An Architectural Blueprint for a Multi-purpose Anomaly Detection on Data Streams

Christoph Augenstein[1], Norman Spangenberg[1] and Bogdan Franczyk[1,2]
[1]*University of Leipzig, Information Systems Institute, Grimmaische Straße 12, Leipzig, Germany*
[2]*Wroclaw University of Economics, ul. Komandorska 118/120, Wroclaw, Poland*

Keywords: Neuronal Nets, Deep Learning, Anomaly Detection, Architecture, Data Processing.

Abstract: Anomaly detection means a hypernym for all kinds of applications finding unusual patterns or not expected behaviour like identifying process patterns, network intrusions or identifying utterances with different meanings in texts. Out of different algorithms artificial neuronal nets, and deep learning approaches in particular, tend to perform best in detecting such anomalies. A current drawback is the amount of data needed to train such net-based models. Moreover, data streams make situation even more complex, as streams cannot be directly fed into a neuronal net and the challenge to produce stable model quality remains due to the nature of data streams to be potentially infinite. In this setting of data streams and deep learning-based anomaly detection we propose an architecture and present how to implement essential components in order to process raw input data into high quality information in a constant manner.

1 INTRODUCTION

Threats, anomalies and outliers are growing challenges for almost all companies. Since they occur in every corner of operations, like in their sensor and computer networks, or in business processes, companies have to be aware of intrusions and other abnormal or malicious activities on many different levels. According to the Oxford dictionary, anomalies are deviations from what is regarded as normal, or more specifically, describe events or measurements that are extraordinary, whether they are exceptional or not (Oxford Dictionaries, 2019). In data mining, the term describes data objects that are not compliant with the general behaviour or model of the data (Han et al., 2011). Anomalies occur in all areas of life, whether biology or business. Even for the latter, there is a multitude of use cases, that are dependent on the detection of anomalies in the behaviour of running operations. In this work we present an architecture with examples from two use cases from different areas: Operating Room (OR) Management and Network Intrusion Detection. In the OR area it is necessary for decision makers to know about the current states of running interventions and possible deviations in duration and surgical task order. For cybersecurity e.g. in corporate networks it is important to identify abnormal network traffic to prevent attacks like denial of services or hijacking of clients or servers. In addition, in all these – and other – cases of application, business operations increasingly become faster with higher demands on latest information about anomalies and deviations. Hence, data streams are essential to fulfil these requirements but are accompanied with additional challenges in real-time data processing. In these use cases, anomaly detection systems are used to detect such deviations, each with specific characteristics and peculiarities in terms of system components and detection methods.

The relevance and contribution of this work for information systems research are twofold: An architecture usable for a set of anomaly detection problems is important to generate synergies in terms of functional and technical components as well as in know-how to ease the realization of anomaly detection applications. This work postulates an architecture approach that is applicable in several use cases, providing a framework and set of components for multi-purpose deep learning (DL)-based anomaly detection applications. The corresponding research question is: *How should a generic architecture for DL-based anomaly detection be designed to support different use cases?*

The reminder of this paper is structured as follows: After this introduction, section 2 provides a brief overview of related work concerning DL-based anomaly detection systems as well as of approaches

Augenstein, C., Spangenberg, N. and Franczyk, B.
An Architectural Blueprint for a Multi-purpose Anomaly Detection on Data Streams.
DOI: 10.5220/0007760404700476
In *Proceedings of the 21st International Conference on Enterprise Information Systems (ICEIS 2019)*, pages 470-476
ISBN: 978-989-758-372-8

that try to place these approaches in architectures. Sections 3 and 4 contain the description of our solution approach, including an architecture model as well as implications for the realization of the approach. This is followed by an evaluation of the artefact as well as a discussion of the results. The paper closes with a conclusion featuring limitations, implications and an outlook on future research.

2 RELATED WORK

In accordance with the exploding advances in the machine learning and artificial intelligence area, a recent trend for anomaly detection is to use DL methods to model abnormal behaviour in data. The rise of powerful graphics hardware, with its ability to perform fast matrix multiplications is a technological advancement that brought many theoretical approaches into practice. Artificial neuronal networks (ANN) outperform other machine learning approaches in several fashions. For instance, even perceptrons, a simple type of ANN, is capable of identifying non-linear relationships in a dataset (Sadegh, 1993). Deeper nets or DL approaches respectively, i.e. nets that have many hidden layers are even more powerful (LeCun et al., 2015), but used to be impractical. Another interesting feature relevant for anomaly detection is the so called "course of dimensionality" (Goodfellow et al., 2016, p. 151), at which DL can tackle high dimensional data much better than other algorithms.

Specifically in the field of anomaly detection there are plenty of examples also using DL like (Zhou and Paffenroth, 2017) (deep auto encoders for feature engineering), (Revathi and Kumar, 2017) (anomalies in videos) or (Paula et al., 2016) (fraud detection). Specific to our use cases, several approaches for utilizing ANNs were recently presented to identify relevant information for OR management. (Bodenstedt et al., 2018) analyse multimodal sensor data of laparoscopic surgeries to identify their states and durations with the help of recurrent Convolutional Neural Networks (CNN). As well, (Twinanda et al., 2018) use CNN for surgical phase recognition based on video data.

In the intrusion detection systems area (IDS), the disadvantages of conventional machine learning approaches are the lack of automatic feature engineering, low detection rate, and incapability of detecting small mutations of existing attacks and zero-day attacks (Diro and Chilamkurti, 2018). Thus, (Elsherif, 2018) build a DL-based IDS system using Recurrent Neural Networks (RNN) and Long-Short term memories neural networks (LSTM). In addition, they use bidirectional techniques to avoid sequence dependencies by considering forward and backward order of request sequences. (Doshi et al., 2018) develop a DL pipeline that performs data collection, feature extraction, and binary classification for DDoS detection in IoT traffic. As well, they describe a high-level attack detection architecture for fog networks.

In strongly business process-oriented research areas like process mining, ANNs become also more popular for anomaly detection. Especially LSTMs are a focal point for research, e.g. (Tax et al., 2017) create generalizable predictive process monitoring applications for remaining cycle times and future states in running process instances with LSTMs. Others extend these methods with Natural Language Processing (NLP) methods and RNN to predict future events based on activity sequences (Hinkka et al., 2018) or in business processes without using explicit process models (Evermann et al., 2017). Finally, (Nolle et al., 2018) present an autoencoder approach that works without prior knowledge about considered processes and anomalies to classify business events.

Since, most of these approaches in different domains provide novel concepts for specific anomaly detection systems, but lack in describing the framework to embed their results in automated and productive environments. Indeed, there are also approaches that embed architectural decisions like (Larrinaga et al., 2018) for condition-based maintenance in cyber-physical systems or (Caselli et al., 2015) for describing a general architecture for IDS in industrial settings. But most of the work in the context of ANN's is about architecture of the ANN's themselves. By contrast, we also focus on architectural aspects in order to come up with a complete architectural solution for anomaly detection, including data ingestion, preprocessing and model building. For instance, (Papazoglou et al., 2015) provide much more details for a reference architecture that also includes anomaly detection. For our work we do not provide a reference architecture but one that is reusable in multiple similar scenarios in the context of anomaly detection. Comparable approaches often adopted the microservice paradigm as a basic principle in their architecture. Research in the mentioned use cases provides for example microservice-based architectures for an end-to-end IoT architecture that is usable for IDS (Datta and Bonnet, 2018). (Thramboulidis et al., 2018) created cyber-physical microservices, that are reusable in many settings in manufacturing. Additionally, (Carrasco et al., 2018) provide an overview of best practices and architectural hints while (Cerny et al., 2018) compare microservices with service-oriented architectures and also emphasize the possibility of independent deployments. Finally, (Steffens et al., 2018)

provide a solution for continuous integration & development (CI / CD) with microservices, which is also part of our solution approach.

3 ARCHITECTURE & INFRASTRUCTURE

In this section we describe a multi-purpose architecture for anomaly detection with focus on data streams. The main purpose for developing such an architecture is to process raw input data into high quality information in a constant manner in different contexts. As we especially focus on using ANNs and mostly DL, there is still a high demand for training data and thus for comfortable methods to process them. But the same is true for many machine learning approaches (e.g. random forest).

For this, we extracted necessary components out of several use cases and identified abstract patterns how to combine them. The use cases described in section 1 vary in terms of domain, data structure, amount and velocity of data and purpose of modelling. On a more abstract level, it can be stated that all these use cases can be approached using same or similar strategies and methodologies. Thus, our purpose is to build an architecture based on common components and infrastructure that replaces only specific elements for a certain use case. For a single use case this might seem too much effort but data processing pipelines are needed anyway and remain the same for each one in principle. Thus, a multi-purpose architecture allows to quickly adapt to new use cases. Based on the use cases and by using the separation of concerns paradigm (Parnas, 1972), which allows to decompose the overall architecture in building blocks, four essential generic building blocks were obtained: data (pre-) processing, model building, user interaction and cross-cutting (cf. figure 1).

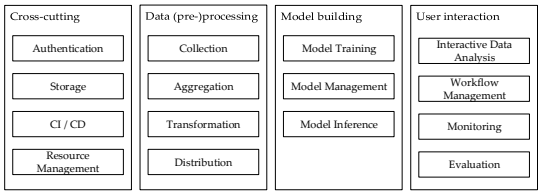

Figure 1: separation of concerns.

Cross-cutting concerns are necessary to support managing and maintaining other components (*Continuous Integration/Continuous Development*), gain access to the system (*Authentication*), provide *Storage* for user-specific and use case-specific data and allow for an abstraction of the underlying hardware and system resources (*resource management*). User interaction concerns are components, where users of the system interact with system components. Especially, this is about *Monitoring* training phases and about prediction *Evaluation* of the trained algorithms. Aside from that, the module for *Interactive Data Analysis* allows users to gain insights on small subsets of data for experimenting with algorithms and thus for evaluating (hyper-)parameters prior to training. Finally, the *Workflow Management* components allows users to develop and manage data processing pipelines based on available processing components. Data (pre-)processing concerns include components helping to prepare raw data in a way that they can be fed into selected algorithms. For instance, components such as load data from disk or interface with external sources (*Collection*), components that integrate data from various sources, handle null values, etc. (*Aggregation*) and components that provide *Transformations* (e.g. format conversions, decoding, cropping or augmentations) on (raw) data. The *Distribution* component acts as a kind of middleware and is responsible for ingesting data into other processing and model building components. In particular, distribution is responsible for handling data streams within the architecture and as such is also a special kind of collection component. The last and most important concern is about the model building itself. The three components *Model Training, Model Management* and *Model Inference* provide frameworks for training algorithms, allow to store, choose and update existing models and to put existing models into practice.

The architecture depicted in Fig. 2 provides a high level overview of the necessary components, already described as part of the different concerns and additionally essential data and control flows between them. There is no predefined order for using the components but basically data enters the system either by pushing or pulling it, passes through a variety of preprocessing steps before it is ingested into mass storage and is injected into the training engine. Trained algorithms are then checked into the model server or directly passed to the inference engine to deliver insights into the data. Things get interesting on a more use case and technology specific view.

Each component addresses a specific piece of functionality required by a use case pipeline. Collection, aggregation and transformation functionalities have to be designed for these specific requirements and thus have to be independent from each other. The only way they are able to interact is by pushing data into the distribution component and the mass storage respectively. Thus, our solution to handle data streams is to consequently build the architecture around streaming functionality (cf. the Kappa archi-

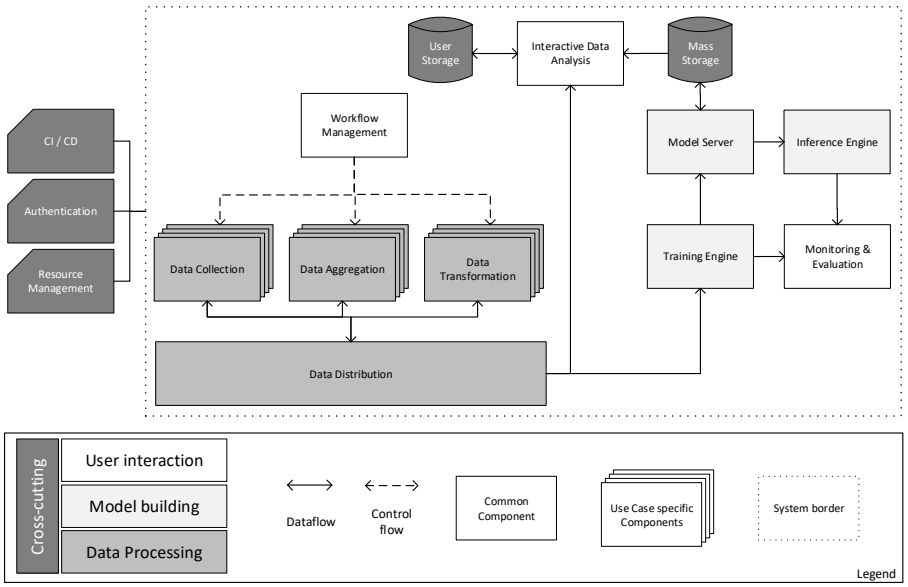

Figure 2: architectural blueprint.

tecture (Kreps, 2014)). Unfortunately, there are some drawbacks regarding ANN's, which we will discuss in section 4.2. To enforce independence of the components we make use of the microservice paradigm and package implemented functionality into distinct services with a common interface. In our architecture we use a JSON-based intermediate exchange format for this and to annotate data with metadata. JSON is used due to its widespread dissemination and most programming languages already have parsers and other utilities to easily work with it. Aside from an architectural approach, the underlying infrastructure as well as efficient possibilities of utilizing restricted resources is as important as developing necessary components. To enable a microservice-based approach we restrict implementation of the architecture to use a virtualization based on Docker and Kubernetes. These infrastructural components are necessary (enforcing and leveraging microservices) but also provide benefits. In practice, we are able to dynamically assign resources to dedicated components, e.g. for resource-intensive transformation steps, while other components can spare remaining resources. Especially, Kubernetes allows us to assign specialized hardware like graphics to dedicated components (e.g. model training). Moreover, we can benefit from container registries to manage already existing preprocessing components. To sum up, isolated components on the one hand and a flexible infrastructure allowing to quickly adapt to resource demands on the other, enable the realization of dynamic pipelines.

4 FROM DATA TO MODEL TO INFORMATION

4.1 Data Processing Pipelines

By the use of these architectural components, we describe in this section how to build processing pipelines and hence how to apply the proposed architecture. Processing pipelines are built with selected frameworks. Currently, the implementation supports Apache Spark and Apache Flink in the Kubernetes environment. For both, one master and some worker nodes were deployed, whereby the number of workers is dynamically adjustable according to the requirements. Both frameworks can be used by implemented clients or within the interactive, batch-oriented data analysis component or their dedicated streaming functionalities. Apache Kafka in conjunction with Apache Zookeeper is also deployed in Kubernetes and serves as the primary streaming component. As such, it is also focal element in every processing pipeline and helps to move data through a pipeline with a publish-subscribe pattern. Each collection, aggregation and transformation component is derived from a basic Docker image that incorporates client libraries and configurations for Spark, Flink or Kafka Streams. For instance, in order to build a Flink-based transformation component the corresponding base image is taken and use case specific code is added prior to the build process. The resulting image or container respectively can then be deployed and is

added to the registry for further usage.

Aside from programmatically defining data processing pipelines, e.g. by implementing fixed routes with Kafka, the workflow management component in the user interaction concern is a viable alternative for deploying complete processing pipelines. At present, Apache Airflow is evaluated for the management component, but is subject to further investigations and discussions. The idea behind is that, in this case, pipelines are configured by declaring a workflow on an abstract level. Airflow uses directed acyclic graphs for pipeline definitions and makes use of a special Kubernetes operator that is capable of starting Docker images with a specified command. Hence, we can use the same components as described before but the exchange of data happens based on mass storage, e.g. HDFS or databases. Such a pipeline definition also offers the ability to execute the pipeline based on a schedule. The drawback with workflow management is the current lack of support for streaming functionality, i.e. data streams have to be continually split into chunks and ingested into storage.

4.2 Model Building & Management

Having defined and operated processing pipelines, use case specific data is well prepared to train the desired models. For working with ANNs our implementation at present provides Tensorflow, Keras, TFLearn and to a certain extend PyTorch. Additionally, we also use the implemented machine learning algorithms from Spark and Flink. Development of net architectures and parameterisation of algorithms like random forests is also supported in the interactive data analysis component so that running training jobs can be tested in advance for error pruning. By means of available data, a training job is triggered e.g. every night, depending on size of dataset and complexity of the desired model. When it is trained we deploy the model for the specific uses case, e.g. in form of a container (Spark or Flink) or with the help of the inference engine (Tensorflow). As a result, we currently obtain a use case specific model for anomaly detection, i.e. we develop distinct models based on concrete use case requirements. In future research this is also subject to discussion (cf. section 6).

From a technical point of view, handling of data streams is easy if Spark- or Flink-based models are used because they have distinct APIs for data streams. But, for the moment, to train ANNs, data streams have to be broken into chunks and be fed into with (micro-)batches. Major problems arise from typical big data characteristics (cf. (Augenstein et al., 2017)) whereas data streams can be regarded as potentially indefinitely and thus represent big data at high velocity. Although there are approaches to tackle big data related problems, e.g. concept drift, class imbalance or non-linearity can have a lasting effect on model quality. Thus, we use for the approach continuous monitoring and define thresholds (e.g. for accuracy) to spot decreasing model quality by leveraging the monitoring and evaluation components. With this, we have a pragmatic possibility at hand to gain information about the quality of the models and to control already trained models in order to adapt the model to new or changing data.

5 EVALUATION STRATEGY

The evaluation of the proposed approach is an ongoing work. For a proper evaluation, a twofold strategy is planned, which follows the principles of design science research artefact evaluation (Peffers et al., 2012; Venable et al., 2016). Hence, technical experiments will be conducted to evaluate the approach's performance in addition to more formative evaluation methods. As a first step, the architecture model is mapped on the two use cases described in section 1 to implement the components necessary to fulfil their functional needs.

For the planning and optimization tasks in Online Surgery Scheduling of OR management, the architecture approach described in section 3 is utilized as follows. In this use case, functionalities for data distribution and storage layer are needed, which are realized by components of *Data Collection, Data Transformation and Data Distribution*. The previously used Complex Event Processing-based (cf. (Spangenberg et al., 2017a) component for anomaly detection is replaced by the *Model Server* and the *Training* respectively *Inference Engine*. These building blocks allow a highly automatized approach relying on ANNs (see (Spangenberg et al., 2017b)) which reduces the need of expert knowledge for defining a hard ruleset to identify current surgical phases as well as remaining intervention times. By using Spark, based on its Kubernetes version, and an implementation of a Multilayer Perceptron Regression algorithm (Spangenberg et al., 2017b), a processing pipeline and a DL-model training approach for anomaly detection in surgical interventions was realized. The inference component uses Kafka Streams for loading the persisted DL-model from file system and processes each detected surgical phase to infer a remaining intervention time prediction based on the model. An ongoing challenge at the moment is to find an appropriate cycle for model recomputation to avoid streaming-based

problems (cf. section 4.2).

Besides, we utilized an existing IDS approach (Radford et al., 2018) that uses a LSTM to detect abnormal activities and intruders in network traffic. The necessary transformation functionality that extracts features from the generated network traffic maps to the *Data Transformation* component of the architecture blueprint. With the help of the *Interactive Data Analysis* component it is possible to successively identify details of the dataset and the problem. The resulting Tensorflow model of the DL application is mapped to the *Training Engine* to train a valid classification model for anomaly detection. This model is finally utilized by the *Inference Engine* to realize anomaly detection on network data streams in a real-time fashion. This approach uses Word2Vec, an algorithm used in NLP to create word embeddings to classify text anomalies, as part of the preprocessing.

Based on these implementations, public datasets for IDS (Sharafaldin et al., 2018) and artificial datasets for OR management (Spangenberg et al., 2017a) have been utilized and scaled to benchmark the approach in different settings and loads. Further, formative methods will be utilized that focus on the outcome of the artefact for the described problems. The resulting artefacts and implementations will be demonstrated to domain experts to assess diverse criteria for Information Systems artefacts (Prat et al., 2015) like operational feasibility, usefulness or adaptability on other contexts. The reason for this is to identify potential consequences for further anomaly detection use cases, e.g. to derive additional components as architectures building blocks.

6 CONCLUSION

In this paper we presented a novel solution approach for a multi-purpose architecture for anomaly detection. The novelty of this approach lies in a holistic view on the overall processing of data beginning with pushing or pulling it from source and ending with a common representation being fed into ANNs or machine learning based algorithms. A general applicability of this architectural approach is given because of the restriction to anomaly detection and the focus on finding a common representation of data over several use cases. For future work our focus will be on an exhaustive approach that also encompasses the model learning, i.e. we also want to produce a neuronal net architecture that is capable of uniformly handling data from different uses cases. For this, we choose multi-task learning (cf. (Caruana, 1997)) as a starting point. It has already been shown that ANNs trained with different data for the same reason tend to provide even better results than training multiple nets (e.g. (Yang et al., 2018) (image captions), (Kong et al., 2018) (sparse data), (Luo et al., 2018) (multiple domains, text processing). For instance, transferred to our scenario, we have multiple classification problems (tasks, one per use case) and train with different datasets in order to receive a good classification per use case. Being able to apply such a multi-task learner, we are still confronted with the draw backs of (big) data streams. A major question arises from the problems of continuous model updates. Updates usually force (re-)training of the underlying model which might not be applicable at all hours, so the question remains how to adapt to fast changing data.

ACKNOWLEDGMENTS

This research was funded by the European Social Fond ESF and the Federal State of Saxony, Germany (100341518).

REFERENCES

Augenstein, C., Spangenberg, N., and Franczyk, B. (2017). Applying machine learning to big data streams : An overview of challenges. In *IEEE 4th International Conference on Soft Computing and Machine Intelligence (ISCMI 2017)*, pages 25–29.

Bodenstedt, S., Wagner, M., Mündermann, L., Kenngott, H., Müller-Stich, B., Mees, S. T., Weitz, J., and Speidel, S. (2018). Prediction of laparoscopic procedure duration using unlabeled, multimodal sensor data. *arXiv preprint arXiv:1811.03384*.

Carrasco, A., van Bladel, B., and Demeyer, S. (2018). Migrating towards microservices: migration and architecture smells. In Ouni, A., Kessentini, M., and Cinnéide, M. Ó., editors, *Proceedings of the 2nd International Workshop on Refactoring - IWoR 2018*, pages 1–6, New York, New York, USA. ACM Press.

Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

Caselli, M., Zambon, E., and Kargl, F. (2015). Sequence-aware intrusion detection in industrial control systems. *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*.

Cerny, T., Donahoo, M. J., and Trnka, M. (2018). Contextual understanding of microservice architecture. *ACM SIGAPP Applied Computing Review*, 17(4):29–45.

Datta, S. K. and Bonnet, C. (2018). Next-generation, data centric and end-to-end iot architecture based on microservices. In *IEEE International Conference on Consumer Electronics*, pages 206–212. IEEE.

Diro, A. A. and Chilamkurti, N. (2018). Deep learning: The frontier for distributed attack detection in fog-to-

things computing. *IEEE Communications Magazine*, 56(2):169–175.

Doshi, R., Apthorpe, N., and Feamster, N. (2018). Machine learning ddos detection for consumer internet of things devices. pages 29–35.

Elsherif, A. (2018). Automatic intrusion detection system using deep recurrent neural network paradigm. *Journal of Information Security and Cybercrimes Research (JISCR)*, 1(1).

Evermann, J., Rehse, J.-R., and Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*, 100:129–140.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT Press, Cambridge, Massachusetts and London, England.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Hinkka, M., Lehto, T., Heljanko, K., and Jung, A. (2018). Classifying process instances using recurrent neural networks. *arXiv preprint arXiv:1809.05896*.

Kong, Y., Shao, M., Li, K., and Fu, Y. (2018). Probabilistic low-rank multitask learning. *IEEE transactions on neural networks and learning systems*, 29(3):670–680.

Kreps, J. (2014). Questioning the lambda architecture: The lambda architecture has its merits, but alternatives are worth exploring.

Larrinaga, F., Fernandez, J., Zugasti, E., Zurutuza, U., Anasagasti, M., and Mondragon, M. (2018). Implementation of a reference architecture for cyber physical systems to support condition based maintenance. *5th International Conference on Control, Decision and Information Technologies*, pages 773–778.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Luo, Y., Wen, Y., and Tao, D. (2018). Heterogeneous multitask metric learning across multiple domains. *IEEE transactions on neural networks and learning systems*, 29(9):4051–4064.

Nolle, T., Luettgen, S., Seeliger, A., and Mühlhäuser, M. (2018). Analyzing business process anomalies using autoencoders. *Machine Learning*, pages 1–19.

Oxford Dictionaries (2019). anomaly.

Papazoglou, M., van den Heuvel, W.-J., and Mascolo, J. (2015). Reference architecture and knowledge-based structures for smart manufacturing networks. *IEEE Software*.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058.

Paula, E., Ladeira, M., Carvalho, R., and Marzagao, T. (2016). Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *15th IEEE International Conference on Machine Learning and Applications*, pages 954–960. IEEE.

Peffers, K., Rothenberger, M., Tuunanen, T., and Vaezi, R. (2012). Design science research evaluation. *Design Science Research in Information Systems. Advances in Theory and Practice*, pages 398–410.

Prat, N., Comyn-Wattiau, I., and Akoka, J. (2015). A taxonomy of evaluation methods for information systems artifacts. *Journal of Management Information Systems*, 32(3):229–267.

Radford, B. J., Richardson, B. D., and Davis, S. E. (2018). Sequence aggregation rules for anomaly detection in computer network traffic. *CoRR*, abs/1805.03735.

Revathi, A. R. and Kumar, D. (2017). An efficient system for anomaly detection using deep learning classifier. *Signal, Image and Video Processing*, 11(2):291–299.

Sadegh, N. (1993). A perceptron network for functional identification and control of nonlinear systems. *IEEE transactions on neural networks*, 4(6):982–988.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116.

Spangenberg, N., Augenstein, C., Franczyk, B., Wagner, M., Apitz, M., and Kenngott, H. (2017a). Method for intra-surgical phase detection by using real-time medical device data. *IEEE 30th International Symposium on Computer-Based Medical Systems*, pages 254–259.

Spangenberg, N., Wilke, M., Augenstein, C., and Franczyk, B. (2017b). A big data architecture for intra-surgical remaining time predictions. *Procedia Computer Science*, 113:310–317.

Steffens, A., Lichter, H., and Döring, J. S. (2018). Designing a next-generation continuous software delivery system: Concepts and architecture. *2018 IEEE 4th International Workshop on Rapid Continuous Software Engineering*, pages 1–7.

Tax, N., Verenich, I., La Rosa, M., and Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. *International Conference on Advanced Information Systems Engineering*.

Thramboulidis, K., Vachtsevanou, D. C., and Solanos, A. (2018). Cyber-physical microservices: An iot-based framework for manufacturing systems. In *IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 232–239. IEEE.

Twinanda, A. P., Yengera, G., Mutter, D., Marescaux, J., and Padoy, N. (2018). Rsdnet: Learning to predict remaining surgery duration from laparoscopic videos without manual annotations. *arXiv preprint arXiv:1802.03243*.

Venable, J., Pries-Heje, J., and Baskerville, R. (2016). Feds: a framework for evaluation in design science research. *European Jour. of Information Systems*, 25(1):77–89.

Yang, M., Zhao, W., Xu, W., Feng, Y., Zhao, Z., Chen, X., and Lei, K. (2018). Multitask learning for cross-domain image captioning. *IEEE Transactions on Multimedia*, page 1.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In Matwin, S., Yu, S., and Farooq, F., editors, *KDD2017*, pages 665–674, New York, NY. ACM.