Article

Deep machine learning of the MobileNet, Efficient and Inception models

Monika Rybczak * and Krystian Kozakiewicz

Posted Date: 30 November 2023

doi: 10.20944/preprints202311.2003.v1

Keywords: artificial intelligence; deep learning; CNN; Mobilenet; EfficientNetB0; Inception

Article

Deep Machine Learning of the MobileNet, Efficient and Inception Models

Monika Rybczak * and Krystian Kozakiewicz

Faculty of Marine Electrical Engineering, Gdynia Maritime University, Morska 83, 81-225 Gdynia, Poland
* Correspondence: m.rybczak@we.umg.edu.pl

Abstract: Today, specific CNN models assigned to specific tasks are often used. In this article, the authors have investigated three models MobileNet, EfficientNetB0 and InceptionV3. Artificial intelligence methods for deep networks will be presented but implemented in a limited computer resource. Three types of training bases were investigated, starting with a simple base verifying five colors, followed by recognition of two different orthogonal elements, and then more complex images discriminating between elements. The research aimed to demonstrate the models' capabilities based on training base parameters such as the number of images and epoch types. The architectures proposed by the authors in these cases were chosen based on simulation studies conducted on a virtual machine with limited hardware parameters. The proposals present the advantages and disadvantages of different models based on TensorFlow, Keras libraries in the Jupiter environment written based on the Python language. The authors' proposal of the selected AI model enables further work on e.g. image classification, but in a limited computer resource for industrial implementation e.g. on a PLC.

Keywords: artificial intelligence; deep learning; CNN; Mobilenet; EfficientNetB0; inception

1. Introduction

As reported by this source [1] convolutional neural network is a 74703 publications have been published within this topic receiving 2050983 citations. The earliest article on CNNs is from the 1990s [2] where the LenET-5 model was proposed by LeCun. After many years, due to the lack of suitable tools, the possibility of reducing the dimensionality of data using neural networks was demonstrated in 2006 [3]. The most popular models such as AlexNet, DeepFace and DeepID in image classification including face recognition were demonstrated in the work [4], among others. In the medical industry, AI is used, for example, to recognise inconsistencies in lung lesions, during X-ray analysis [5]. Image recognition is useful for autonomous cars when recognising signs on the road [6]. In CNNs, activation functions, convolution operations, merging and training data are significant. The second aspect of classical neural networks is reliability in security systems. Python can be used to implement image classification. Using high-performance computing (HPC) and parallel processing, CNN models can be accelerated to reduce classification time. Different data preprocessing methods and hardware configurations can also affect the effectiveness of image categorisation. Optimised Python environments for HPC can significantly increase the speed of image classification while maintaining high accuracy. The paper [7] discusses the performance comparison of spiking CNNs with traditional CNNs in image classification tasks using Python. Currently, CNNs or deep neural networks have real-world, industrial application examples in the food and automation industries. In a paper [8], Mobilenet was shown to provide better accuracy compared to other models such as DenseNet and traditional CNN models. In the context of bird species classification, the Mobilenet model outperformed other models in terms of accuracy. Similarly, in a study of tomato seed variety classification, the MobileNet model achieved the highest classification accuracy [9]. In a study of rice plant leaf classification, the Mobilenet model with 150 epochs resulted in the highest accuracy [10]. These findings highlight the effectiveness of the Mobilenet model in various image classification tasks. Another example of CNN applications is the EfficientNetB0 model, which has shown promising results in skin disease medicine [11], haemoglobin level classification for anaemia diagnosis [12] or white blood cell classification [13]. In the diagnosis of anaemia, the EfficientNetB0 model achieved a high accuracy of 97.52%, and in the classification of white blood cells, an accuracy of 99.02% was achieved. In the industrial sector, a solution can be proposed by the authors [14], where the learnt model directed the electromagnetic leakage information in AES encryption chips, learning the attack response model of cryptographic FPGA chips. Another proposal of the CNN model is Inception, an example is the contactless identification of people based on facial features read from an image, where the validation accuracy level was 99.7% [15]. Interesting results were reported in a paper [16] where the Inception model was used to accurately decode and recognise EEG motor images. In the paper [17], it was shown that to recognise hand-held football weapons from digital images is possible at are results of 87% accuracy.

The study of convolutional networks has been transferred to industrial conditions. For example, a lightweight model, CondenseneTV2, was proposed to identify surface defects and successfully detected faults during low-frequency operation on edge equipment [18]. Deep neural networks have been used in social media in the context of Industry 4.0 to analyse social sentiment for websites to investigate customer satisfaction [19]. A paper [20] used CNNs to predict the 'trajectory' of object grasping in real time in an industrial application. Siemens introduced the ability to observe and respond to images in real time [21,22]. For the task, the manufacturers proposed a specialised artificial intelligence module enabling the inclusion of an artificial intelligence model algorithm based on the CNN architecture.

In this work, the authors investigated a new training base for three different groups of RGB images. Subsequently, they analysed the performance and accuracy of the built model for several epoch ranges. The existing MobileNet model was compared to the CNN architecture, EfficienNetB0 and InceptionV3 were proposed, programmed, and implemented. The models were verified, and characteristics were plotted for a range of different epochs, to test the accuracy of the learned model for image verification. The research is the first step to realise image recognition by a real obiket configured with an S7-1500 family controller compatible with the AI module and Intel RealSeans camera.

2. Materials and Methods

The testbed is the result of a collaboration with Siemens, which proposed the main algorithm of the convolutional network for image recognition. The paper includes the steps of learning the image recognition neural network. Based on the analysis of the obtained results, the authors justify the CNN model selected for validation.

The study deals with the training of the models based on constructed databases and the comparison of results for different TensorFlow and Keras libraries. Finally, based on the existing MobileNet model, the authors compare and finally propose the artificial neural network models EfficieneNetB0 and InceptionV3.

2.1. Research station

The testbed concept was created to verify the implementation process of artificial intelligence algorithms using a PLC, a special Siemens artificial intelligence NPU (neutral processing unit), a RealSense image capture camera and a computer running Linux software [31].

The artificial neural network is built in a model-training environment on a virtual disk in Linux and is transferred to an SMC card, a Siemens Simatic memory card. The card allows the data stored on it to be used in Siemens products, such as the NPU artificial intelligence module used in the project [21–23], among others. This means so much that the learned model can be further interpreted in PLC-based control of real objects.

2.2. Converting the CNN artificial intelligence model to the SMC card

The virtual machine uses tools to convert artificial neural network models based on image classification. Once the .pb model is obtained from the python calculations, the model needs to be converted to a .blob type. This is because the .pb model recognises static images, whereas the camera in the test bench uses real-time image data reading and it is the .blob extension that allows such an image to be processed for the artificial neural network model. The testbed is built on a virtual machine running the Linux Pop!_OS operating system. The TensorFlow library [24–26] was used, on which the neural network models were trained. In addition, the AI library was used, in Jupyter Notebooks on the first docker and the OpenVINO library [30] on the second docker. Oracle VM VirtualBox was used to run the virtual disk. The NPU module is equipped with the chip contains 16 low-power programmable SHAVE (Streaming Hybrid Architecture Vector Engine) cores and in addition an accelerator for overbulding deep neutral network structures Intel Movidius MyriadTMX [31]. In this configuration, it is possible to upload the learned neural network to the SMC card, which can be fed into the artificial intelligence module, and this is recognised by the PLC. The main purpose of the applied classification is the intelligent recognition of specific objects and visual quality control. The test rig is included in Figure 1.

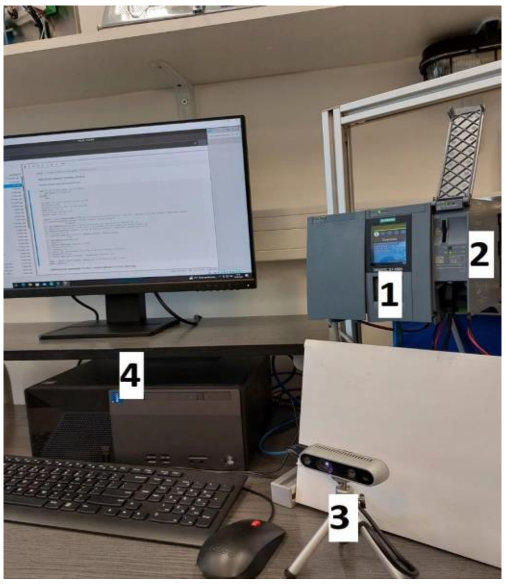


Figure 1. Point 1 Controller from the S7-1500 family, 1517 TF -3PN/DP. The controller configuration was performed in the TIA Portal v16 programming environment. The program takes into account 2 function blocks and 4 data blocks DB.2 Simatic TM NPU artificial intelligence module with SMC card, which contains configuration files and a converted artificial neural network module. 3 Intel RealSense image camera (reading images and verifying them in the TIA Portal environment). The camera is connected via USB 3.1 directly to the NPU module. 4. computer with TIA Portal v16 software, must be a separate PC or laptop, which is also recommended by the manufacturer.

The proposed testbed allows:

- Python programming using the TensorFlow and Keras libraries to create and teach an artificial intelligence model,
- producing models according to research needs,
- conversion of artificial neural networks for use in the test bench.

3. Method – MobileNet, Efficient and Inception

The method for creating a new artificial intelligence model architecture is based, among other things, on a properly prepared learning database. The database in this example is a collection of three different groups of images, that is:

- The first is in the form of coloured squares made in a graphics programme by the authors. The training images were simple and uniform. From these images, the relationship of the number of epochs to the effectiveness of the artificial intelligence models is investigated, as well as the rationale for selecting the correct number of epochs.
- The second relates to two-colour images of a cuboid called a container in the paper. These are labelled in red and blue.
- The third group of images is a collection of multiple elements, electronic parts, where the model is supposed to learn correct elements and incorrect elements in order to possibly select the correct element by further work of the object.

Several functions [28,29] were used to build the artificial intelligence model, allowing the AI model parameters to be refined:

- ReLU: Rectofied Linear Unit activation function in neural networks. Mathematical formula:

$$f(x) = \max(0, x) \qquad (1)$$

This means that if f(x) is positive (the function is a straight line with a 45-degree slope), the function returns x; whereas if it is negative or equal to 0, the function returns 0 (graphical function - horizontal line).

- Convolution2D refers to a two-dimensional convolution operation, which is a fundamental element in convolutional networks (CNNs) that process images. Convolution is the mathematical operation of combining two functions to produce a third function as a result. In the context of image processing, convolution involves moving a 'filter' or 'kernel' through an image and calculating the sum of the products of the filter elements and the corresponding image elements. For the formula:

$$(I * K)(x, y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} I(i, j)K(x - i, y - j) \qquad (2)$$

where I – image, K – convolution kernel. The convolution process allows the network to detect local features in the image: texture, shape and edges. In practice, the image and the kernel are finite-dimensional matrices, so the summation is done only by finite dimensions. In the example presented here, the convolution kernel has a matrix of (3x3).

- Maxpooling2D – technique allows the selection of the maximum value from a specific region in the input matrix otherwise a feature map. For example, the feature is of the form:

$$output(i, j) = max_{m=0}^{k-1} max_{n=0}^{k-1} input(i * stride + m, j * stride + n) \qquad (3)$$

where: input (I,j) is the value of a pixel at position (i,j) of the input feature map, output(i,j) is the value of a pixel at position (i,j) of the output feature map; k – kernel size; pooling; „stride" is the step by which the window is moved during the operation „pooling". Stride can be equal to the size of the kernel (window), leading to non overlapping windows and reduced dimensionality in each dimension. The example shown here has a set value of (2,2).

- Dropout is a regularisation technique, which means that it helps to prevent overfitting (overfitting) of the model to the training data. During training for each iteration, Dropout selects a certain random percentage of neurons in the stratum, set to zero. The percentage of neurons is a parameter and has the name 'dropout rate'. Mathematical notation:

$$y_i = x_i * d \ {*}(1/p) \qquad (4)$$

where: x_i is input neuron, yi is the output from the neuron after the application of Dropout, di is a random Bernoulli variable that takes the value 1 with probability p (the probability of the neuron's behaviour); p is the 'dropout rate', the probability of each neuron's behaviour.

- BlobalAveragePooling2D is a layer often used as a layer before the last densely connected layer (Dense) in CNN models, it reduces dimensionality and prevents over-fitting. Mathematical notation:

$$output(c) = \frac{1}{height * width} \sum_{i=1}^{height} \sum_{i=1}^{width} input(i, j, c) \qquad (5)$$

where: output (c) is the output value for the feature channel c; input (i,j,c) is the value under the heading (i,j) for feature channel c; it should be noted that the sums are calculated by all spatial positions (i,j).

A control algorithm for network learning based on CNNs, or convolutional neural networks, is presented below.

Table 1. General working diagram of the presented system used during model learning.

```
Algorithm: Trenning AI model for image
IMPORT TensorFlow, compact version 1
       Convvolution2D, MaxPooling2D, BlobalAveragePooling2D, Dropout, ReLU
  1. Training data – categorisation of folders and files
  2. Image_size (224,224), batch_size = 256, RGB
  3. Prepare generate function my_preproc (img), shift, brightness, rotation image.
  4. Processing test:
       • New AI model based on imported libraries: Convvolution2D, MaxPooling2D,
         BlobalAveragePooling2D, Dropout, ReLU.
       • Finished model MobileNet.
       • Model classifier – AI learning (with transfer learning classifier).
       • Loading of finished, clean model (without transfer learning classifier).
  5. Network training
       • Hyperparameter settings
       • Network training
  6. Evaluation of results
```

The study concerns the verification of three artificial intelligence models implemented on a virtual disk running on TensorFlow and Keras libraries. A Siemens manufacturer's algorithm based on the MobileNet model was investigated. Based on the selected kernel parameters and the selection of databases on the basis of epoch analysis, it was decided to verify the performance of the EfficenNet and Inception model. The website [27] shows the main performance characteristics for 38 models. As several architectures deal with RGB images it was decided to try to compare the Mobilenet model with the VGG model, but due to the limitations of the virtual disk hardware it was decided to investigate two models EfficenNetB0 and InceptionV3.

First traning date base – simple data base

Each colors was duplicated four times, giving a total of 20 training images per color. The models differ only in the number of learning epochs. Hence, with the same epochs, there should be a similar value for the effectiveness of the artificial neural network, the probability of making a correct decision about the class of a given image. The difference between epochs is due to issues of randomness when training the artificial neural networks. Figure 2 shows an example of the images generated by the training data generator for the first database.

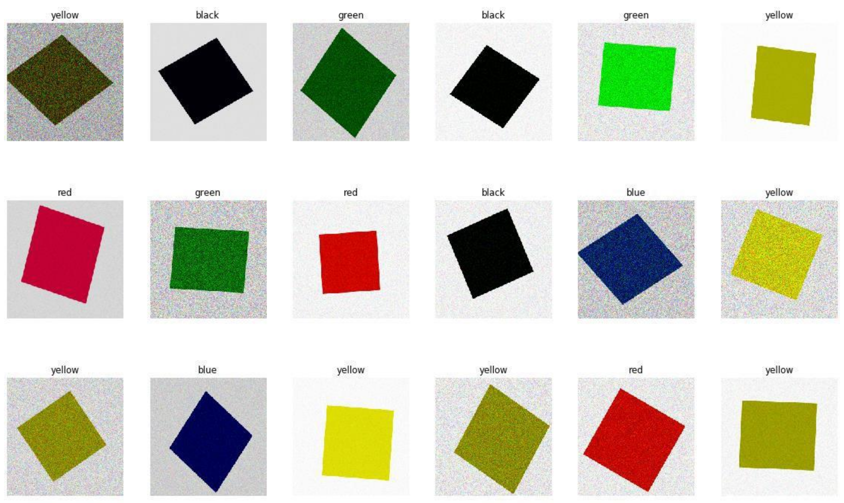


Figure 2. Example result of a training data generator - database for determining five colours.

Figure 2 shows the end result of the training image generators. After adding deviation and noise, new training images used during learning were created.

Second training data base – data base with cuboid

This study was intended to show how a learned model would behave, where the training data are the images in the example presented, a blue and red cuboid in the task referred to as a container.

In this training data set, there are actual images of two container models with a background and the object itself changing its rotation and its illumination. Figure 3 shows a high accuracy model prediction with a training image base of 1000.

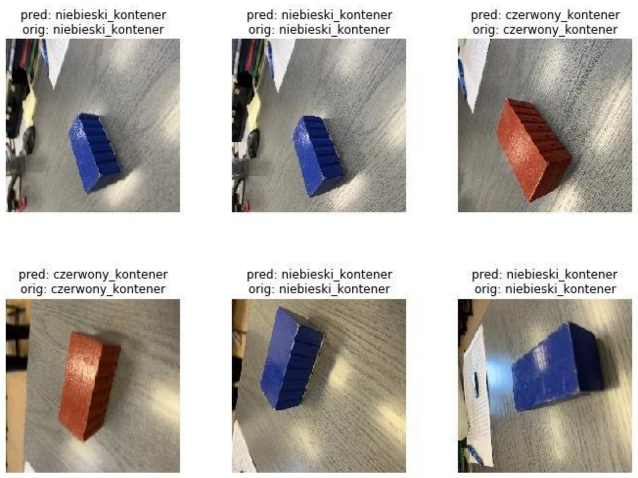


Figure 3. Examples of model predictions based on inference, 'pred' denotes model prediction, 'orig' the actual class of image.

Third training data base – data base with difeerent object

Training base three is made up of several different objects. The model has to learn to distinguish between pictures of electrical elements and other elements. This time the model has to be able to distinguish between the designated elements, i.e. the electrical elements, and other coloured elements, completely unrelated to the training base.

Figure 4 shows a prediction with accuracy built from 100 training images. Exactly 50 images of the correct element and 50 images of the other elements.

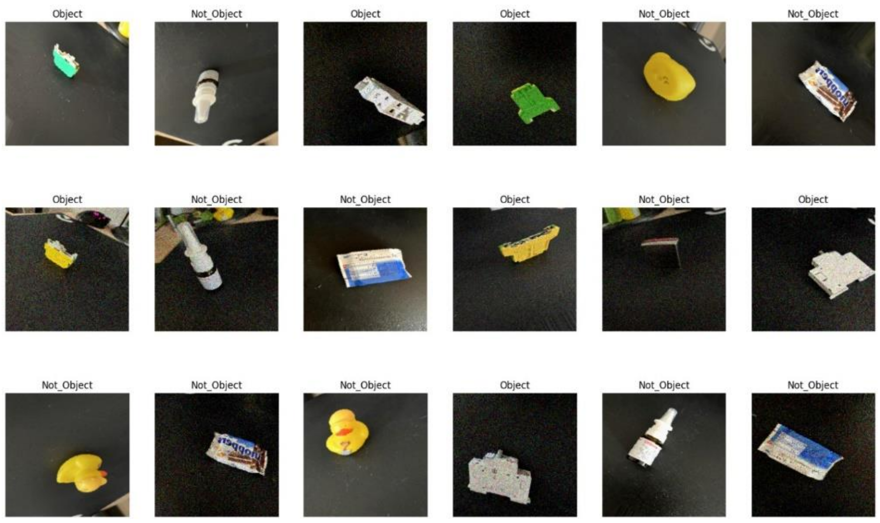


Figure 4. Example model predictions based on inference, 'Object' denotes a valid element, 'Not_object' an element outside the set.

4. Results – verifications tree models CNN

The results presented here are based on three training bases. The artificial intelligence models used in the first two examples presented relate to the MobileNet model, while the third training database example was built on three CNN models, specifically MobileNet, EfficienNetB0 and InceptionV3. A study was carried out for three different training databases. A verification of the accuracy of the artificial intelligence model based on the MobileNet model was performed. Then for the third task, where there was to be information about the object being searched for in the image with an indication of which object in the image was wrong. After analysing the performance of the MobileNet model, it was decided to perform the 3rd task based on the verification of the EfficienNetB0 and InceptionV3 models.

The effectiveness of the models depends on several basic factors, such as the number of learning epochs and the amount of training data. The following examples demonstrate the differences between the factors selected based on specific sets of images and the accuracy.

4.1. CNN-based artificial intelligence model research, including MobileNet

First traning date base – simple data base

The first study concerned the learning of a model to recognise five computer-entered colours, not even the recognition of colours in photographs, but colours entered in a computer programme.

At the very beginning, an attempt was made to check the accuracy of the learned model for 10, 15 and 20 epochs successively. This work showed that the model under study could be configured with sufficient accuracy for as few as 15 epochs. The results are shown in Table 1.

Table 2. Effectiveness of learned models across eras.

Epoch	Accuracy of learned model 10 epochs	Accuracy of learned model 15 epochs	Accuracy of learned model 20 epochs
1	0,20	0,35	0,19
2	0,49	0,40	0,31
3	0,64	0,63	0,55
4	0,74	0,71	0,65
5	0,78	0,74	0,66
6	0,78	0,84	0,78
7	0,78	0,84	0,81
8	0,76	0,90	0,88
9	0,78	0,94	0,84
10	0,80	0,95	0,93
11	-	0,95	0,94
12	-	0,99	0,98
13	-	1,00	0,96
14	-	0,95	0,93
15	-	0,96	0,91
16	-	-	0,94
17	-	-	0,96
18	-	-	0,97
19	-	-	0,97
20	-	-	0,99

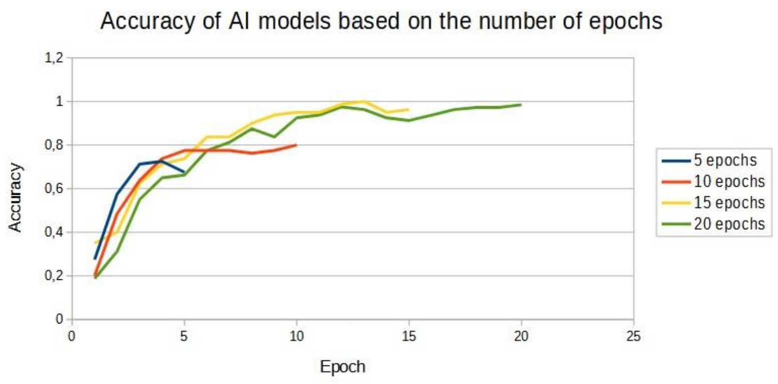


Figure 5. Accuracy graph for models based on different numbers of learning epochs.

Second training data base – data base with cuboid

Four models trained for 20 epochs were created to discuss the results. However, the models differ in the number of training photos. The images remain the same but the number is duplicated or truncated as the case may be. The training sets were divided into four ranges: 10, 50, 100 and 1000 images are presented in Table 2.

Table 3. Efficacy of learned models with different numbers of training images across eras.

Epoch	Accuracy of the learning model 10 images	Accuracy of the learning model 50 images	Accuracy of the learning model 100 images	Accuracy of the learning model 1000 images
1	0,50	0,49	0,43	0,80
2	0,75	0,63	0,64	0,98
3	0,75	0,71	0,70	0,98
4	0,50	0,71	0,84	0,99
5	0,50	0,88	0,89	1,00
6	0,75	0,93	0,93	1,00
7	0,75	0,88	0,88	1,00
8	0,75	0,98	0,96	1,00
9	0,63	0,88	0,96	1,00
10	0,75	0,85	0,98	1,00
11	0,75	0,93	0,94	0,99
12	0,75	0,95	0,98	1,00
13	0,88	0,93	0,98	1,00
14	1,00	0,93	0,93	0,99
15	0,63	0,93	0,96	1,00
16	0,88	0,95	0,96	1,00
17	0,88	0,95	0,96	1,00
18	0,88	0,93	0,99	1,00
19	0,75	0,98	0,98	1,00
20	0,88	0,93	0,96	1,00

Figure 6 shows a similar increasing trend in accuracy for all models. Each model increased its efficiency similarly to the others. Their highest accuracy value depended mainly on the number of epochs. It is therefore necessary to select an appropriate number of epochs for each data set. According to the graph in Figure 6, it can be seen that too few training images of 10 does not allow the neural network to learn to recognise image classes effectively. The accuracy of this model reaches

100% and it is the only one of the four neural networks that is able to maintain such high performance, unfortunately at the expense of learning time by increasing the number of steps per epoch related to the number of training images.

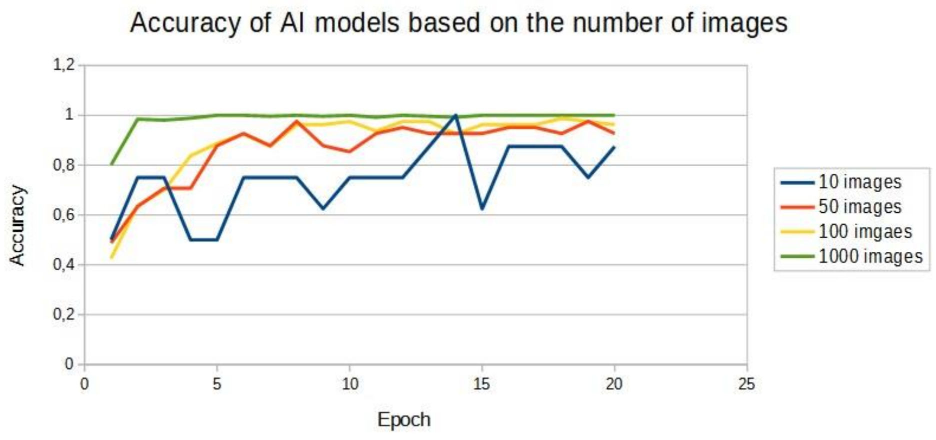


Figure 6. Graph of model accuracy based on number of training shots.

3.2. Thries models: MobileNet, EfficientNetB0 and Inception

Third training data base – data base with difeerent object

The verification of the programmed models includes a comparison of results between the MobileNet model, and EfficientNetB0 and InceptionV2 written based on TensorFlow and Keras library functions such as Convolution2D, BatchNormalisation, ReLU and MaxPooling2D. The above studies show that Mobilenet has a high accuracy of 98% and even 100% under certain favourable conditions. Therefore, due to the scope of the tasks to be performed, it was decided to select two more models from the 38 models proposed in the Keras [27] library. The choice of the proposed models depends in this example on the hardware parameters of the virtual disk on the PC. Parameters is: Intel(R) Core(TM) i5-9400F CPU @2.90GHz 2.90 GHz; Installed RAM 16.0 GB; System type 64-bit operating system, x64 processor, Virtual disk limitations: RAM 10 GB, number of processor cores: 4 of 6. It was decided to verify the third database on three models, where the training base was built from 100 images:

- EfficieNetB0
- InceptionV3
- MobileNet

Table 4. Efficacy of learned models with different numbers of training images across eras.

Epoch	Accuracy of the MobileNet model	Accuracy of the EfficientNetB0 model	Accuracy of the InceptionV3 model
1	0,491666667	0,51666667	0,47916667
2	0,50416667	0,50833333	0,58333333
3	0,50416667	0,47916667	0,51666667
4	0,50	0,45416667	0,5875
5	0,53333333	0,50833333	0,60833333
6	0,45	0,50833333	0,7
7	0,5125	0,51666667	0,725
8	0,59166667	0,55	0,6875
9	0,54583333	0,50416667	0,71666607

10	0,525	0,52083333	0,74583333

The graph shown in Figure 7 shows the accuracies of the three models carried out for learning on a third dataset, on images of electrical element models mixed with objects outside the electrical element set.

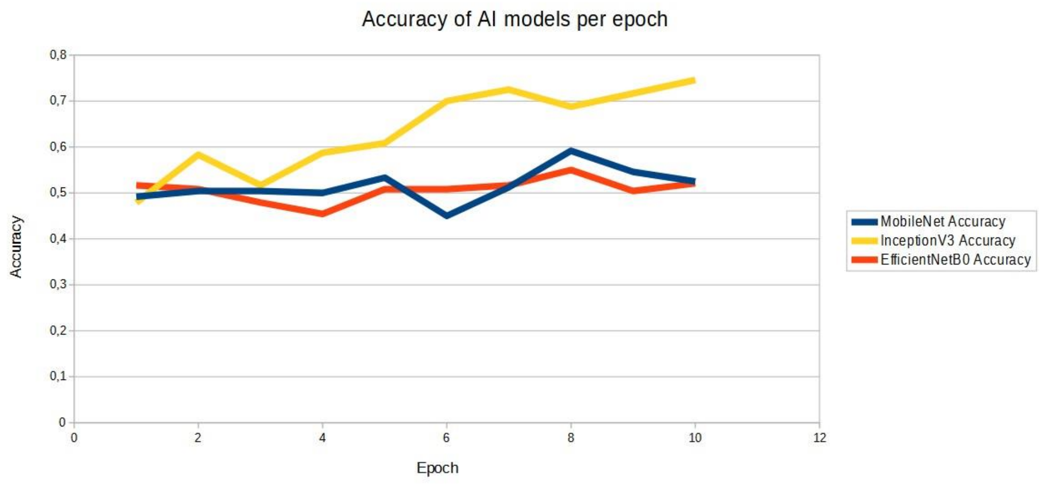


Figure 7. Accuracy of AI models per epoch for third example.

Due to the interesting results obtained for the InceptionV3 model, the dependencies of the three models against learning time were plotted.

The results on Figure 8 presented here concern the training of a third example of an artificial intelligence model at the level of three models such as MobileNet, EfficientNetB0 and InceptionV3.

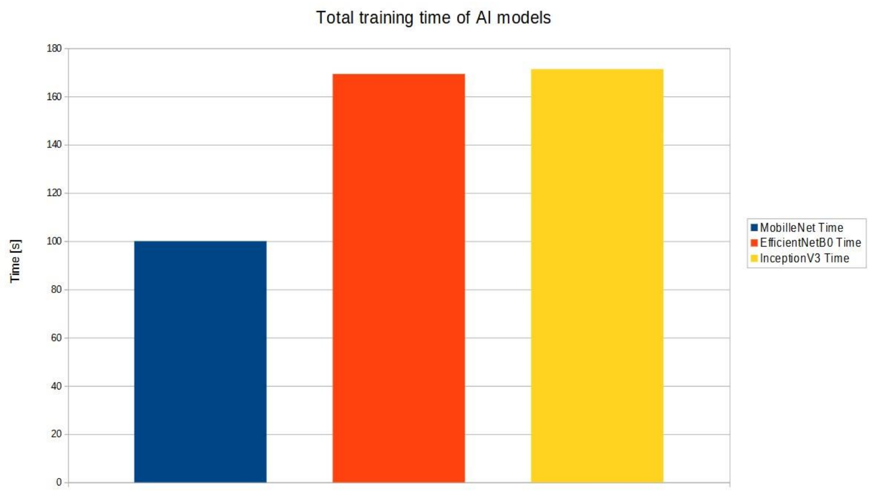
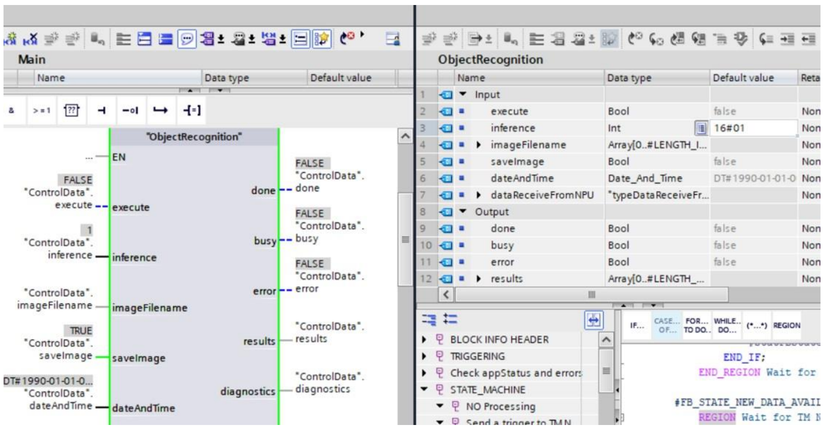


Figure 8. Total training time of three models for third example.

3.3. Verification model Inception on PLC

Teaching the model to recognise elements was introduced to the PLC for further research. In the first instance, the model is in a file format with a .blob extension. The file is then saved to an SMC card on the computer where the model was taught. This card is transferred to the NPU module and, following the steps suggested by the manufacturer, the S7-1500 family PLC reads the parameters from the card [see Figures 9, 10 and 10]. These are not string variables, but only numerical information stored in the range [0....1]. A result closer to 1 means that an element taught by the AI model is detected in the image.


Main
"ObjectRecognition"
EN
FALSE "ControlData". execute — execute
1 "ControlData". inference — inference
"ControlData". imageFilename — imageFilename
TRUE "ControlData". saveImage — saveImage
DT#1990-01-01-0... "ControlData". dateAndTime — dateAndTime
done — FALSE "ControlData". done
busy — FALSE "ControlData". busy
error — FALSE "ControlData". error
results — "ControlData". results
diagnostics — "ControlData". diagnostics

ObjectRecognition

	Name	Data type	Default value	Reta
1	Input			
2	execute	Bool	false	Non
3	inference	Int	16#01	Non
4	imageFilename	Array[0..#LENGTH_I...		Non
5	saveImage	Bool	false	Non
6	dateAndTime	Date_And_Time	DT#1990-01-01-0	Non
7	dataReceiveFromNPU	"typeDataReceiveFr...		Non
8	Output			
9	done	Bool	false	Non
10	busy	Bool	false	Non
11	error	Bool	false	Non
12	results	Array[0..#LENGTH_...		Non

BLOCK INFO HEADER
TRIGGERING
Check appStatus and errors
STATE_MACHINE
NO Processing
Send a trigger to TM N

```
    END_IF;
END_REGION Wait for

#FB_STATE_NEW_DATA_AVAIL
REGION Wait for TM N
```

Figure 9. Part Program on PLC – 1515TF.

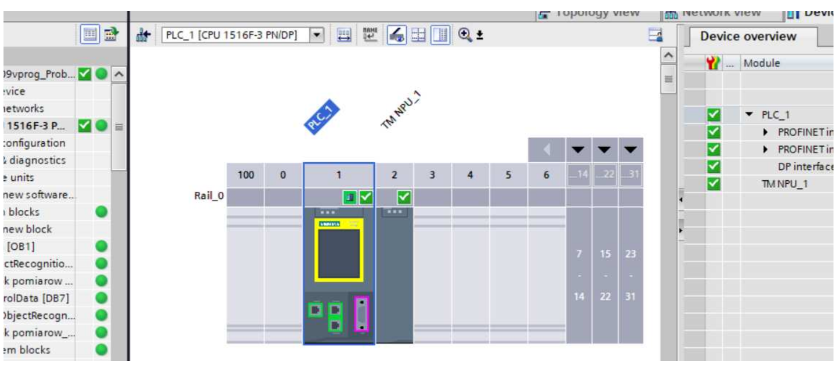


Figure 11. Configuration S7-1516TF with NPU module.

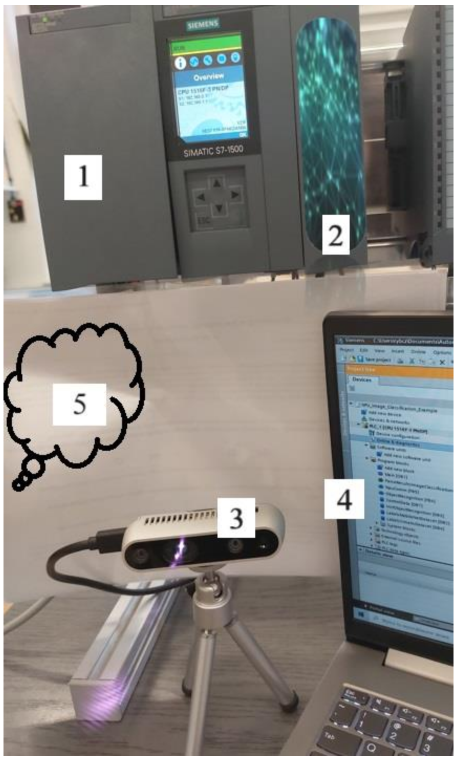


Figure 12. Research station: 1 S7-1516TF, 2 – NPU module with model learning (Inception), 3- Camera RealSense, 4 – Komputer with TIA Portal for PLC, 5 – virtual machine with Keras.

5. Discussion

The task given in the article, among other things, was to recognise the next steps needed to create a model based on the uploaded libraries, including support for the Jupyter environment with the ability to input a model with the extension: .blob to an SMC card integrated with an NPU module compatible with the Siemens S7-1516 TF controller. One of the first problems was the lack of file transfer between the Windows host system and the Linux virtual system. The problem was solved by adding a shared folder from within the Oracle VM VirtualBox virtualisation software and then adding permissions for the Linux user to access and edit the folder.

The first neural network model build was for colour recognition. The first test and its results are shown in Table 2 and Figure 5. The results obtained showed that an accuracy of 98% could be achieved for as few as 15 epochs.

Study two investigated the learning of a model of images with a red and blue container. It turned out that training such a model is prone to random drops in effectiveness despite a single peak. In contrast, the model with a baseline number of training images of 50 and 100 retains similar results achieving more than 95% effectiveness in the final epochs of training. The best performance is achieved by the MobileNet model with a dataset of 1000 training images. This model has a high success rate already in the initial training epochs and maintains this for most of the learning time. In the table 3 and in the figure 6, it can be seen that satisfactory accuracy is already achieved with a model based on 100 photos, with 90% accuracy between 10 and 15 epochs. In addition, it is worth emphasising that the number of images and their degree of sophistication is important. With too few epochs, the model has low efficiency, while with too many epochs the result may not improve sufficiently.

The third measurement was based on the analysis performed on the results obtained in study one and two. It was noted that with 100 images for 10 epochs, an accuracy of 80-90% could be obtained. On this basis, it was decided to investigate a third training base based on three CNN models. Due to the parameters of the computer on which the virtual disk was built, the only model choices were Inception and Efficient. The trained models were built based on 50 images of a 'recognised object' and a 'non-recognised object'. Promising results with an accuracy of 75% were obtained for the analysis of 10 epochs for the Inception model. In contrast, the Efficient and MobileNet models oscillated between 50 and 60 % accuracy. In this situation, a study was carried out based on the model training time, which is also relevant for industrial control process analyses. It can be seen here that the best time was achieved by the MobileNet model, while Efficient and Inception turned out to have 80% more model training time.

6. Conclusions

The research performed was intended to show, firstly, that any model can be implemented using the Keras library, but secondly, that the learning of training models is closely related to the computer hardware on which the analysis was performed. The choice of each training model affects the effectiveness of the artificial neural network and its capabilities. Depending on the desired effects and the trainability, the ultimate goal of learning the models must be taken into account. With fewer images, the number of epochs should be increased until the model achieves consistent and sufficiently high accuracy. Unfortunately, with such a large number of training photos, the model may remain inaccurate even at 100% efficiency, as it will learn to recognise specific photos instead of the actual object. On the other hand, the collection of a large database of training images may require more work, resulting in increased training time for the artificial neural network. It should be noted that the size of the set of training images significantly affects the progress of training the network, but also slows it down. Too few images rely on the randomness of the learning. A large number of images improves the performance of artificial intelligence models through a larger number of data samples.

The results presented were for image recognition based on a model proposed by the manufacturer and two models proposed by the authors. The results show that, for low computer requirements, it is possible to try to teach AI to recognise specific objects at an accuracy level of up to

90%, in a short time (in this example, about 180 s). The chosen models are about training time and depend on the number of epochs, but are still a very convenient tool that could have great potential in the industrial sector. This is made possible by an NPU module that is compatible with the controller, while integrated with the learned AI model. It also shows how artificial intelligence, together with artificial neural networks, can be used to optimise the production process and is no longer just a theoretical part locked away in computer simulations, but an opportunity to introduce it into Industry 4.0. This is demonstrated by a study that has been carried out, which makes it possible to verify the CNN model, based on a PLC. Further research can demonstrate how the programme uses the information fed from the NPU to control the real object, the pneumatic actuator, the actuator or the robot arm.

Author Contributions: Conceptualization, M.R. and K.K.; methodology, M.R.; software, K.K; validation, K.K., M.R..; formal analysis, M.R..; investigation, K.K.; resources, M.R..; data curation, K.K.; writing—original draft preparation, M.R.; writing—review and editing, M.R.; visualization, M.R.; supervision, M.R.,K.K.; project administration, M.R.; funding acquisition, M.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded as part of a research project in the Marine Electrical Engineering Faculty, Gdynia Maritime University, Poland, No. WE/2023/PZ/03, entitled “New methods of controlling the motion of an autonomous ship in open and restricted waters”.

Acknowledgments: Special thanks to the company “Siemens”—headquartered in Warsaw—for providing documentation for the NPU module. Subsidized by the Poland Minister of Education and Science with funds from the state budget as part of the program “Student Scientific Circles create innovations”. SKN/SP/495972/2021.

Conflicts of Interest: “The authors declare no conflict of interest.”.

References

1. https://typeset.io/ Available online:19.10.2023
2. Quan, Zhang. Convolutional Neural Networks. 2018
3. Xujing, Yao., Xinyue, Wang., Shuihua, Wang., Shuihua, Wang., Shuihua, Wang., Yudong, Zhang., Yudong, Zhang. A comprehensive survey on convolutional neural network in medical image analysis. Multimedia Tools and Applications, 2020.
4. Junhai, Zhai., Liguang, Zang., Su-Fang, Zhang. Some Insights Into Convolutional Neural Networks. 2017
5. Kai, Zhao., Sheng, Di., Sihuan, Li., Xin, Liang., Yujia, Zhai., Jieyang, Chen., Kaiming, Ouyang., Franck, Cappello., Zizhong, Chen. FT-CNN: Algorithm-Based Fault Tolerance for Convolutional Neural Networks. IEEE Transactions on Parallel and Distributed Systems, 2021
6. Kai, Zhao., Sheng, Di., Sihuan, Li., Xin, Liang., Yujia, Zhai., Jieyang, Chen., Kaiming, Ouyang., Franck, Cappello., Zizhong, Chen. FT-CNN: Algorithm-Based Fault Tolerance for Convolutional Neural Networks. IEEE Transactions on Parallel and Distributed Systems, 2021
7. Mandar, Kalbande., Punitkumar, Bhavsar. Performance Comparison of Deep Spiking CNN with Artificial Deep CNN for Image Classification Tasks. 2022
8. M, Kavitha. Analysis of DenseNet -MobileNet-CNN Models on Image Classification using Bird Species Data. 2023
9. Kadir, Sabanci. Benchmarking of CNN Models and MobileNet-BiLSTM Approach to Classification of Tomato Seed Cultivars. Sustainability, 2023
10. Fauzan, Masykur., Mohammad, Bhanu, Setyawan., Kuntang, Winangun. Epoch Optimization on Rice Leaf Image Classification Using Convolutional Neural Network (CNN) MobileNet. CESS (Journal of Computer Engineering, System and Science), 2022
11. A, M, Rafay., Waqar, Hussain. EfficientSkinDis: An EfficientNet-based classification model for a large manually curated dataset of 31 skin diseases. *Biomedical Signal Processing and Control*., **2023**
12. A. A, A. Amrutesh, G. B. C G, A. R. K P and G. S, "EfficientNet Models for Detection of Anemia Disorder using Palm Images," 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2023, pp. 1437-1443, 2023.
13. B., Srinivasa, Rao. (2023). EfficientNet - XGBoost: An Effective White-Blood-Cell Segmentation and Classification Framework. Nano Biomedicine and Engineering, 2023.

14. Hong, Zhang., Daniel, Wang., Fan, Fan., Lei, Shu. (2023). EfficientNet-based electromagnetic attack on AES cipher chips. doi: 10.1117/12.2678839
15. Steven, E., Lindow. (2023). Deep CNN-Based Facial Recognition for a Person Identification System Using the Inception Model. doi: 10.1007/978-3-031-29265-1_11
16. Syed, Umar, Amin., Hamdi, Altaheri., Ghulam, Muhammad., Mansour, Alsulaiman., Wadood, Abdul. (2021). Attention based Inception model for robust EEG motor imagery classification. doi: 10.1109/I2MTC50364.2021.9460090
17. Ivandi, Christiani, Pradana., Eko, Mulyanto., Reza, Fuad, Rachmadi. (2022). Deteksi Senjata Genggam Menggunakan Faster R-CNN Inception V2. Jurnal Teknik ITS, doi: 10.12962/j23373539.v11i2.86587
18. Edge Intelligence with Light Weight CNN Model for Surface Defect Detection in Manufacturing Industry. Journal of Scientific and Industrial Research, 2023
19. D., Venkatesan., Senthil, Kumar, Kannan., Muhammad, Arif., Muhammad, Atif., Arulkumaran, Ganeshan. (2022). Sentimental Analysis of Industry 4.0 Perspectives Using a Graph-Based Bi-LSTM CNN Model. Mobile Information Systems, doi: 10.1155/2022/5430569
20. Toward Performing Image Classification and Object Detection With Convolutional Neural Networks in Autonomous Driving Systems: A Survey. IEEE Access, 2022, doi: 10.1109/access.2022.3147495
21. Pratik, Yadav., Naveen, Madur., Vishwajeet, Rajopadhye., Mr., Shrikant, Salatogi. Review on Case Study of Image Classification using CNN. International Journal of Advanced Research in Science, Communication and Technology, (2022). doi: 10.48175/ijarsct-5094
22. E. Solowjow et al., "Industrial Robot Grasping with Deep Learning using a Programmable Logic Controller (PLC)," 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), Hong Kong, China, 2020, pp. 97-103.
23. Monika, Rybczak., Natalia, Popowniak., Krystian, Kozakiewicz. Applied AI with PLC and IRB1200. Applied Sciences, 2022
24. Parisi, L.; Ma, R.; RaviChandran, N.; Lanzillotta, M. hyper-sinh: An accurate and reliable function from shallow to deep learing in TensorFlow and Keras. *Mach. Learn. Appl.* **2021**, *6*, 100112.
25. Haghighat, E.; Juanes, R. SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks. *Comput. Methods Appl. Mech. Eng.* **2021**, *373*, 113552.
26. PS Janardhanan. Project repositories for machine learning with TensorFlow. *Procedia Comput. Sci.* **2020**, *171*, 188–196.
27. https://keras.io/api/applications/ avaible 10.09.2023
28. Salman, Khan., Hossein, Rahmani., Syed, Afaq, Ali, Shah., Mohammed, Bennamoun. A Guide to Convolutional Neural Networks for Computer Vision. 2018
29. Aghdam, H. H., & Heravi, E. J. Guide to convolutional neural networks. New York, NY: Springer, 10(978-973), 51., 2017
30. https://docs.openvino.ai/2023.1/home.html avaible 02.04.2023
31. https://cache.industry.siemens.com/dl/files/877/109765877/att_979771/v2/S71500_tm_npu_manual_en-US_en-US.pdf avaible 10.01.2023