DeepAI: Further Novel Applications of Deep Autoencoder Networks to Detect Anomalies in Large-Scale Financial and Accounting Data

Akhil Sreedhara, Andrew Aquino, Nicholas Gresh

New Jersey Institute of Technology, Newark, NJ

{as3638, ama347, ng584} @njit.edu

DS 677

11 May 2025

Video Presentation Link

GitHub Project Link

Abstract

In the modern world, financial and accounting data is vast and complex. With advances in technology, big data and its management is a large industry. With the scale of big data, systems are so large and complex that mistakes and fraud are inevitable. This has negative consequences for corporations, so there is a business incentive to account for, identify, manage, and prevent such risks. In a previous paper (Schreyer, M., & Sattarov, T., 2018) an autoencoder deep learning model was used for accounting data analysis, a novel technique from the typical use of autoencoder in image compression and reconstruction. In this paper, we implement and analyze the original model on the original dataset. This paper implements the analysis portion through examining the effects of tuning hyperparameters and model architecture, to identify what attributes make the model successful.

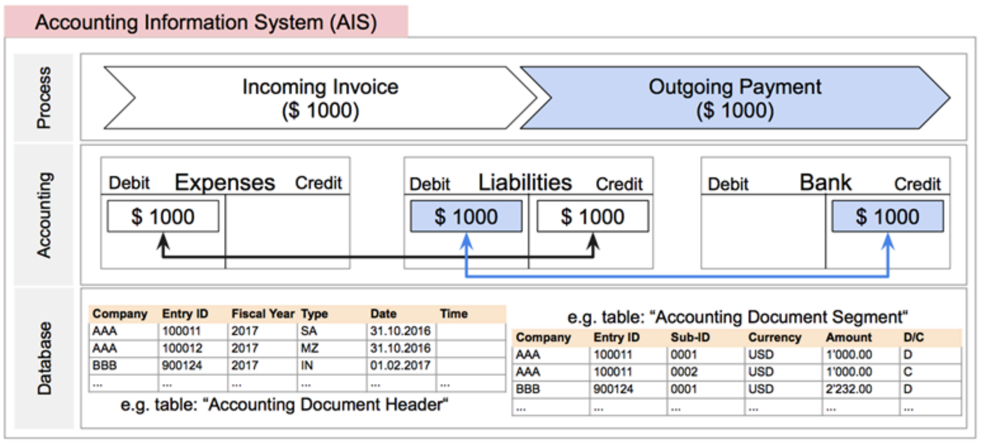
1 Introduction

First, we implemented the model from the original paper, then we built different models for the effects of different hyperparameters and model structure. In our work we tested different epochs, different learning rates, and a shallow model in comparison to the original deep multi-layered one.

2 Background on Financial and Accounting Processes

The model we created, like the original paper, was tested on accounting data. Such data is tracked by businesses as part of a system called ERP (enterprise resource planning), defined as “a software to help businesses streamline their core business processes” (*What is SAP?: Definition and meaning*. SAP. (n.d.)). ERP is used when a business has a need to track and edit complex sets of data. Such data included potentially thousands of data row entries, where each row contains a set number of column feature attributes. The types of features can be classified into the categories numeric and non-numeric, as defined below.

- Numerical Features: number values, could be amount of money, weight, length, physical dimension, etc.
- Non-Numerical Features: features not represented in numerical format, could be name of company, time of day, location address, transaction type, cost center, etc.



Company	Entry ID	Fiscal Year	Type	Date	Time
AAA	100011	2017	SA	31.10.2016	
AAA	100012	2017	MZ	31.10.2016	
BBB	900124	2017	IN	01.02.2017	
...	...	...	...	...	...

e.g. table: “Accounting Document Header“

e.g. table: “Accounting Document Segment“

Company	Entry ID	Sub-ID	Currency	Amount	D/C
AAA	100011	0001	USD	1'000.00	D
AAA	100011	0002	USD	1'000.00	C
BBB	900124	0001	USD	2'232.00	D
...	...	...	...	...	...

Figure 1. From the original paper, showing the database structure of an accounting transaction (Schreyer, Marco, et al.).

An example of an accounting process tracked in an ERP is shown above in Figure 1. The transaction is illustrated in three levels: process, accounting, and database. The process level is simple: a supplier invoices the buyer $1000 for a good or service supplied, then the buyer sends the $1000 payment to the supplier, completing the transaction. The accounting level shows the tracking of the invoice as an expense, and tracks the invoice payment as credit, and when those equalize the transaction is complete. The database is where it starts to get complicated. Each transaction is a row, with columns tracking transaction data. This particular data tracks the company name, entry ID, fiscal year, type, data, and more.

Figure 1 shows an accounting transaction specifically, but an ERP is a broad term that can apply to a variety of businesses, including different kinds of procurements, orders, sales, and more. There are a variety of possible ERP implementations, but there is one common denominator highlighting the challenges of any ERP: successfully managing vast data sets with thousands of entries, and potentially hundreds of rows for the feature elements.

To the human eye, you might be able to catch a million-dollar outlier cost, but the challenge is when the outlier is hidden in the data. For example, consider data where each entry has dozens of features, and an outlier entry has a particular unusual combination of features values for that entry. The human mind isn't equipped to observe that on our own, we need tools to do that. Machine Learning and particularly Deep Learning has a unique ability to find these hidden outliers.

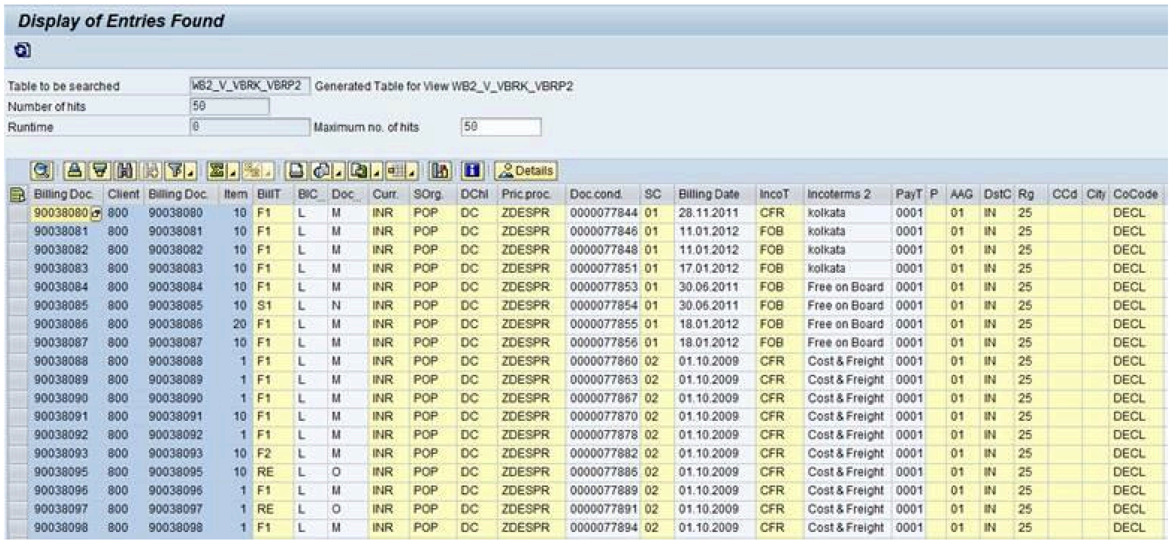

Display of Entries Found

Table to be searched	WB2_V_VBRK_VBRP2 Generated Table for View WB2_V_VBRK_VBRP2
Number of hits	50
Runtime	0
Maximum no. of hits	50

Billing Doc.	Client	Billing Doc.	Item	BillT	BlC	Doc.	Curr.	SOrg.	DChl	Pric.proc.	Doc.cond.	SC	Billing Date	IncoT	Incoterms 2	PayT	P	AAG	DstC	Rg	CCd	City	CoCode
90038080	800	90038080	10	F1	L	M	INR	POP	DC	ZDESPR	0000077844	01	28.11.2011	CFR	kolkata	0001		01	IN	25			DECL
90038081	800	90038081	10	F1	L	M	INR	POP	DC	ZDESPR	0000077846	01	11.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038082	800	90038082	10	F1	L	M	INR	POP	DC	ZDESPR	0000077848	01	11.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038083	800	90038083	10	F1	L	M	INR	POP	DC	ZDESPR	0000077851	01	17.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038084	800	90038084	10	F1	L	M	INR	POP	DC	ZDESPR	0000077853	01	30.06.2011	FOB	Free on Board	0001		01	IN	25			DECL
90038085	800	90038085	10	S1	L	N	INR	POP	DC	ZDESPR	0000077854	01	30.06.2011	FOB	Free on Board	0001		01	IN	25			DECL
90038086	800	90038086	20	F1	L	M	INR	POP	DC	ZDESPR	0000077855	01	18.01.2012	FOB	Free on Board	0001		01	IN	25			DECL
90038087	800	90038087	10	F1	L	M	INR	POP	DC	ZDESPR	0000077856	01	18.01.2012	FOB	Free on Board	0001		01	IN	25			DECL
90038088	800	90038088	1	F1	L	M	INR	POP	DC	ZDESPR	0000077860	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038089	800	90038089	1	F1	L	M	INR	POP	DC	ZDESPR	0000077863	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038090	800	90038090	1	F1	L	M	INR	POP	DC	ZDESPR	0000077867	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038091	800	90038091	10	F1	L	M	INR	POP	DC	ZDESPR	0000077870	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038092	800	90038092	1	F1	L	M	INR	POP	DC	ZDESPR	0000077878	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038093	800	90038093	10	F2	L	M	INR	POP	DC	ZDESPR	0000077882	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038095	800	90038095	10	RE	L	O	INR	POP	DC	ZDESPR	0000077886	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038096	800	90038096	1	F1	L	M	INR	POP	DC	ZDESPR	0000077889	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038097	800	90038097	1	RE	L	O	INR	POP	DC	ZDESPR	0000077891	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038098	800	90038098	1	F1	L	M	INR	POP	DC	ZDESPR	0000077894	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL

Figure 2. SAP sales billing data (Sardar, S., 2018).

SAP is a prevalent ERP software used across various corporations today, SAP's organization alone employs over 109,000+ employees across 157+ countries. (Company information | about SAP SE., n.d.). In Figure 2 is an example of a real life SAP business transaction. (Sardar, S., 2018) A transaction in SAP is a command sent to the system, telling it to pull a specific data table according to specific criteria. This particular transaction is a command to pull a "sales billing table". When SAP gets the command, it goes into the database and searches for each specific feature column needed for the transaction, as well as all the relevant row entries. It searches for sales billing data only, so it searches for entries that are entered when a sale is made. This data forms one large data table which is returned and shown on the screen. From here, the data can be filtered in each column to search for specific useful data points. The real-life usage of this could be to identify a particular sales entry within the larger returned data.

3 Integration of Autoencoder Model with Accounting Data

For the integration of our model with the accounting data, the first step was understanding the type of model we were seeking to build.

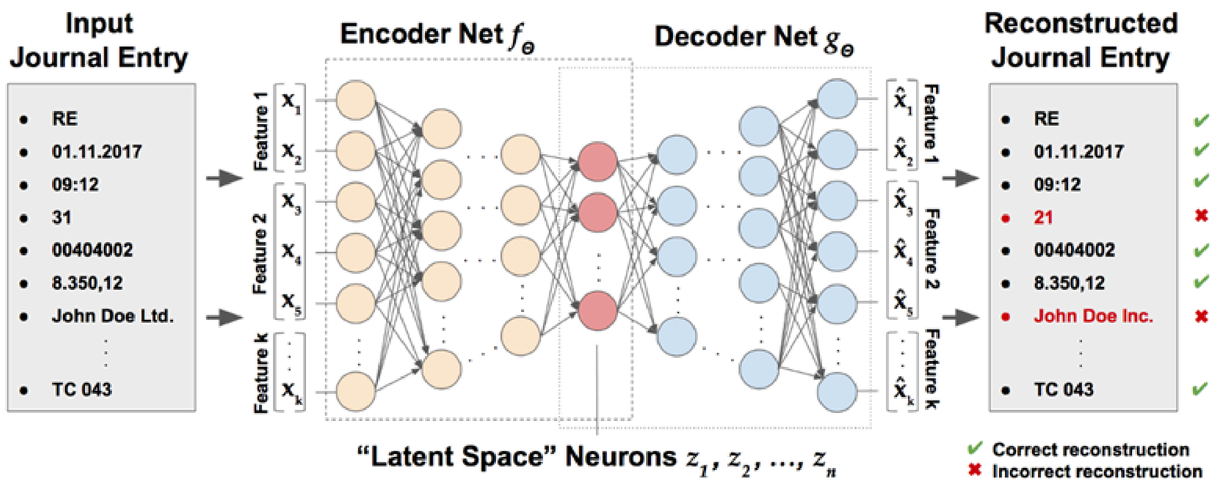


Figure 3. From the original paper, showing integration of accounting data with autoencoder models (Schreyer, Marco, et al.).

The autoencoder model architecture is illustrated in the figure from the original paper (Schreyer, Marco, et al.). The dataset is a table with thousands of data entries and around 10 features, spread across two Excel spreadsheets. The non-numeric features must be encoded into numeric via one-hot encoding, so eventually all the data becomes numerical, and the data increases to hundreds of features. The data is then input into neural network layers, so it becomes "encoded" by being displayed in progressively fewer neurons. It eventually reaches its most compressed state of three neurons. Then, it is "decoded" symmetrically to the encoding. This means the number of layers and number of dimensions per layer are symmetrical, but each neuron doesn't necessarily have the same weights and biases as its symmetric twin. However, the symmetric aspect of the autoencoder allows data to be compressed smaller, then later expanded to replicate the input data. Any normal data should be output the same as before, but the outlier data output will change to become data that more closely resembles the normal data. At the end the reconstruction loss can be measured, where data with low loss is likely normal, whereas data with high loss are likely outliers. Thus, the loss metric is what can flag potential outliers, and further review can confirm the data to be true outliers.

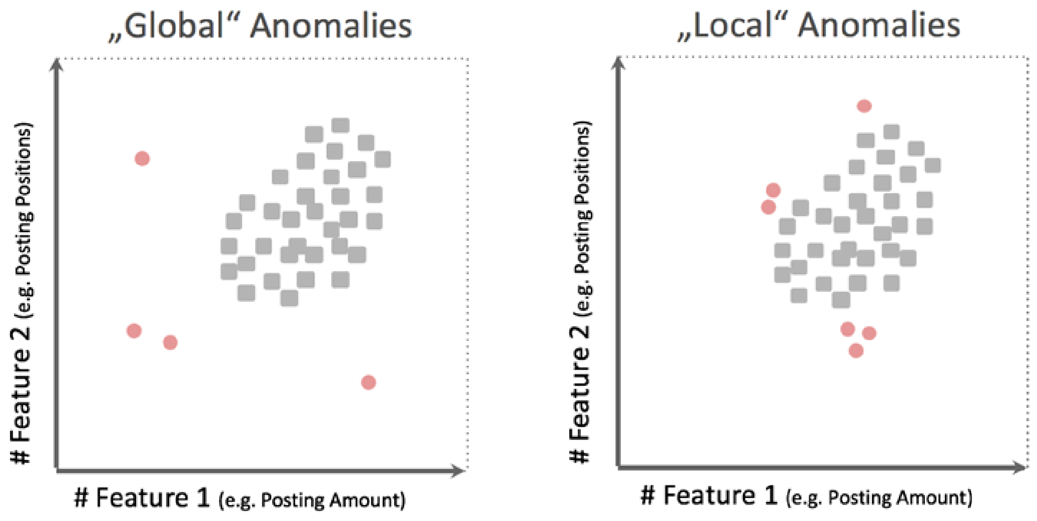


Figure 4. From the original paper, showing global vs. local anomaly (Schreyer, Marco, et al.).

In the figure from the original paper (Schreyer, Marco, et al.), the concept of "global" vs. "local" anomalies is illustrated. Each graph shows a plot of one feature vs. another feature for multiple data points.

A global anomaly could be like when transactions are usually $2,000 but now there is a $1,000,000 transaction. It refers to a data entry where one specific feature has a clear quantifiable outlier, visually observable on a graph for example.

A local anomaly is a data point where each entry feature is normal on its own, but the amalgamation of all the specific features together is unusual. Thus, the overall data point is an anomaly with everything put together. For example, if a raw wood materials supplier is invoicing an electronic component instead of the usual raw wood material. The raw wood supplier is in the system, electric components are in the system, but the fact that a wood supplier is buying electronics is an unusual combination.

In this paper, we implemented three different models to compare the reconstruction loss of the different model structures.

4 Related Works

One study that we came across that we thought would be helpful is "Detection of Anomalies in Large-Scale Accounting Data using Deep Autoencoder Networks" by Zahra Zamanzadeh Darban (Schreyer, Marco, et al.). Their strategy is to take the reconstruction error generated by the autoencoder and further refine them with individual attribute probabilities of journal entries. This dual faceted scoring system increases the adaptability to varying distributions, global anomalies, and local anomalies.

The study showed its efficacy of their proposed method on 2 datasets from SAP ERP systems. The autoencoder models achieved high f1-scores indicating a strong association between precision and recall in anomaly detection techniques. This approach outperformed other modern anomaly detection systems.

Another anomaly detection advancement we came across was a hybrid model (Najafi, S. M., Rezvani, S. J., & Wong, W. K., 2024). Attention and autoencoder Hybrid Model for Unsupervised Online Anomaly Detection by Seyed Amirhossein Najafi and group introduces an architecture that combines attention mechanisms and autoencoders, and also performed as well as transformer models.

This model is unsupervised and operates online which makes it suitable for real time anomaly detection without the need for labeled datasets. It uses positional encoding which allows for parallel computation, making the model more efficient. When evaluating on real world datasets, it was able to detect point, collective, and contextual anomalies. This hybrid model showcases the potential of combining different deep learning techniques.

5 Dataset Explanation

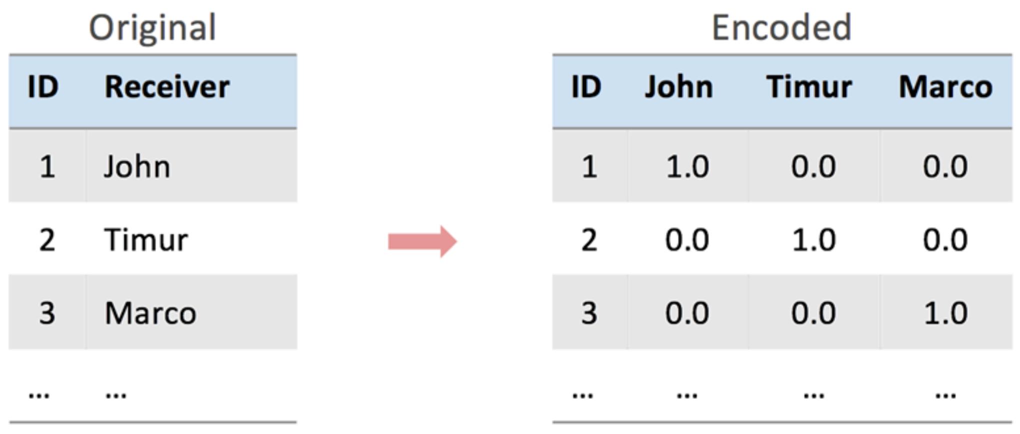
For our datasets, we used real world accounting journal entry data from SAP ERP systems. We use two datasets for journal entries which contain both natural and artificially injected anomalies. The dataset features consist of company code, fiscal year, document number, account type, posting key, amount fields, and metadata. The label has 3 possible outcomes, regular, local anomalies, and global anomalies. In total, we had over 1,000,000 records of data.

6 Model Explanation

The first step in creating the model is a Python notebook that brings in the initialization to allow all the necessary Python libraries to create a model. Then we load the data from a Github webpage where the Excel spreadsheet is stored. We plot the data table in Python for convenience so that the model is pulled correctly.

The actual data has some amount of normal data, but we also inject a small amount of global and local anomaly data (~0.03%) to see what the model detects for each anomaly type.

We plot the distribution of numerical values to see how the distribution of data points is numerically. Based on the plot, the categorical values have a couple high frequency options but most of the options are low frequency.



Original

ID	Receiver
1	John
2	Timur
3	Marco
...	...

Encoded

ID	John	Timur	Marco
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
...	...	...	...

Figure 5. From the original paper, one-hot encoding of non-numeric data (Schreyer, Marco, et al.).

After implementing the data into the notebook, we have to perform data preprocessing on the data. For any non-numerical data, we turn that non-numeric feature into multiple one-hot features to represent all the unique string phrases of that non-numeric feature. The figure from the original paper has an example of this (Schreyer, Marco, et al.). Then, these new numerical features are combined with the original numerical features, thus resulting in a new dataset of all numerical features. The one-hot encoding transforms the data from around 10 features to 618 numerical features.

We implemented an autoencoder network that compresses the data throughout multiple deep layers, thus it is a 'deep learning' model. The data starts out with the 618 features as input, then reduces neurons each layer into gradually smaller dimension multiples of 2. It reduces from 618 to 512, its output becoming a multiple of 2. Then it goes 512 to 256, to 128, 64, 32, 16, 8, 4. Finally, it goes from 4 to 3. At this point it reaches the center of the autoencoder, the most compressed format. Then, it decodes symmetrically in reference to the number of encoding layers, and how many neurons each layer has. Also for this model, key hyperparameters are established. A learning rate is established to affect how much each forward step modified the weights and biases of each neuron. Epoch number is established to determine how many passes the model has over the data. Mini-batches are established to improve efficiency instead of a pass over the whole date for each epoch.

The model trains in repeated passes, and we track the reconstruction loss over each epoch. We see that the model learns, and the loss goes down accordingly. Thus, it learns how to replicate the normal regular data and also convert outlier data into a normalized format. Now, in this context normalized means in terms of what made it an outlier before. The outliers are changed to be more normal in terms of a high-level combination of all the features in context. Throughout training, the reconstruction loss is tracked to measure how the model improves in detecting normal vs. outlier data.

7 Experiments and Results

We test the original data on the original model and find that after training, outlier data can clearly be distinguished from normal data via measuring reconstruction loss.

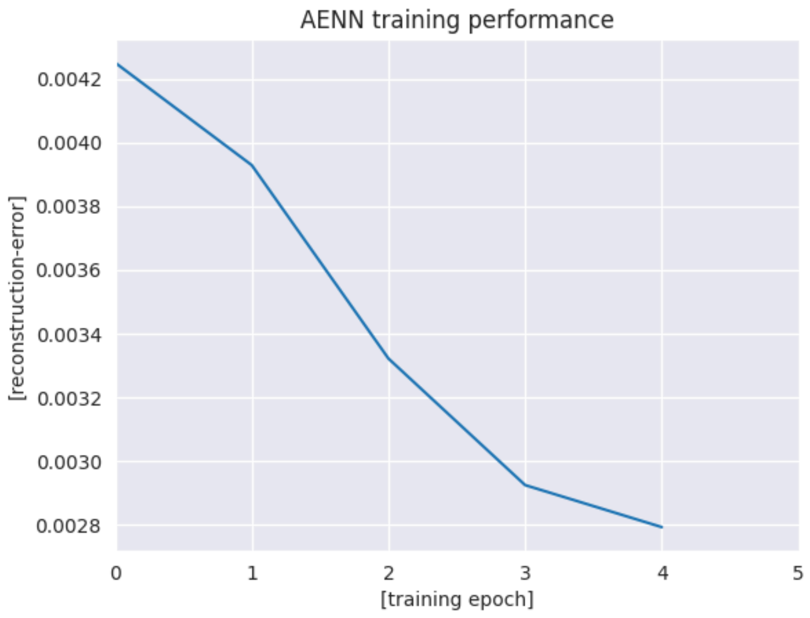


Figure 6. Original model reconstruction loss.

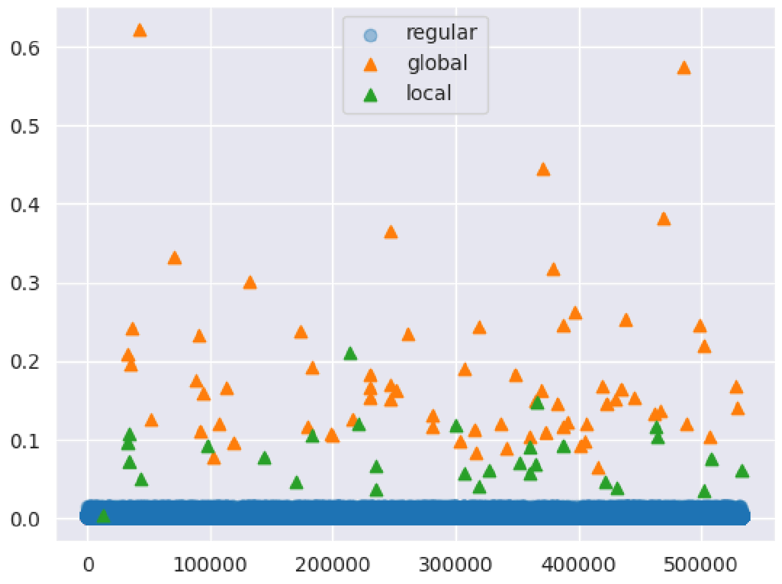


Figure 7. Original reconstruction loss for regular vs. the two outlier categories.

The measured loss by the end of training is quite low, ~.003. Visually, we can see the reconstruction loss is close to zero for the regular data, but for the global and local outliers it is between 0.1 and 0.6. From this we have a clear visual indicator of regular vs. outlier, and we can set a horizontal boundary line to distinguish the two categories.

We created a new model by changing the epoch hyperparameter from 5 to 50.

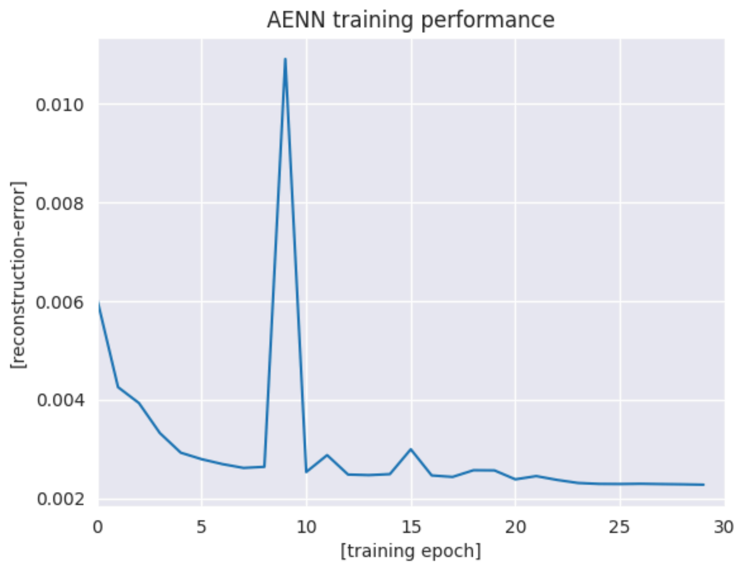


Figure 8. Epoch 50 model reconstruction error.

We can see that past a certain point, the information gain from succeeding epochs decreases. We see that past 5 epochs, any knowledge gain is marginal, and at 9 epochs the model becomes more inaccurate for a couple epochs. This may be due to the fact that the network did not fully capture the nuances of data and caused this spike. So essentially the network overfit the data temporarily in reconstructing the input data. When it encountered a slightly different but still non-anomalous entry the reconstruction error temporarily spiked because the network didn’t perfectly memorize that specific variation.

We also created a new model by changing the learning rate hyperparameter from 0.001 to 0.01.

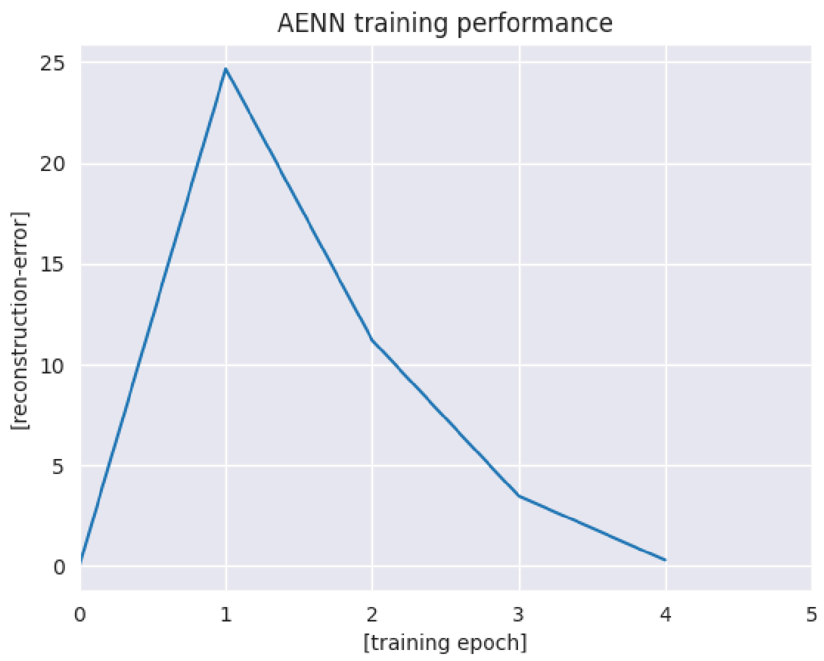


Figure 9. Learning rate .01 model reconstruction error.

We notice that the reconstruction error goes from a decimal previously, to now above one and reaching 25 at its peak. From this observation we can see that the model's training process became unstable. So the network adjusted the gradient aggressively. This caused the network to overshoot the optimal solution, preventing convergence.

While the network tried to capture the pattern in the data, it would most likely never find the most optimal solution due to the aggressive nature of a higher learning rate. It would end just keep oscillating around a local minimum failing to find the global minimum. We can conclude keeping the learning rate initialized at that lower value yields the best return.

Finally, we created a new model for shallow models instead of the deep learning multi-layered model.

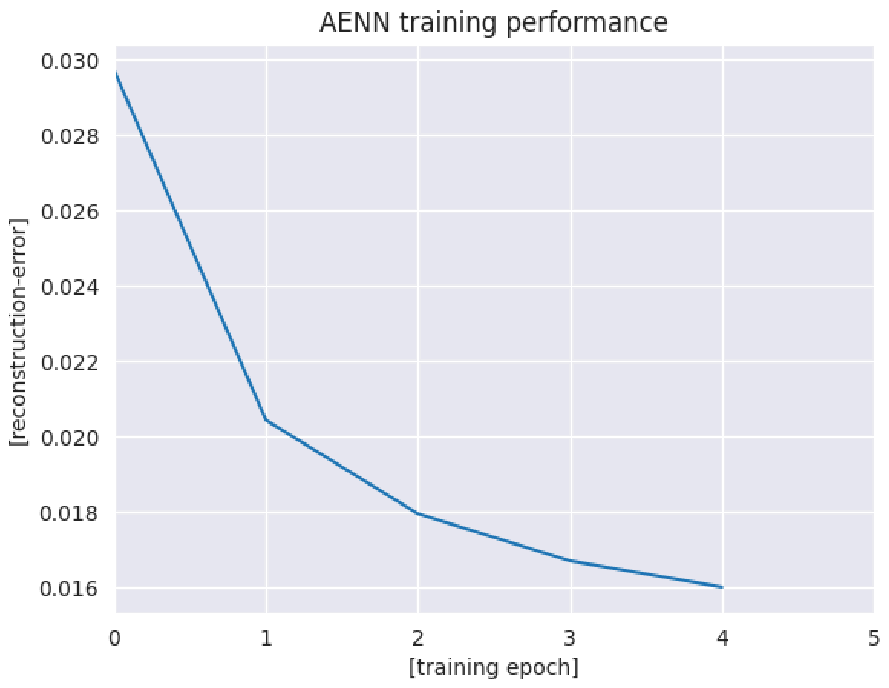


Figure 10. Shallow network model reconstruction error.

In this model, we go directly from the 618 transformed numeric input features, directly to 3 neurons in the latent space. We notice that over multiple epochs the error goes down, although the magnitude of error starts at .03 and approaches 0.016. When creating this shallow network we created a severe information bottleneck. We constructed a network with a drastic reduction in dimensionality causing the network to discard a substantial amount of information. The network was not able to capture the intricate patterns from the journals.

This new model never reaches the original small error of ~0.003. This model has a much more aggressive compression which can only identify significant anomalies. It would never be able to detect subtle anomalies due to the smaller latent space. This is why the original model performs much better.

The technical innovation of this model is the ability to use an autoencoder to analyze financial and accounting data, instead of the usual other machine learning methods people have done before. In this model, we used the datasets provided in the original model to show that this novel approach could be used in real data pulled from SAP or another ERP platform for a business.

Limitations

There are the model limitations, then also the limitations of real-life data, and the complexity of data.

The model itself may not be accurate enough to detect specific close data, so there might be false positives that cause delays via review of normal data. There could also be false negatives too, which run the risk of incorrect if not fraudulent data being missed and its consequences affecting a business.

Then there is the limitation of real-life data. Perhaps real-life data isn't accurate for the data points or is missing a data feature that if we did have, it could easily help identify outliers.

Real life data is complex and although deep learning is abstract, so are real life principles. No model is perfect, each model is generally a specialized model to meet a certain thing, but again no model in science is perfect.

Alternative Approaches

We want to introduce some alternative neural networks that can be used for anomaly detection in accounting/financial data. One approach that can be tested is the use of a Long Short-Term Memory (LSTM) Networks for fraudulent sequence detection. A LSTM network, trained on sequences of normal user behavior, can learn these patterns. When it encounters an unusual sequence, it can flag it as suspicious. A Convolutional Neural Network (CNN) can be used for financial statement analysis. The CNN can analyze a collection of financial statements like balance sheets or income statements and scan these documents. It can learn correlations between elements and points out any abnormalities or any unexpected deviations from the statements. An implementation of Attention Mechanisms may also be suited for this type of application. This can be used in conjunction or completely replace the one-hot encoding used in the paper. The attention mechanisms can help the model focus on the most relevant parts of the sequence when making anomaly detection decisions, potentially improving accuracy and clarity.

Applying neural networks to financial anomaly detection is ever changing. Experimentation and exploration is critical in finding new techniques to improve robustness and effectiveness for this task.

Future Applications

In this project we showcased the versatility and effectiveness of deep autoencoder networks. As we discussed previously, these types of networks work efficiently by compressing input data into a smaller, latent space representation and then reconstructing the original input from this representation. In doing so, for this application we were able to find anomalies from journal entries.

The application of deep autoencoder networks can be applied to numerous applications across various domains. They can be used for denoising data before the data is passed through another neural network. Training the autoencoder using corrupted input data (e.g, MRI scans, Music, CT scans) forces the autoencoder to learn the structure of the input data and learns to ignore the corruptions. Then when the data is "clean" it can be fed to another neural network which now only focuses on the relevant features. Autoencoders can also be applied to time series analysis. They can learn temporal dependencies and fill in missing data points in time series. One example of this is the prediction of traffic congestion. Unlike most networks that focus on traffic flow prediction, using an autoencoder you can take a snapshot of the road and use feature extraction to predict future traffic congestion. These are just a few examples of the practicability and power of deep autoencoder networks.

Ethics Statement

This is a model and no model is perfect, therefore applications of our different models should be used with discretion. One should also implement manual human analysis to check a few things. One to make sure it is ethical to even use the specific data you are planning to use. Two, if a conclusion is reached from the model analyzing data, following decisions should be made ethically. Third, in general this model shouldn't be used for harm or evil.

Acknowledgements

We would like to thank the original model authors and paper authors. We would also like to thank NJIT and the DS 677 course instructors for their support.

References

Company information | about SAP SE. (n.d.). https://www.sap.com/about/company.html

El Zein, Mohamad & Laz, Wissam & Laza, Malak & Wazzan, Taha & Kaakour, Ibrahim & Abu Adla, Yasmine & Baalbaki, Jad & Diab, Mohamad & Sabbah, Maher & Zantout, Rached. (2023). A Deep Learning Framework for Denoising MRI Images using Autoencoders. 10.1109/BioSMART58455.2023.10162068.

Najafi, S. M., Rezvani, S. J., & Wong, W. K. (2024). *Attention and autoencoder hybrid model for unsupervised online anomaly detection*. arXiv preprint arXiv:2401.03322. https://arxiv.org/abs/2401.03322

Padovano, A., & Cardamone, M. (2024). Towards human-AI collaboration in the competency-based curriculum development process: The case of industrial engineering and management education. Pattern Recognition, 145, 109935.

Sardar, S. (2018, March 9). *Some useful tables with header and item details*. SAP Community. https://community.sap.com/t5/enterprise-resource-planning-blog-posts-by-members/some-useful-tables-with-header-and-item-details/ba-p/13237198

Schreyer, Marco, et al. "Detection of anomalies in large scale accounting data using deep autoencoder networks." *arXiv preprint arXiv:1709.05254* (2017).

Schreyer, M., & Sattarov, T. (2018). *GitiHubi/Deepai: Detection of accounting anomalies using Deep Autoencoder Neural Networks - a lab we prepared for Nvidia's GPU Technology Conference 2018 that*

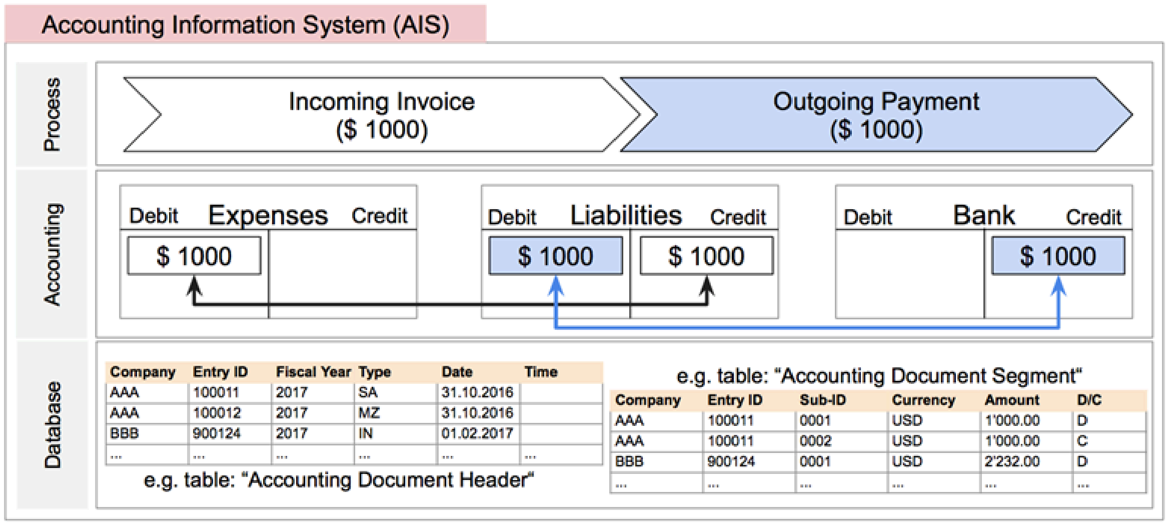
Will Walk You through the detection of accounting anomalies using deep autoencoder neural networks. the majority of the lab content is based on Jupyter Notebook, python and pytorch. GitHub. https://github.com/GitiHubi/deepAI

Srivastava, N. (n.d.). *Denoising autoencoders (DAE) - how to use neural networks to clean up your data?*. India's Top Provider of Advanced Cloud GPUs. https://www.e2enetworks.com/blog/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data

What is SAP?: Definition and meaning. SAP. (n.d.). https://www.sap.com/about/what-is-sap.html

Zhang, S., Yao, Y., Hu, J., Zhao, Y., Li, S., & Hu, J. (2019). Deep Autoencoder Neural Networks for Short-Term Traffic Congestion Prediction of Transportation Networks. *Sensors*, *19*(10), 2229. https://doi.org/10.3390/s19102229

Appendices



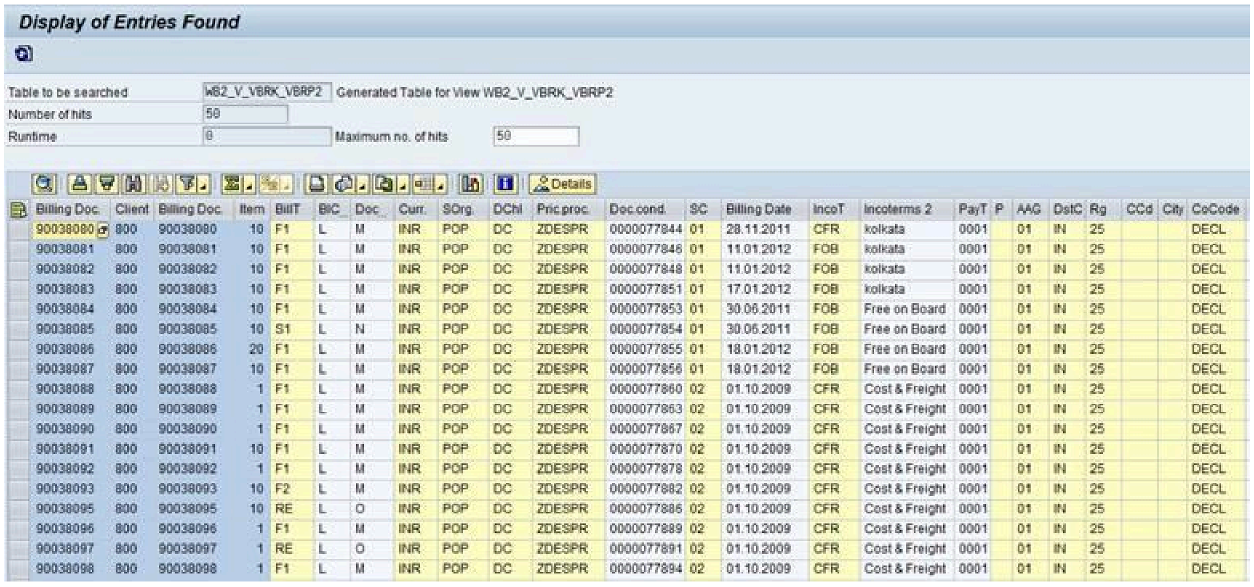
Company	Entry ID	Fiscal Year	Type	Date	Time
AAA	100011	2017	SA	31.10.2016	
AAA	100012	2017	MZ	31.10.2016	
BBB	900124	2017	IN	01.02.2017	
...	...	...	...	...	...

e.g. table: "Accounting Document Header"

e.g. table: "Accounting Document Segment"

Company	Entry ID	Sub-ID	Currency	Amount	D/C
AAA	100011	0001	USD	1'000.00	D
AAA	100011	0002	USD	1'000.00	C
BBB	900124	0001	USD	2'232.00	D
...	...	...	...	...	...

Figure 1. From the original paper, showing the database structure of an accounting transaction (Schreyer, Marco, et al.).



Display of Entries Found

Table to be searched	WB2_V_VBRK_VBRP2	Generated Table for View WB2_V_VBRK_VBRP2
Number of hits	58	
Runtime	0	Maximum no. of hits 50

Billing Doc.	Client	Billing Doc.	Item	BillT	BIC	Doc.	Curr.	SOrg.	DChl	Pric.proc.	Doc.cond.	SC	Billing Date	IncoT	Incoterms 2	PayT	P	AAG	DstC	Rg	CCd	City	CoCode
90038080	800	90038080	10	F1	L	M	INR	POP	DC	ZDESPR	0000077844	01	28.11.2011	CFR	kolkata	0001		01	IN	25			DECL
90038081	800	90038081	10	F1	L	M	INR	POP	DC	ZDESPR	0000077846	01	11.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038082	800	90038082	10	F1	L	M	INR	POP	DC	ZDESPR	0000077848	01	11.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038083	800	90038083	10	F1	L	M	INR	POP	DC	ZDESPR	0000077851	01	17.01.2012	FOB	kolkata	0001		01	IN	25			DECL
90038084	800	90038084	10	F1	L	M	INR	POP	DC	ZDESPR	0000077853	01	30.06.2011	FOB	Free on Board	0001		01	IN	25			DECL
90038085	800	90038085	10	S1	L	N	INR	POP	DC	ZDESPR	0000077854	01	30.06.2011	FOB	Free on Board	0001		01	IN	25			DECL
90038086	800	90038086	20	F1	L	M	INR	POP	DC	ZDESPR	0000077855	01	18.01.2012	FOB	Free on Board	0001		01	IN	25			DECL
90038087	800	90038087	10	F1	L	M	INR	POP	DC	ZDESPR	0000077856	01	18.01.2012	FOB	Free on Board	0001		01	IN	25			DECL
90038088	800	90038088	1	F1	L	M	INR	POP	DC	ZDESPR	0000077860	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038089	800	90038089	1	F1	L	M	INR	POP	DC	ZDESPR	0000077863	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038090	800	90038090	1	F1	L	M	INR	POP	DC	ZDESPR	0000077867	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038091	800	90038091	10	F1	L	M	INR	POP	DC	ZDESPR	0000077870	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038092	800	90038092	1	F1	L	M	INR	POP	DC	ZDESPR	0000077878	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038093	800	90038093	10	F2	L	M	INR	POP	DC	ZDESPR	0000077882	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038095	800	90038095	10	RE	L	O	INR	POP	DC	ZDESPR	0000077886	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038096	800	90038096	1	F1	L	M	INR	POP	DC	ZDESPR	0000077889	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038097	800	90038097	1	RE	L	O	INR	POP	DC	ZDESPR	0000077891	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL
90038098	800	90038098	1	F1	L	M	INR	POP	DC	ZDESPR	0000077894	02	01.10.2009	CFR	Cost & Freight	0001		01	IN	25			DECL

Figure 2. SAP sales billing data (Sardar, S., 2018).

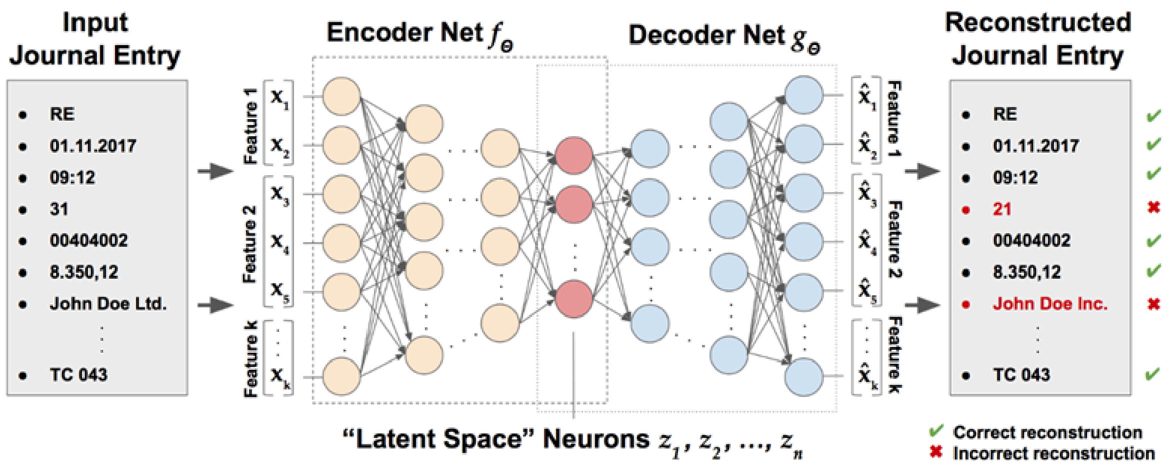


Figure 3. From the original paper, showing integration of accounting data with autoencoder models (Schreyer, Marco, et al.).

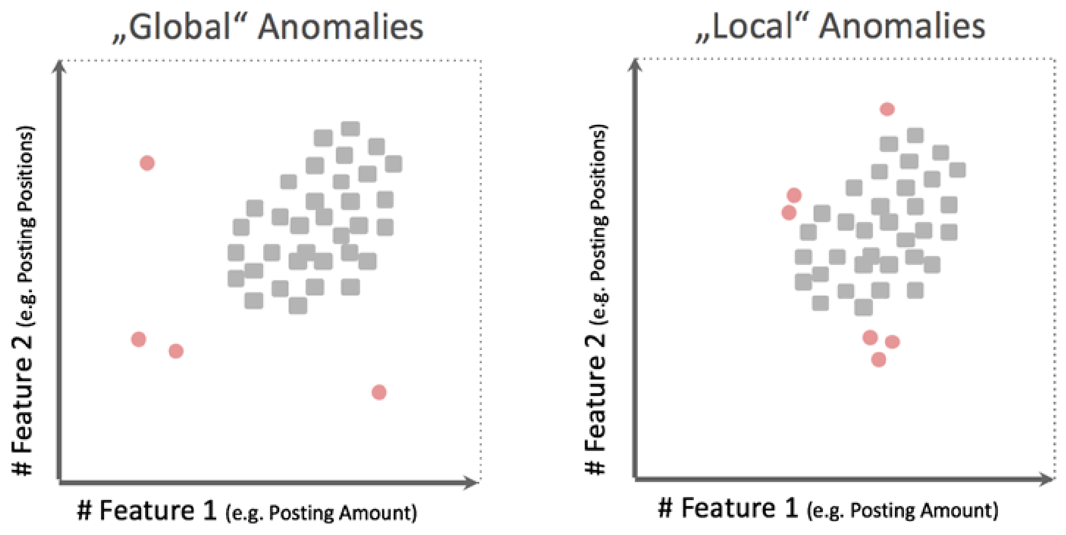
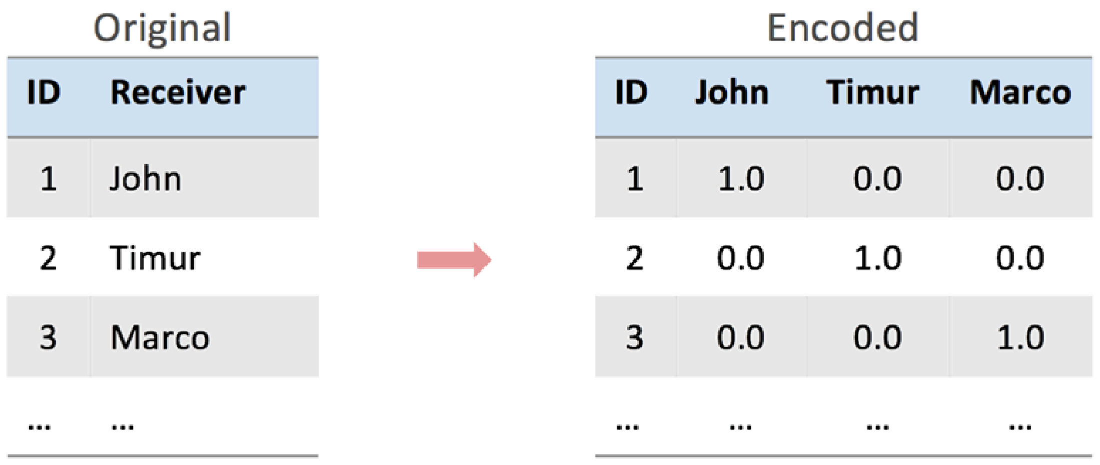


Figure 4. From the original paper, showing global vs. local anomaly (Schreyer, Marco, et al.).



Original

ID	Receiver
1	John
2	Timur
3	Marco
...	...

Encoded

ID	John	Timur	Marco
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
...	...	...	...

Figure 5. From the original paper, one-hot encoding of non-numeric data (Schreyer, Marco, et al.).

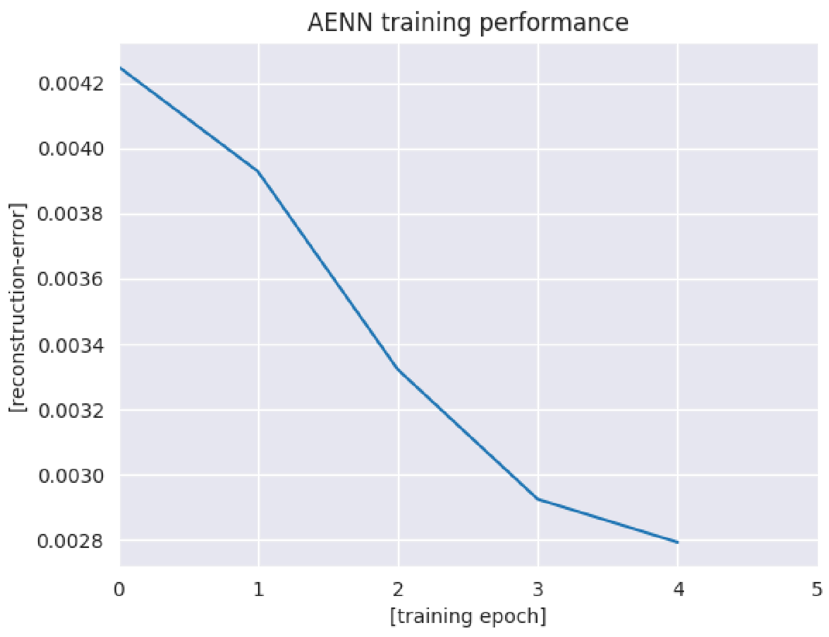


Figure 6. Original model reconstruction loss.

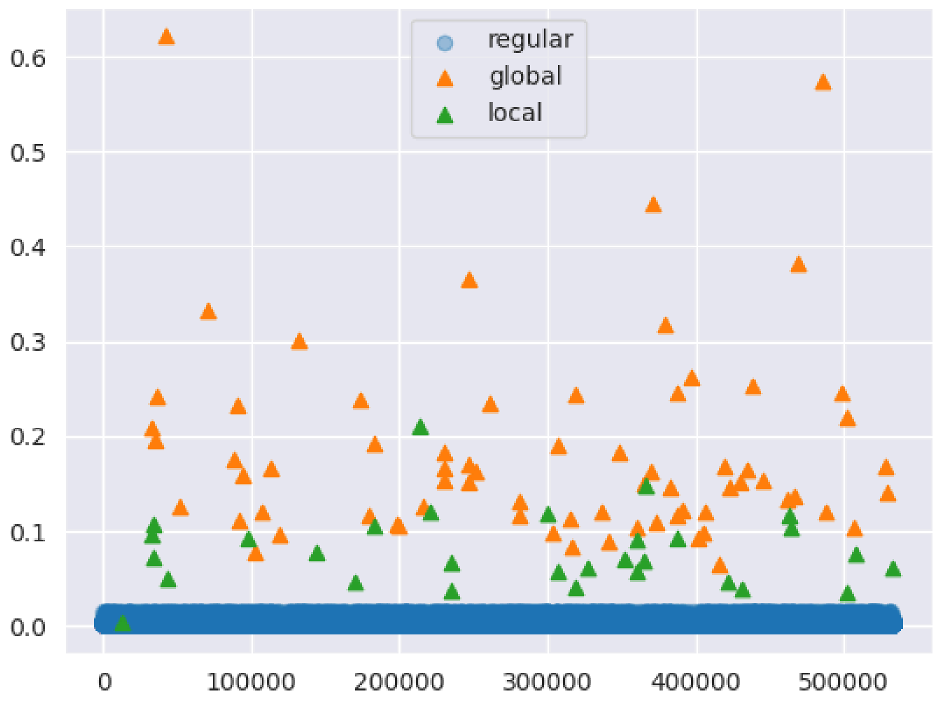


Figure 7. Original reconstruction loss for regular vs. the two outlier categories.

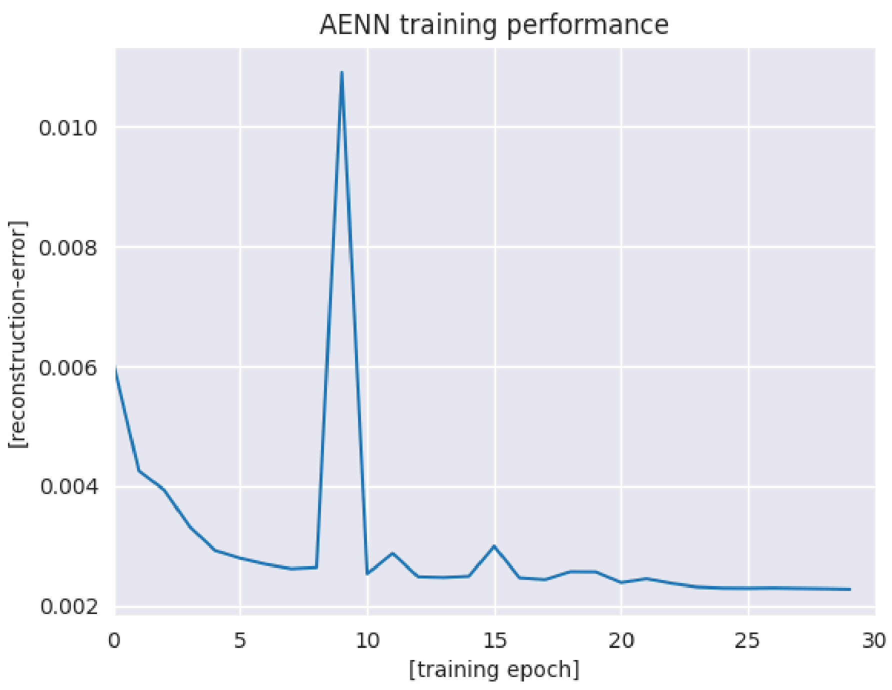


Figure 8. Epoch 50 model reconstruction error.

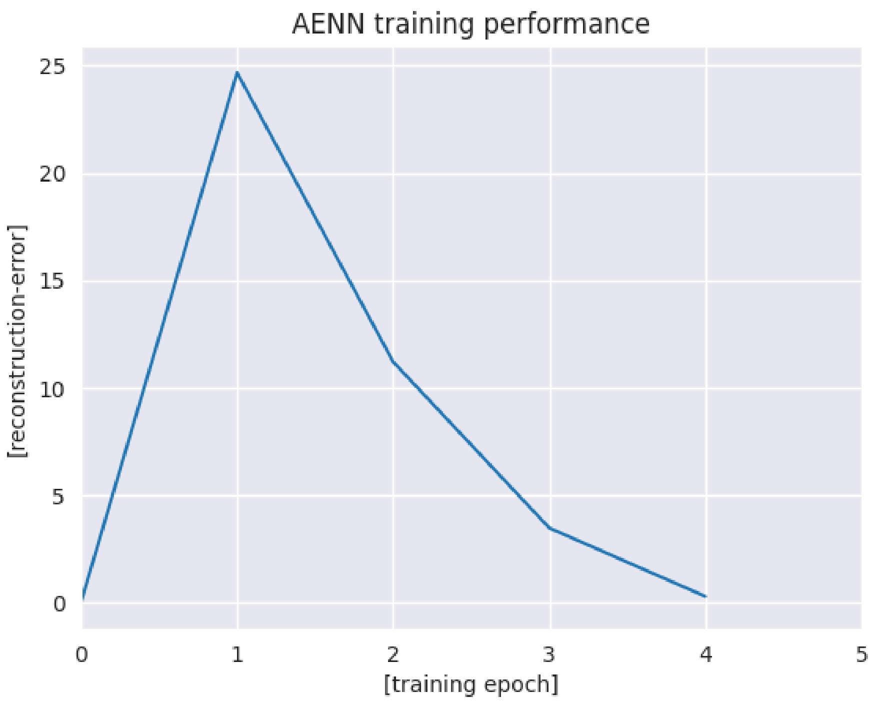


Figure 9. Learning rate .01 model reconstruction error.

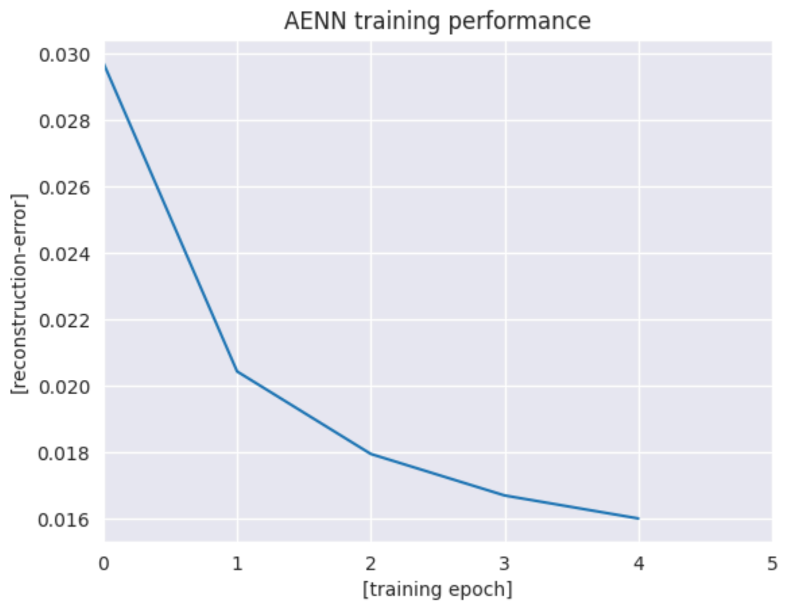


Figure 10. Shallow network model reconstruction error.