(x, y)$. I shall refer to this type of AE as the Convolutional AE (CAE), where we use Convolutional layers to upsample, downsample, and learn the structure of our normal data. CAEs are especially well suited for image reconstruction. Besides that, I shall also use Variational Auto-Encoders (VAE) for the same purpose of anomaly detection. VAEs assume that the input is based on some Probability distribution and work by trying to learn the probability function of the input. So instead of purely learning how to encode, compress, and map the input onto the latent space, which is what CAEs do, VAEs learn to represent the latent space as a feature landscape and map input to their attributing features, instead of

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

just trying to embed the features on the latent space. This helps us in generating pictures more clearly since the whole latent space is utilised and the generated output will always be at least a merge of features, which might not be the case with CAEs (for reference look at [9][6]).

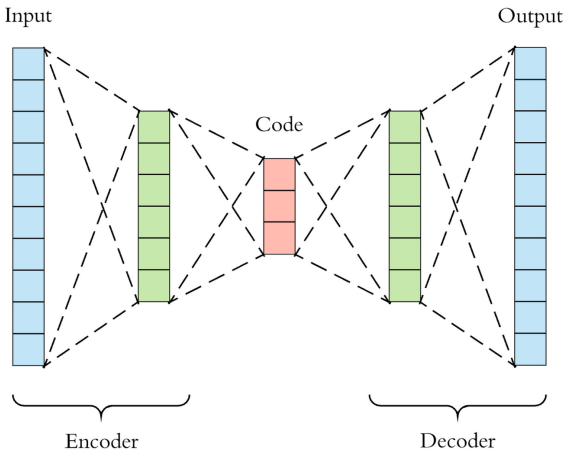

Figure 1. Basic Structure Of an Auto-Encoder

The purpose of this paper will be to compare AEs and VAEs with similar network structures on a dataset of industrial cases.

Within this research, I will focus on the following dataset [1]. This dataset contains pictures of castings of manufacturing products.

2 Problem Statement

Different AEs have been developed and tested on different applications, nevertheless, the industrial context is a standalone thing and the results in other areas might not compare. Thus this research will analyse different AEs with changes in their architecture and their training data, by adding noise to the original dataset, accenting some features more and some less. Different measurements in the accuracy of the individual models will reveal what combination of AE and data preparation shall prove most successful in industrial background.

2.1 Research Questions

The problem statement leads to the following research questions:

1. How do AEs and VAEs perform in terms of precision-recall trade-off for different kinds of anomalies introduced in real-world industrial dataset.
   The different kinds of anomalies examined will be:
   a. artificial additive noise (Gaussian, Salt & Pepper)
   b. additive noise made up of added objects
   c. real-world observable anomalies, found in the dataset
2. What performance improvements can AEs and VAEs achieve when using transfer learning?
3. How do the AEs and VAEs compare against each other in terms of precision-recall tradeoff when pre-trained models are used?

3 Related Work

In this section, I will go over some of the related work done in Anomaly Detection with Auto Encoders.

O.K. Oyabade and D. Aouada [10] explored how CAEs, Denoising AEs(DAE), Contractive AEs, and VAEs compare to each other in Anomaly Detection using 5 different medical datasets, specifically, they wanted to see how the different attributes of these AEs' latent space impact performance in unsupervised anomaly detection. They found that the probabilistic approach of VAEs does not always turn out better results than the other AEs. The most consistent was the CAE, while VAE even completely failed on one dataset.

An J. and Cho S. [2] used VAE in anomaly detection with Reconstruction Probability instead of Reconstruction Error as their evaluation metric. They found that their VAE outperformed classical AEs and Principal Component based methods. They concluded that the generative capabilities of VAEs allowed for the reconstruction to be utilised in deriving the causes of the anomalies.

Guo J. et all [5] created a contrastive AE for medical anomaly detection, where they used classical auto encoders for the baseline of their research. Interestingly the baseline AE did quite well, at times outperforming other more complex models.

4 Methodology

This section will describe how I plan to answer the research questions.

4.1 RQ1

To answer RQ1 I implemented a CAE and CVAE. Both of these models were trained using Google Colab and their NVIDIA T4 GPUs; they were trained on non-anomalous training data from the castings dataset.

The dataset includes 300x300 and 512x512 pictures of metal castings of products. Within these sets of pictures, there exists one training set and one testing set. Both of these sets include non-anomalous and anomalous pictures. In my experiment, I used the 300x300 sized pictures, which were further reduced to 256x256 to reduce the training times of the models. In addition to that, I normalize the pixel values

to be in the range $[0; 1]$ instead of $[0; 255]$. Otherwise, the pictures were not altered in any other way, however, they were used to create further datasets, with artificial anomalies, this is described later. In the 300x300 set, there were 2875 non-anomalous pictures. These were then split into a 80:20 ratios. 80% used for training (2300 pictures) and 20% used as validation data for testing. This validation set was used in one of the callbacks of the models which I will talk about later.

For the CAE I created an architecture consisting of convolutional layers followed by pooling layers, reducing the dimensions of the input from the shape (256, 256, 3), to a shape of (64, 64, 64), this includes having 3 convolution layers and 2 max-pooling layers - the output of the last convolution layer being the 'bottleneck'. The last 64 in the latent space shape represents 64 convolution kernels. The decoder architecture was the same except backwards. Thus the input shape being (64, 64, 64), and output - (256, 256, 3), with transposed convolution layers used for up-sampling. The loss function used was root mean squared error. No extra 'tricks' were used.

For the CVAE the architecture used was very similar to the CAE, except for 2 differences. The first is that there is one more pooling layer, meaning that the last convolution layer is also being pooled. The other difference is - that after the encoder's last pooling layer, I introduce a flattening layer, followed by dense layers to reduce the input to a latent space of shape (32) twice. One of these outputs is the mean of the learned distribution and the other - the logarithm of the variance. I use these to perform the reparameterisation trick [7] and make a sample for the latent space, which then gets decoded reshaped and up-sampled to generate a reconstruction of the input. The loss function used was $MSE - 0.5KLD$ where KLD is Kullback-Leibler Divergence.

Both of these models used these hyper-parameters for training:

- Epochs - 50
- Batch Size - 16
- Base learning rate - 0.003
- Optimizer - Adam
- Seed - 42

As well as these callbacks:

- Reduce Learning Rate on Plateau - if the validation data loss does not improve (decrease in value) for 2 epochs, we reduce the learning rate.
- Model saving callback, which would save the model weights every epoch, if the validation data loss has decreased.
- Early stopping - this callback stops training if, for 7 consecutive epochs, the validation data loss does not decrease.

To test these models I generate 3 types of anomalies, Gaussian noise, Salt & Pepper and additive pictures on top of the original images, as well as, the real-world anomalies. For Gaussian, I generated noise from a normal distribution with $\mu = 0.005$, and $\sigma = 0.5$ and added it to the original non-anomalous pictures. For S&P I randomly pick pixels to colour them white and pick pixels to colour them black and change them in the non-anomalous pictures. Lastly, for the added pictures I use the the Pillow library to add random objects on top of the original non-anomalous images Fig.(2 portrays an example). I created 3 datasets to test the different types of additive anomalies. One dataset for each different type. These test sets are comprised of 50% non-anomalous pictures and 50% of anomalous. These sets contain 524 pictures each. The only exception is the test set containing the real-world anomalies since this set is provided with the initial dataset. This test set contains 715 pictures in total, 453 of them being anomalous and 262 being normal.

For each type of anomaly, I test the models on 4 different metrics - precision, recall, f1-score and precision-recall AUC of the models to see how they perform. Below are the formulas for each metric.

$$Precision = TruePositive/(TruePositive + FalsePositive)$$

$$Recall = TruePositive/(TruePositive + FalseNegative)$$

$$F1Score = 2 * Precision * Recall/(Precision + Recall)$$

PRC-AUC is calculated by calculating the area under the precision-recall curve. I use these metrics to evaluate these models because they tell us the precision vs recall trade-off meaning that they measure how well the model identifies true anomalies while minimizing the amount of false positives/false negatives.

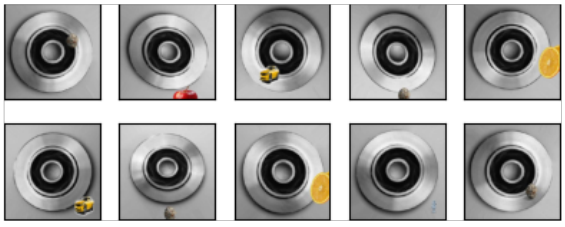

Figure 2. Example of additive noise of object like anomalies

4.2 RQ2

For RQ 2 I implemented a CAE and Convolutional VAE but this time using a pre-trained model with transfer learning, replacing the encoding layers, thus this time training the decoder layers. I used the same training data, and testing data and evaluated these models with the same metrics as the previous RQ.

For the transfer learning CAE, I used the VGG16 pre-trained model with frozen weights for the encoding. The latent space is gathered from the 'block_pool_ 2' layer, to

	RECALL	PRECISION	F1-SCORE	PRC-AUC
CAE	0.996	1	0.998	1
CVAE	0.996	1	0.998	1
TCAE	0.996	1	0.998	1
TCVAE	0.996	1	0.998	1

Table 1. Metrics Results of Basic Models on Additive Artificial Noise

	RECALL	PRECISION	F1-SCORE	PRC-AUC
CAE	0.855	0.941	0.896	0.966
CVAE	0.8625	0.918	0.89	0.955
TCAE	0.859	0.945	0.9	0.9705
TCVAE	0.84	0.905	0.871	0.953

Table 2. Metrics Results of Basic Models on Additive Artificial Noise Made Up of Objects

keep consistent with the number of pooling layers in the basic CAE. The decoder architecture has transposed convolution layers upsampling from a shape of (64, 64, 128) back to the original shape of (256, 256, 3).

For the transfer learning CVAE, I used the VGG16 pretrained model, exactly like for the CAE, again using 'block_pool_ 2' to stay consistent with the basic version of the model. Where I use a flatten layer, followed by a dense layer to go from shape (516096) to (32) twice, exactly like in the basic CVAE implementation, to perform the reparameterisation trick and get a sample from the learnt distribution. Where this sample is then densed back to 516096, reshaped and upsampled to get the input back.

Both models used the same hyperparameters as the models in RQ1.

I perform the same tests as for RQ1.

4.3 RQ3

For RQ3 I evaluate the performances of the models overall, picking which one performs best and comparing the models between themselves - for what types of anomalies which models do best - by looking at what scores are most appropriate for anomaly detection tasks.

5 Results

In this section, I will talk about the results gathered from performing the experiments presented in the methodology section. One thing to mention is that every result in every table is rounded to the 3rd decimal.

5.1 RQ1

The results of RQ1 can be split into 3 parts. The models' performances on artificial additive noise, additive noise made up of added objects and anomalies observable in the real-world.

First of all, table1 sums up the models' performance on the mentioned dataset. The models were perfectly accurate with both of them having 1 False Negative.

Secondly, the performances of the models, on the set of artificial added noise made up of objects, are summarized in Fig2. We can see the Precision-Recall curves plotted in Fig3. We can see that the CAE performs better in this regard. Only the recall is a better result for the VAE.

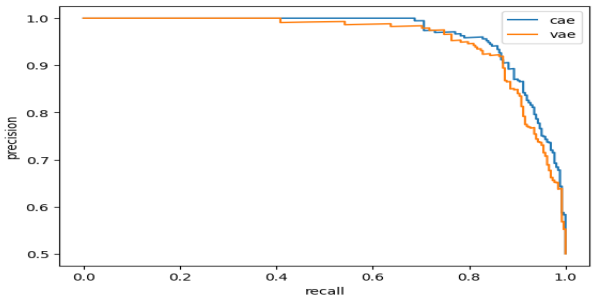

Figure 3. Precision-Recall Curves of the Basic Models' Performances on Additive Noise Objects

Thirdly, the results for the real-world anomalies are summarized in Table3 and the PRC plot in Fig.4. Most interestingly the CAEs results have diminished quite heavily. Going from high F1-score and PRC-AUC to much lower numbers. With the greatest decrease in precision. Meaning that the model quite frequently identifies normal samples as defects. Besides that, the recall score is improved. On the other hand, the results of the VAE have increased from great to nearly perfect scores with AUC being 0.998 and f1-score being 0.992 and both recall and precision being almost perfect.

Overall, both these models are perfect at identifying additive artificial noise. Which could, however, lead these models to fail at some points. Salt & Pepper noise could look similar to dust on the camera lens, meaning that if these models were used in the industry, an unkept camera, could start interfering and make the model predict anomalies. In terms of precision-recall trade-off, the VAE functions much better when tested on real-world anomalies, which is the type of anomaly which is encountered most in practice, for this reason, I would say that the VAE is the more appropriate choice. The VAE's performance slightly decreases when tested on the set of artificial added noise made up of objects, while, interestingly, the CAE's performance increases greatly. This could be due to convolution layers being great at deriving features from images, leading to the resulting features of the anomalous added pictures greatly diverging from the normal (learnt) ones.

	RECALL	PRECISION	F1-SCORE	PRC-AUC
CAE	0.993	0.653	0.788	0.837
CVAE	0.9955	0.989	0.992	0.998
TCAE	0.987	0.665	0.795	0.83
TCVAE	0.9955	0.989	0.992	0.997

Table 3. Metrics Results of Basic Models on Real-World Anomalies

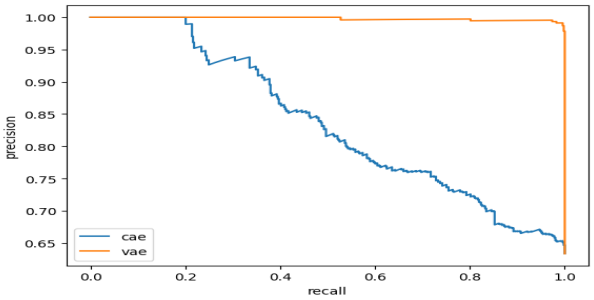

Figure 4. Precision-Recall Curves of the Basic Models' Performances on Real-World Anomalies

5.2 RQ2

The results of RQ2 can be split into 4 parts. The models' performances on the test sets, and how they compare to their respective basic models.

Firstly, The performance of the transfer CVAE on the set of artificial additive noise is summarized in Table 1 where we see that the performance on this set is identical to the the basic model. Its performance on the artificial additive noise made up of objects is summarized in Table 2, where the performance is similar to the basic VAE. Its performance on real-world anomalies is summed up in Table 3 where we can see nearly perfect results, essentially identical to the basic VAE, the only difference being the AUC, being very slightly worse. The difference in the AUC here tells us that the basic model performs just barely better over many thresholds. In general, the performance follows a very similar trend to how the basic VAE performed. Interestingly enough the basic VAE overall performed slightly better. This is especially clear when we look at their performances on the set of artificially added objects. Reflecting upon the posed question, the VAE model did not gain any performance improvements, the performance has even slightly decreased which is most apparent in Table 2.

Secondly, The performance of the transfer CAE is summarized in the same 3 tables as the previous models. Following the trend, the performance on the set of artificial additive noise is near perfect, with one false negative prediction on both S&P and Gaussian. The performance on the set of artificially added noise made up of objects is again very similar to the basic CAE model, with the transfer learning version gaining small performance increases in all metrics chosen. Similarly to the basic version, the performance also falls quite heavily when tested on the set of real-world anomalies. With the PRC-AUC falling from 0.97 to 0.83, again, having an increase in recall and a large decrease in precision.

Comparing the basic and transfer learning models scores we see that the transfer learning model performs slightly better in all measured metrics, regarding the performance on the set of added noise made up of objects, however, the interesting part is its performance in the set of real-world anomalies. The recall of the transfer learning model is slightly decreased, but the precision is slightly higher. This means that the transfer learning model will produce fewer false positives. In terms of this dataset and expenses in the industry, this implies fewer non-defect items will be classified as defects, thus reducing costs. On the other hand, the recall is slightly worse meaning more broken products are let through. However, the f1-score tells us that overall the precision-recall trade-off is slightly better for the transfer learning model. You could argue that the PRC AUC score is slightly better for the basic model but the f1-score, in my opinion, is more practical, since you have to choose a threshold for it. All in all, I would say that for the CAE, the transfer learning model gains performance increases as a whole on artificial anomalies; and in particular on precision and f1-score on the real-world anomalies.

5.3 RQ3

The results of RQ3 come from looking at the transfer learning models and comparing their performances with each other. Looking at the results of RQ2, it is no surprise that these models compare essentially the same as their basic versions compare to themselves. The CAE outperforms the VAE in terms of artificial added noise, nevertheless, the VAE is much more consistent across all datasets and greatly outperforms the CAE when it comes to real-world anomalies. Beating it in all of the chosen metrics. Essentially the same conclusions are drawn here as for RQ1 - VAE functions much better when tested on real-world anomalies, which are of most importance to us.

5.4 Discussion

Concluding the results of all tests we can see that the basic VAE is the model that performs best. The transfer learning version, even if being just slightly worse than the basic version, still overall outperforms the CAE models. Transfer learning managed to improve the performance of the CAE model just slightly. The fact that the respective transfer learning and basic models compare so closely to each other leads to show that for the task of anomaly detection in a specific field transfer learning will not provide an advantage, in some cases might even perform worse.

6 Future Work

Future work on this related topics should include (i) comparisons with other AE types, like Sparse AEs or Contractive AEs, and other Generative models like Generative Adversarial Networks (GANs). (ii) There can also be similar tests done on more various industrial datasets.

References

[1] 2020. casting product image data for quality inspection. https://www.kaggle.com/datasets/ravirajsinh45/real-life-industrial-dataset-of-casting-product

[2] Jinwon An and Sungzoon Cho. 2015. Variational Autoencoder based Anomaly Detection using Reconstruction Probability. https://api.semanticscholar.org/CorpusID:36663713

[3] "Adathakula Sree Deepthi". 2014. Anomaly Detection Using Principal Component Analysis. https://api.semanticscholar.org/CorpusID:27789499

[4] Tharindu Fernando, Harshala Gammulle, Simon Denman, Sridha Sridharan, and Clinton Fookes. 2021. Deep Learning for Medical Anomaly Detection – A Survey. arXiv:2012.02364 [cs.LG]

[5] Jia Guo, Shuai Lu, Lize Jia, Weihang Zhang, and Huiqi Li. 2024. Encoder-Decoder Contrast for Unsupervised Anomaly Detection in Medical Images. *IEEE Transactions on Medical Imaging* 43, 3 (2024), 1102–1112. https://doi.org/10.1109/TMI.2023.3327720

[6] Diederik P. Kingma and Max Welling. 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392. https://doi.org/10.1561/2200000056

[7] Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML] https://arxiv.org/abs/1312.6114

[8] Joseph Lowe, Balfour Beatty, and Laurie Brady Ljmu. 2014. Defects-Costs and Sustainability. https://api.semanticscholar.org/CorpusID:56113487

[9] Ryan O'Connor. 2022. Introduction to Variational Autoencoders Using Keras. https://www.assemblyai.com/blog/introduction-to-variational-autoencoders-using-keras/

[10] Oyebade K. Oyedotun and Djamila Aouada. 2022. A Closer Look at Autoencoders for Unsupervised Anomaly Detection. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3793–3797. https://doi.org/10.1109/ICASSP43922.2022.9746898

[11] Sjsu Scholarworks and Charles Thane MacKay. 2019. Learning for Free – Object Detectors Trained on Synthetic Data. https://api.semanticscholar.org/CorpusID:220883299

A Usage of Generative AI

During the course of this research I used the help of ChatGPT 4o to help decode and debug some code. It helped me debug and understand why my implementation of the CAE models would not learn anything, with the loss functions returning NaN values. It helped me find that the encoding of the logarithmic variance would return NaN values as well. With its help I managed to clip the values of the logarithmic variance between -10;10. Which then helped fix my models. I also used as a substitute for google sometimes in order to understand some libraries, for example, instead of looking through docs for a method and how to use it, I would ask chatGPT if there are methods for x thing in y library. It would tell me and if there would be, it would give me an example of how to use those methods. There was no other use of Generative AI.