Concept of Easy-to-use Versatile Artificial Intelligence in Industrial Small & Medium-sized Enterprises

Blixt Hansen, Emil; Iftikhar, Nadeem; Bøgh, Simon

Published in:
Procedia Manufacturing

DOI (link to publication from Publisher):
10.1016/j.promfg.2020.10.161

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 51 (2020) 1146–1152

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021)
15-18 June 2021, Athens, Greece.

Concept of easy-to-use versatile artificial intelligence in industrial small & medium-sized enterprises

Emil Blixt Hansen[a,c,*], Nadeem Iftikhar[b], Simon Bøgh[c]

[a]*Technology Education and Sustainable Growth, University College of Northern Denmark, Aalborg, Denmark*
[b]*Department of Computer Science, University College of Northern Denmark, Aalborg, Denmark*
[c]*Department of Materials and Production, Aalborg University, Denmark*

Abstract

In this paper, the concept of what we call AI-Box is presented. This concept is targeting small and medium-sized enterprises within the manufacturing industry sector. The AI-Box aims to bring technologies from Industry 4.0 to them, with the use of easy-to-use and versatile implementation. Preliminary experiments have been conducted at Aalborg University and at an industrial partner to solve vision tasks, which would be too expensive with conventional vision techniques. Moreover, three different convolutional neural networks were tested to find the best-suited architecture. The three networks tested were the simple AlexNet, the complex ResNeXt, and small and complex SqueezeNet. Our results show that it is possible to solve the classification problem in a few epochs. Furthermore, with the use of augmented data, the performance can be improved. Our preliminary results also showed that the simpler convolutional neural network architecture from AlexNet yields a better result when classifying simple data.

Keywords: Artificial Intelligence; Machine Learning; Concept; Small and Medium Sized Enterprises; Versatile

1. Introduction

The topic of *artificial intelligence*, or its commonly used abbreviation, AI, is used in more and more sectors and is only expected to grow [1, 2]. More specifically, the use of the AI sub-category *machine learning* is showing tremendous potential with the different breakthroughs throughout the 2010s in, e.g. image data [3], and can now reach and outperform humans in games [4]. With quickly advances in AI and machine learning, the manufacturing industry is also looking towards these technologies. AI, along with other new technologies such as Big Data and 3D printing, is categorised under the name Industry 4.0 [5]. Large enterprises have already begun using these technologies within their products and production [6, 7]. Moreover, the use of machine learning has also been exploited in welding and robotic context [8, 9]. Most of the research and developed solution on the market is concerning big enterprises or research, and there is a lack of focus on the small- and medium-sized enterprises (SMEs). The lack of focus on SMEs is also evident in a study from 2018, which showed that the current maturity assessment for Industry 4.0 is not suitable for SMEs [10]. Gartner's steps of analytics can be used to reflect on how advanced analytic capabilities a company has [11]. It includes four steps: descriptive, diagnostic, predictive, and prescriptive, where SMEs generally is placed on the first step *descriptive*. The descriptive step specifies that the system can describe what the problem is but nothing else. The step of *predictive* analytics is a step that would benefit the industry; hence they would be able to predict equipment failure and maintenance. With Industry 4.0, different companies have created *smart devices* that enables manufactures to monitor their production, OEE, and live data. Examples of such devices are the Factbird by Blackbird [12] and M-Box by Monitor-Box [13]. Within research, the Danish Institute of Technology has created what they call Vision Box, which brings 2D and 3D quality inspections to manufactures with the use of deep learning [14]. Research exists in the context of visual inspection of production to find faults [15]

* Corresponding author. Tel.: +45 42494707
E-mail address: ebh@mp.aau.dk (Emil Blixt Hansen).

2351-9789
10.1016/j.promfg.2020.10.161

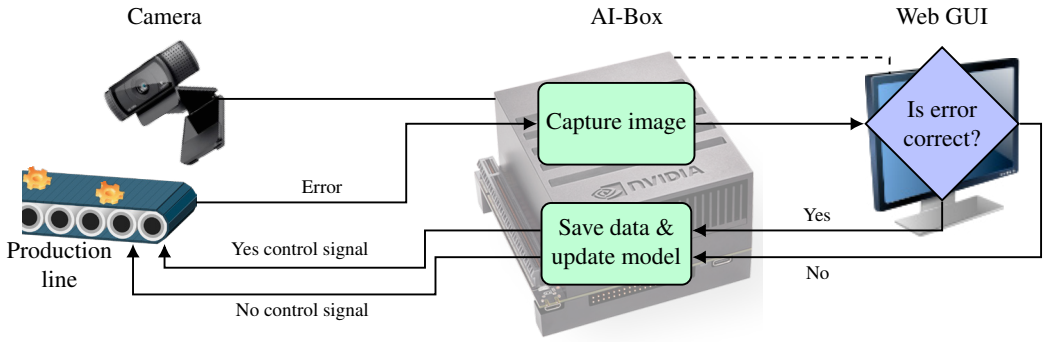

Fig. 1: A use case example of the AI-Box. The line between the camera and the AI-Box represent a physical USB connection between the two. The dashed line between the monitor and the AI-Box illustrates a wireless connection between the GUI and AI-Box.

and distribute the computation in between other edge nodes, also known as fog computing [16].

In this paper, we present the concept of what is called AI-Box. The AI-Box enables manufacturing SMEs to utilise the Industry 4.0 technologies of machine learning and IoT without them needing software engineers for setup and maintenance. Firstly in Section 2 three examples of use cases are described. In Section 3, the concept of the AI-Box is described and in Section 4 preliminary experiments are described. Finally, in Section 5, the conclusion and future work for the project are described.

2. Use case description for SMEs

Before introducing the underlying architecture, it is essential to have an understanding of what kinds of problems the AI-Box aims to solve. This section contains three hypothetical cases describing different use cases of the AI-Box at industrial SMEs.

Use case 1: False alarm

A company is producing plastic gears, and after they come out of the mould, sometimes there is leftover material on the cut teeth. This leftover material is not crucial to be removed at this stage. Still, the production triggers a false alarm, hence the production is temporarily stopped until an operator acknowledges the false error. Here the AI-Box will be set up with a camera pointing at the place from where the alarm was triggered and is connected to the relevant PLC out- and inputs. Each time the alarm occurs a picture is taken and is labelled depending on the input from the operator. When enough data is collected, the AI-Box starts to train on the data, and when it has trained enough, it can take over control and operate the PLC inputs by itself. This type of operation is illustrated in Fig. 1.

Use case 2: An audible error

At a production line, an experienced operator can hear when a tool should be changed but is unable to see it. The AI-Box is connected to a microphone that would monitor the utilisation. The operator will record the tool sound under normal operation and under a situation where the tool is in a condition where it should be changed. The AI-Box will learn this signature and will create an alert when the tool is about to be worn down. This will help inexperienced operators with performing tool change and helper monitor the machine.

Use case 3: Unknown error occurs

At a production line sometimes an operation fails, but no alarms are created. However, it is believed that it can be measured through vibration, and thus an accelerator sensor is attached to the place where the operation fails. The AI-Box is then set into an *outlier* detection mode, and here it will then create an alarm if the vibration is at an abnormal level.

These three use cases are examples of the different aspects of the AI-Box, but not limited to. An input could also be temperature, magnetic or video feeds. The AI-Box can, therefore, be summed up to the following key features:

- Simple to deploy and use
- Various built-in machine learning models
- Read and write PLC signals
- Handle different types of measurement data

Together, these features will enable SMEs to find and solve problems in there production, which they are unable to solve with traditional means.

3. Concept framework & architecture

The AI-Box concept consists of a complex architecture involving hardware and software. To give a better overview, the section is split up in hardware, software, and the internal system architecture.

3.1. Hardware

The main hardware for the AI-Box is the computer that trains and deploys the network for any given task. It should be balanced between cost, performance, and deployability, where the latter means it should be small, lightweight and not depending on a vast amount of peripherals. An NVIDIA Jetson AGX Xavier was chosen as the main hardware component [17] since it is small, not too expensive and it has a dedicated GPU with CUDA cores and thus enabling GPU hardware to accelerate deep learning performance. Even though the recommendation is that the Xavier is meant for deployment only, it can still be used to train on. Webcam and microphone are connected directly through a USB port while sensors such as accelerometer and temperature sensor, is connected through the GPIO pins on the Xavier.

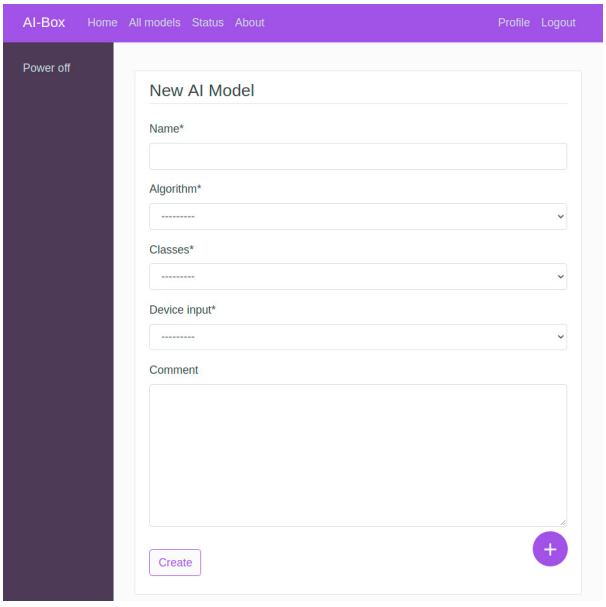

Fig. 2: The setup screen of a new model, where the algorithm is chosen, the number of classes there are, and the input device are chosen too.

3.2. Software

Rapid development was deemed a paramount aspect of creating the AI-Box. Therefore, the primary programming language used was Python 3.6 with Tensorflow 2.0 as the deep learning framework. Moreover, as one of the requirement was the reduced use of peripherals, a web-based interface was chosen as the best solution, here Django 3.0 was chosen. Through the web-interface of the AI-Box, the operator will make the initial setup for the problem at hand, which include specifying if, e.g. it is a camera that should be used (see Fig. 2). This setup screen enables the operator to set up the right deep learning architecture without he/she knowing it. Moreover, the classes that can be selected is a yes/no scenario or an outlier instance. When the model has been created, a running view is displayed

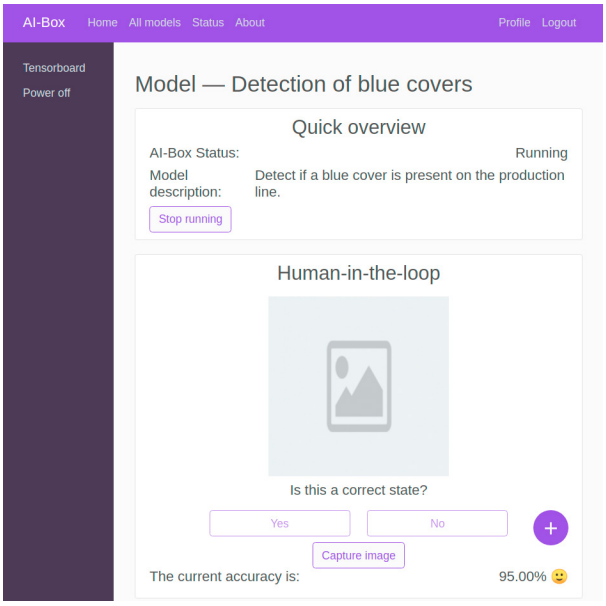

Fig. 3: The running screen shows the status and current accuracy. Moreover, when a new sample is captured, it will ask the operator to label it.

(see Fig. 3) to the operator. Here, the operator can see relevant information, such as current accuracy. Also, when a new sample is taken, and the AI-Box has not trained enough, or the model is not confident enough of the class, the operator is asked. The operator's decision then labels the sample and is saved in the sample database. The database storing all of the data has chosen to be a Zarr database [18]. Moreover, the web-interface also contains a view of all created models along with details view, such that the operator can view and change parameters of the model.

3.3. Architecture

The internal architecture of the model is firstly based on the web interface architecture *model-view-controller* (MVC) [19]. The MVC architecture is native to Django and is thus automatically implemented. The complexity of the AI-Box requires an additional architecture design to handle the state of the device, the deep learning models, and its associated data. This additional architecture is loosely based on the *layered pattern* architecture, where each layer has a different level of abstraction and serves the layers above and below it. In Fig. 4 the system can be seen. The top layer is the *GUI* where the operator interacts with the AI-Box, depending on what is displayed and interacted with, the information is pulled from the SQLite3 database from Django. When the operator changes the state of the device (e.g. starting a model) the *global state handler* ensures to initialise the correct parameters depending on the model information stored in the SQLite3 database. The *model handler* is a singleton python class that handles initialisation of the *loop controller* and the *sensor handler*, moreover, it also handles their intercommunication. The loop controller is where

the deep learning is handled. The dataset is read from the Zarr database and is handling the learning of the model and when to stop to avoid overfitting. It also initialises the actual deep learning model by calling the respected *ML model* class. The *sensor handler* controls the setup, retrieves the data from the sensor. It also starts the *sensor* driver for the corresponding sensor. This driver performs the low-level data collection and preprocessing before it is sent upstream to the sensor handler. With this architecture, it is a simple procedure to add a new deep learning model without breaking the system. Moreover, potentially new sensor inputs can also be added to the system.

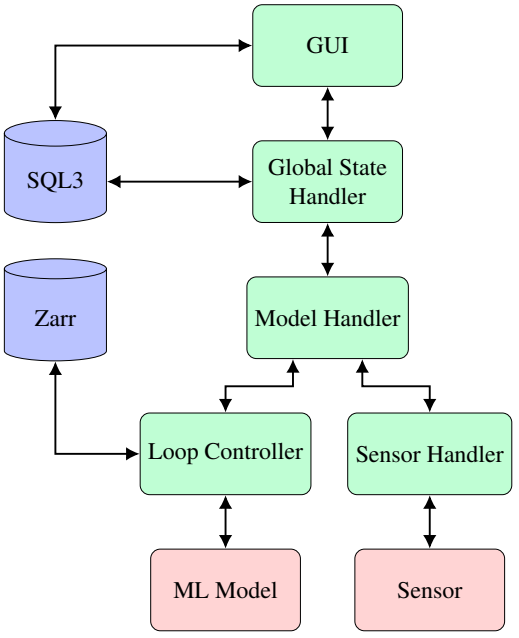

Fig. 4: The system architecture of the AI-Box. The *green* squares indicate that they are static models, the *blue* is databases, and the *red* is non-static model depended layers.

4. Experiment

To validate that it was possible to train and utilise the AI-Box and it would be feasible in an SME, two experiments were conducted. Firstly it was tested at an industrial partner, and secondly, locally at Aalborg University.

4.1. Experiment at industrial partner

The experiment was conducted at an industrial partner specialising in palletizing solutions. A known problem at their palletizer machine is that slip-sheets gets stuck under the picked up layer and no alarm is activated. The AI-Box was placed beside the palletizer, and a webcam was mounted in the corner of the palletizer pointing towards the surface area beneath the picked up layer. The palletizer picked up a layer of cardboard boxes with no slip-sheet below it. Different types and looks of slip-sheets were then placed under the picked-up layer, and images were taken of them. Moreover, images were also captured with no slip-sheets present. The deep learning model used was a simple convolutional neural network (CNN) based on AlexNet [3]. In Fig. 5, the resulting train and validation accuracy is shown. It can be seen that the model converge quickly, with the first 100% test accuracy at epoch 36, which took 37 seconds. After 2 minutes, the test accuracy started to converge. The training first starts after 30 unique samples have been acquired, moreover, all training data was randomly augmented on the fly to increase the variations in the relatively small dataset. There is, in total, nine possible types of augmentation to be performed on each sample, and they are brightness, contrast, flipping, hue, saturation, quality, rotation, blurring, and cropping. On each sample, there is a probability of 0.6 that a random augmentation is applied which is evaluated after each performed augmentation. In Fig. 6, the probability for the number of augmentation per sample is shown. Because of the small dataset, the validation accuracy is the same data as the training just without augmentation applied. Throughout this test, a total of 87 samples was captured.

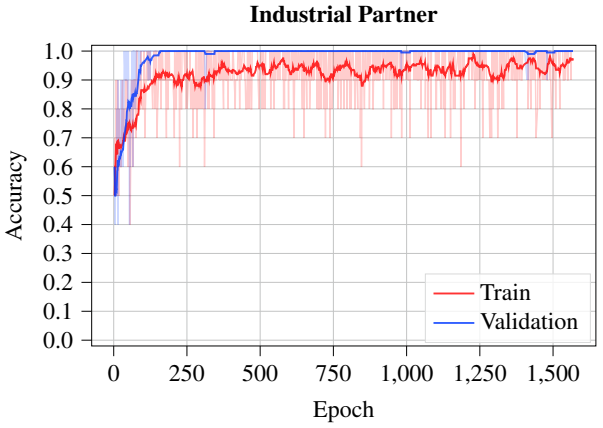

Fig. 5: The train and validation results of the slip-sheet detection at the industrial partner. The train and validation lines are averaged over 20 runs.

4.2. Experiment at FESTO CP Factory

Besides the test at the industrial partner, a test at the FESTO CP Factory line located at Aalborg University was also conducted. The FESTO CP Factory line serves as a learning factory of Industry 4.0 for students and researches at Aalborg University, and it produces smartphones mock-ups. The test had two purposes:

1. Test the applicability of the AI-Box
2. Test the implemented model on a different environment

The objective was to classify whether a blue case or a black case was present at the conveyor. In Fig. 7, the two classes, is shown. The AI-Box was connected to a power outlet and connected to

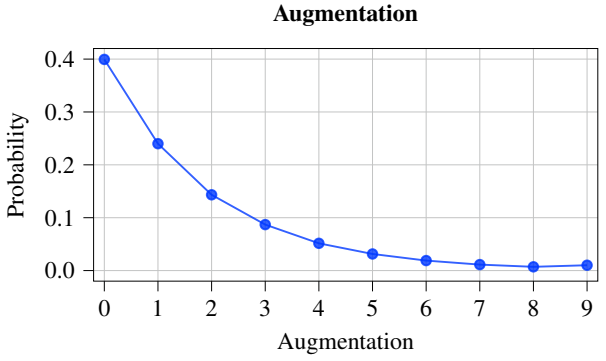

Fig. 6: The probability for the number of augmentations applied to each sample. The values are calculated from historical data from the slip-sheet test.

the LAN of the FESTO CP Factory. Then a webcam was placed at the line and connected to the AI-Box. The AI-Box was turned on, and the URL of the AI-Box was entered in a browser on a laptop. Here the interface of the AI-Box was shown. A new model in the interface was created, and the AI-Box was started and waiting for new image data. In total, 41 samples were collected at the line and, as with the slip-sheet experiment, the training first started after 30 collected samples. The same deep learning model architecture was used, and the result can be seen in Fig. 8. In this experiment, the same premises applied as with the slip-sheet detection, because of the shallow sample size the training data had randomly performed augmentation to it, and the validation data was the same dataset just with no augmentation applied. The results showed that after just 17 epochs, the validation started to converge, which only took 35 seconds after the training began.

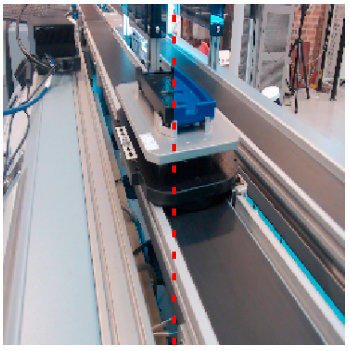

Fig. 7: The view from the AI-Box webcam of the classification object, with a size of 256x256 pixels. The left side is a black case, and the right is a blue case.

4.3. Comparison of CNNs

There exist many high performing CNNs [20]. To select the right one for the AI-Box can be a challenge. Our requirements

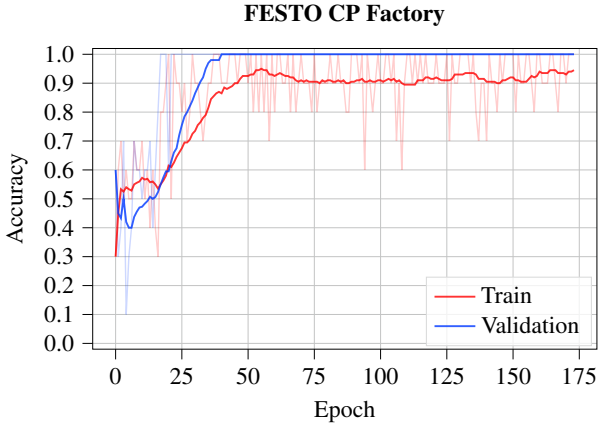

Fig. 8: The train and validation results of the cover detection at FESTO CP Factory. The train and validation lines are averaged over 20 runs.

for the model was high general performance and low complexity. Bianco et al. [20] compared the top of the line CNNs on both a desktop GPU and an Nvidia Jetson TX1. Their research showed that "*there is not a linear relationship between model complexity and accuracy*". For the AI-Box three architectures was chosen to be tested, AlexNet [3], ResNeXt [21], and SqueezeNet [22]. They all three represent diferent apspects of the CNN research:

AlexNet: Is a now classical CNN architecture with convolutional layers, max-pooling for feature extraction and fully connected layers for classification. Compared to more modern architectures, AlexNet has a simple layout though with many trainable parameters.

ResNeXt: Is a more modern CNN built on the idea of residual blocks from He et al. [23]. It contains multi-stacked residual blocks which counteract vanishing gradients. Compared to AlexNet, its architecture is more complicated, and it also has more trainable parameters.

SqueezeNet: Is the last CNN considered in this paper. SqeeuzeNet is meant to be a lightweight model where its fire modules which reduce and expand the image helps with keeping the number of parameters down. It has no fully connected layers in the end for classification but relies on the last convolutional layer to do the classification and extract them with a last global average-pooling layer.

These three architectures have all been altered slightly, mainly to reduce memory footprint, and thus all are a custom implementation in Tensorflow 2.0 and Python 3.6. The changes for AlexNet is that there is no *local response normalization* and reduced convolutional and fully connected layers. ResNeXt changes are mainly the filters for the residual block is halved and some removed. The SqueezeNet is already a rather small model (memory-wise), and thus the only alterations are the changes of the last convolution to match the output classes.

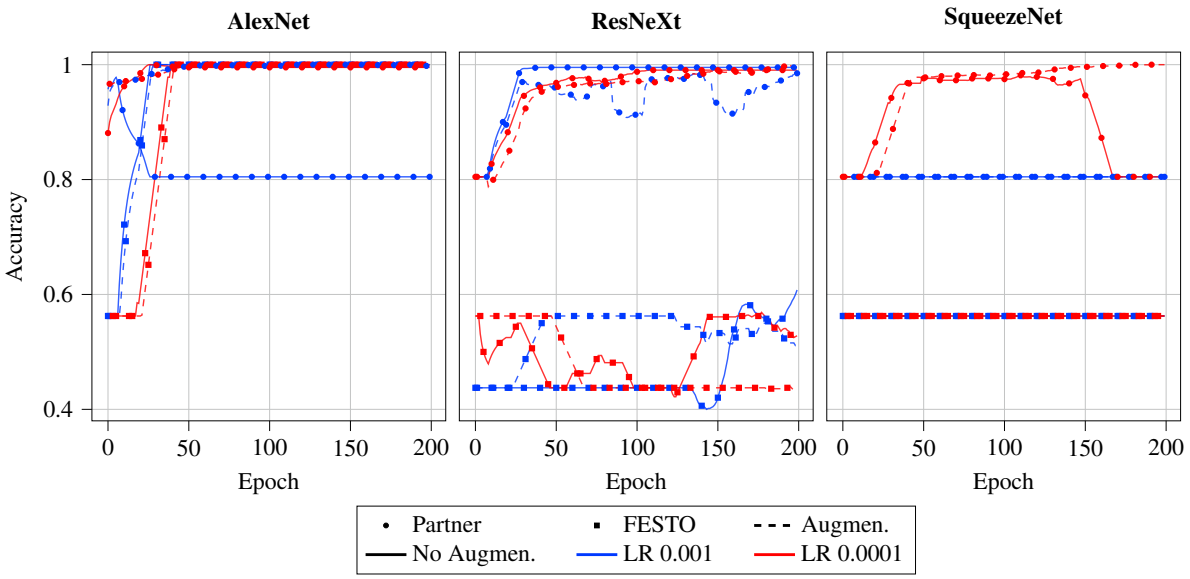

Fig. 9: The validation plot for the different models. The lines are averaged over 20 samples.

In Tab. 1 the architecture of the implemented models are presented. The models were verified with the dataset of the cov-

Table 1: The different specifications for the implemented architectures. For AlexNet a dropout is placed after all convolutional operations. In ResNeXt no-bias is used in the convolutional operations. All models use Adam optimiser with default hyperparameters.

AlexNet	**ResNeXt**	**SqueezeNet**
2 x Conv(32,3,1)	ZeroPad(3)	Conv(96,7,2)
MaxPool(2)	Conv(64,7,2)	MaxPool(3)
Conv(64,3,1)	BatNorm(1.001e-5)	2 x F.Module(16,64,64)
MaxPool(2)	ZeroPad(1)	F.Module(64,128,128)
Conv(128,3,1)	MaxPool(3)	MaxPool(3)
MaxPool(2)	3 x R.Block(64,1,32)	F.Module(32,128,128)
Dense(128)	3 x R.Block(128,2,32)	2 x F.Module(48,192,192)
Dense(68)	3 x R.Block(256,2,32)	F.Module(64,256,256)
Dense(2)	2 x R.Block(512,2,32)	MaxPool(3)
	GlobalAvgPool	F.Module(64,256,256)
	Dense(2)	Conv(2,1,1)
		GlobalAvgPool
Parameters	**Parameters**	**Parameters**
16,888,226	22,576,706	736,450

ers from the FESTO CP Factory with a total of 41 samples and 53/47 class distribution. Eight samples were taken from the 41 samples as validation data. Secondly, the models were also tested on a larger dataset from the industrial partner with a total of 1050 images and a class split of 80-20, where 210 samples were selected as validation data. It should be noted that the validation data was only used in inference mode and was not used to change hyper-parameters. The comparison test was conducted to test the three models and to test if data augmentation increases the performance, and lastly how the learning rate affected the model. The test was only allowed to run for 200 epochs to validate the needed quick learning. In Fig. 9 the result is shown. From the results, we made several observations:

- AlexNet is the overall best performer, with the only time it did not learn was slip-sheet detection with a high learning rate and no data augmentation.
- ResNeXt is able to learn the larger dataset in a few epochs, whereas it did not learn the smaller dataset within the 200 epochs.
- SqueezeNet was only able to learn the larger dataset with a low learning rate and with the use of augmented data.
- A lower learning rate, as expected, halts the learning, but it is also observed that the higher learning rate does not find a sufficient minimum in time.
- Performing random augmentation to the data also decreased the learning rate slightly but also enabled some of the models to learn the dataset because of the added variation to the data.
- AlexNet is located in between the two others regarding the number of trainable parameters, it is the less complex of the three, with only convolutional, max pooling and fully connected layers. Our results show that a less complicated CNN architecture performs better on a less complicated problem (few classes).

5. Conclusion and future work

In this paper, our focus was to describe the current work being done on what is named the AI-Box. The AI-Box serves as a versatile easy to use deep learning device which can aid industrial SMEs to enhance part of their production that will otherwise be too costly/difficult. In its current form, only images are possible to be classified, and experiments were conducted to gain a better understanding of what type of deep learning model is required. The conclusion was that a less complicated CNN showed to be beneficial for small dataset along with an

additional augmentation of the sample data. Moreover, the two experiments conducted showed that the AI-Box is versatile and functional in an expected manner regarding setup and performance.

For future work, regarding the image models, it could be of interest to investigate the use of methods such as Grad-CAM [24] to visualise what in the image is learned. This will improve the easy to use for operators and ensure that the model learns the correct features of the image. As of now, all the models are trained from scratch, so for better performance and faster learning transfer learning should be implemented. Besides the image models, deep learning models that should classify sound, vibration, and temperature should also be implemented and tested where recurrent neural networks should be tested. Moreover, outlier detection for all the different sensor inputs should also be exploited for use cases which do not have specific classes. A crucial task to solve is the preprocessing of data. As of now, the image data is reasonably easy to preprocess:

1. Reduce the size and convert to a three-dimensional matrix.
2. Add to the Zarr database along with the label.
3. Normalise the image between 0-1 before use.

For time-series data, this is a more complicated matter. Here different techniques as slicing the time-series data into short frames for classification can be used. Moreover, the various features such as the spectrogram, MFCC spectrogram and the time domain waveform should be investigated. These aspects need to be addressed in future work, and the core idea of the AI-Box is still the easy-to-use and versatility. Therefore, these feature selection should be made without the knowledge of the user through methods such as principal component analysis. Furthermore, more tests should be performed with different operators to verify the versatility and easy to use of the AI-Box with different data and classification objectives.

Acknowledgements

The work was conducted as a part of the Danish strategy for Industry 4.0 in SMEs called Innovation Factory North.

References

[1] E. Plastino, M. Purdy, Game changing value from artificial intelligence: eight strategies, Strategy & Leadership (2018).

[2] B. W. Wirtz, J. C. Weyerer, C. Geyer, Artificial intelligence and the public sector—applications and challenges, International Journal of Public Administration 42 (7) (2019) 596–615. doi:10.1080/01900692.2018.1498103.

[3] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.
URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[5] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, M. Harnisch, Industry 4.0: The future of productivity and growth in manufacturing industries, Boston Consulting Group 9 (1) (2015) 54–89.

[6] J. Gao, Machine learning applications for data center optimization (2014).

[7] N. Lazic, C. Boutilier, T. Lu, E. Wong, B. Roy, M. Ryu, G. Imwalle, Data center cooling using model-predictive control, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31, Curran Associates, Inc., 2018, pp. 3814–3823.
URL http://papers.nips.cc/paper/7638-data-center-cooling-using-model-predictive-control.pdf

[8] A. Sumesh, K. Rameshkumar, K. Mohandas, R. S. Babu, Use of machine learning algorithms for weld quality monitoring using acoustic signature, Procedia Computer Science 50 (2015) 316–322.

[9] E. B. Hansen, R. E. Andersen, S. Madsen, S. Bøgh, Transferring human manipulation knowledge to robots with inverse reinforcement learning, in: 2020 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2020, pp. 933–937. doi:10.1109/SII46433.2020.9025873.

[10] S. Mittal, M. A. Khan, D. Romero, T. Wuest, A critical review of smart manufacturing & industry 4.0 maturity models: Implications for small and medium-sized enterprises (smes), Journal of Manufacturing Systems 49 (2018) 194 – 214. doi:doi.org/10.1016/j.jmsy.2018.10.005.

[11] C. Sapp, D. Brabham, J. Antelmi, H. Cook, T. Craig, S. Barot, D. Galli, S. Pal, S. Mohan, G. Gilbert, 2019 planning guide for data and analytics (2018).
URL https://www.gartner.com/en/doc/361501-2019-planning-guide-for-data-and-analytics

[12] Blackbird, Blacbird homepage (2020).
URL https://www.blackbird.online/

[13] Monitor-Box, M-box homepage (2020).
URL https://www.monitor-box.com/m-box/

[14] DTI, Dti vision box: Quality control and inspection in 2d and 3d (2020).
URL https://www.dti.dk/specialists/dti-vision-box-quality-control-and-inspection-in-2d-and-3d/35124

[15] Y. Yang, Z.-J. Zha, M. Gao, Z. He, A robust vision inspection system for detecting surface defects of film capacitors, Signal Processing 124 (2016) 54 – 62, big Data Meets Multimedia Analytics. doi:doi.org/10.1016/j.sigpro.2015.10.028.

[16] L. Li, K. Ota, M. Dong, Deep learning for smart industry: Efficient manufacture inspection system with fog computing, IEEE Transactions on Industrial Informatics 14 (10) (2018) 4665–4673. doi:10.1109/TII.2018.2842821.

[17] NVIDIA webpage, Jetson agx xavier developer kit (2020).
URL https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[18] Zarr Developers, Zarr, https://github.com/zarr-developers/zarr-python (2020).

[19] G. E. Krasner, S. T. Pope, et al., A description of the model-view-controller user interface paradigm in the smalltalk-80 system, Journal of object oriented programming 1 (3) (1988) 26–49.

[20] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access 6 (2018) 64270–64277. doi:10.1109/ACCESS.2018.2877890.

[21] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 5987–5995.

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360 (2016).

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[24] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, International Journal of Computer Vision 128 (2) (2019) 336–359. doi:10.1007/s11263-019-01228-7.