algorithms

MDPI

Article

Deep Machine Learning of MobileNet, Efficient, and Inception Models

Monika Rybczak * and Krystian Kozakiewicz

Faculty of Marine Electrical Engineering, Gdynia Maritime University, Morska 83, 81-225 Gdynia, Poland; 44782@student.umg.edu.pl

* Correspondence: m.rybczak@we.umg.edu.pl

Abstract: Today, specific convolution neural network (CNN) models assigned to specific tasks are often used. In this article, the authors explored three models: MobileNet, EfficientNetB0, and InceptionV3 combined. The authors were interested in investigating how quickly an artificial intelligence model can be taught with limited computer resources. Three types of training bases were investigated, starting with a simple base verifying five colours, then recognizing two different orthogonal elements, followed by more complex images from different families. This research aimed to demonstrate the capabilities of the models based on training base parameters such as the number of images and epoch types. 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 the different models based on the TensorFlow and Keras libraries in the Jupiter environment based on the Python programming language. An artificial intelligence model with a combination of MobileNet, proposed by Siemens, and Efficient and Inception, selected by the authors, allows for further work to be conducted on image classification, but with limited computer resources for industrial implementation on a programmable logical controller (PLC). The study showed a 90% success rate, with a learning time of 180 s.

Keywords: artificial intelligence; deep learning; MobileNet; EfficientNetB0; Inception

Citation: Rybczak, M.; Kozakiewicz, K. Deep Machine Learning of MobileNet, Efficient, and Inception Models. *Algorithms* **2024**, *17*, 96. https://doi.org/10.3390/a17030096

Academic Editor: Stefano Mariani

Received: 30 November 2023
Revised: 9 January 2024
Accepted: 12 January 2024
Published: 22 February 2024

1. Introduction

As reported by [1], 74,703 publications have been published about convolutional neural networks, receiving 2,050,983 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 a 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 for image classification, including face recognition, were demonstrated in [4]. 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. In CNNs, activation functions, convolution operations, merging, and training data are significant. The second aspect of classical neural networks is reliability in security systems. The Python programming language 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 pre-processing 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. Paper [6] discusses a performance comparison of spiking CNNs and 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 paper [7], MobileNet was shown to provide better accuracy compared to other models such as DenseNet and

traditional CNN models. In an example 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 [8]. In a study of rice plant leaf classification, the MobileNet model, with 150 epochs, resulted in the highest accuracy [9]. 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 diagnosis [10], haemoglobin level classification for anaemia diagnosis [11], or white blood cell classification [12]. 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%. In medical applications, several papers, including [13], have proposed a CNN-based brain tumour classification model (BCM-CNN) based on CNN hyperparameter optimisation using an adaptive dynamic sinusoidal cosine grey wolf optimiser (ADSCFGWO) algorithm. The training model was built using Inception-ResnetV2. This model uses common pre-trained models (Inception-ResnetV2) to improve brain tumour diagnosis, resulting in binary 0 or 1 (0: normal, 1: tumour). The results of these experiments showed that, as a classifier, BCM-CNN achieved an accuracy of 99.98% in the BRaTS 2021 dataset. More than one learning model can be used for research. For example, paper [14] proposes a technique to segment organs of the gastrointestinal tract (small intestine, large intestine, and stomach) to help radio oncologists treat cancer patients more quickly and accurately. The proposed model uses the segmentation of small-size images to extract local features more efficiently. It uses six transfer learning models as the backbone of U-Net topology. The six transfer learning models used are Inception V3, SeResNet50, VGG19, DenseNet121, InceptionResNetV2, and EfficientNet B0. The results show that the suggested model outperformed all the other transfer learning models. In the industrial sector, a solution has been proposed by the authors of [15], where the learning model directed electromagnetic leakage information in AES encryption chips, learning the attack response model of cryptographic FPGA chips. Another proposal for the CNN model is Inception, an example of which is the contactless identification of people based on facial features read from an image, where the validation accuracy level was 99.7% [16]. Interesting results were reported in paper [17], where the Inception model was used to accurately decode and recognise EEG motor images. In paper [18], it was shown that it is possible to recognise knife-type hand-held weapons carried by armed individuals from digital images with an accuracy of 87%. Study [19] used synthetic and real images applied to CNNs to classify warehouse items. The AI model was based on a combination of DenseNet and Resnet pipelines for colour and depth images and proved outperformance in terms of accuracy and precision rates compared to single CNNs, achieving a 95.23% accuracy.

The study of convolutional networks has been transferred to industrial conditions. For example, a lightweight model, CondenseneTV2, was proposed to identify surface defects during manufacturing and successfully detected faults during a low-frequency operation on edge equipment [20]. 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 [21]. Another example of AI is research of a chemical wastewater treatment plant, which used a large database collected over a period of 20 months, including data such as temperature, machine operation, and water purity. Three algorithms were used to predict wastewater, and these were support vector regression (SVR), long short-term memory (LSTM) neural network, and gated recurrent unit (GRU) neural network. The experimental results showed that the GRU model performed better (MAPE = 10.18%, RMSE = 35.67%, MAE = 31.16%) than LSTM and SVR [22]. A similar theme is addressed in paper [23], where raw data, generated by an environmental Internet of Things (EIoT) platform, part of a real case study implemented in Briatico (Italy), were collected and hosted on a server that can process and manage real-time information about the plant. Paper [24] 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 [25,26]. For this task, the manufacturers proposed a specialised artificial intelligence module enabling the inclusion of an artificial intelligence model algorithm based on CNN architecture.

In this paper, 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 CNN architecture, and 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. This research is the first step to realize image recognition by a real object configured with an S7-1500 family controller compatible with the AI module and Intel RealSeans camera.

2. Materials and Methods

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

This study deals with the training of 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 POP!_OS [27,28].

The artificial neural network was built in a model-training environment on a virtual disk in Linux and was transferred to an SMC card, a Siemens Simatic memory card. The card allows data stored on it to be used in Siemens products, such as the NPU artificial intelligence module used in this project [26,29]. This means that the learned model can be further interpreted in the 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 Python calculations, it needs to be converted to a .blob type. This is because a .pb model recognises static images, whereas the camera in the test bench uses real-time image data reading. It is the .blob extension that allows such an image to be processed for an artificial neural network model. The testbed was built on a virtual machine running the Linux Pop!_OS operating system. The TensorFlow library [30–32] was used, on which neural network models were trained. In addition, AI libraries were used, Jupyter Notebooks on the first docker and OpenVINO [27] on the second docker. Oracle VM VirtualBox was used to run the virtual disk. The NPU module was equipped with a chip that contains 16 low-power programmable SHAVE (Streaming Hybrid Architecture Vector Engine) cores and an additional accelerator for over building deep neutral network structures called Intel Movidius MyriadTMX [28]. In this configuration, it is possible to upload the learned neural network to an SMC card which can be inserted into the artificial intelligence module and is recognised by the PLC. The main purpose of the applied classification is the intelligent recognition of specific objects and visual quality control.

The proposed testbed allows for the following:

- Python programming using TensorFlow and Keras libraries to create and teach an artificial intelligence model;
- Models to be produced 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:

- First, 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 between the number of epochs and the effectiveness of artificial intelligence models is investigated. Additionally, the rationale for selecting the correct number of epochs is determined.
- The second relates to images of a cuboid, called container in the paper, in one of two colours, red or blue.
- Third, a collection of images of various elements, including electrical parts, where the model is supposed to learn correct and incorrect elements in order to possibly select the correct element in a further work.

Several functions [33,34] were used to build the artificial intelligence model, allowing the AI model parameters to be refined:

- ReLU: Rectified Linear Unit activation function in neural networks. Mathematical notation:

$$f(x) = \max(0, x) \tag{1}$$

meaning that if f(x) is positive (the graphical 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 is a 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. Mathematical notation:

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

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, image and kernel are finite-dimensional matrices, so the sum is conducted only by finite dimensions. In the example presented here, the convolution kernel has a matrix of (3×3).

- Maxpooling2D: a technique that allows for the selection of a maximum value from a specific region in the input matrix otherwise known as a feature map. Mathematical notation:

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

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; and pooling "stride" is the step by which the window is moved during the "pooling" operation. 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 helps to prevent overfitting of the model to 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:

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

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

- GlobalAveragePooling2D is a layer often used before the last densely connected layer (Dense) in CNN models; it reduces dimensionality and prevents overfitting. Mathematical notation:

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

where output (*c*) is the output value for 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 (Algoritm 1).

Algorithm 1 general working diagram of the presented system used during model learning.

Algorithm 1: 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**

This study includes a verification of the three artificial intelligence models implemented on a virtual disk operating on TensorFlow and Keras libraries. The Siemens algorithm, based on MobileNet as designed by the manufacturer, was investigated. Based on the selected kernel parameters and selection of databases, as well as epoch analysis, we decided to verify the performance of the EfficenNet and Inception models. Website [35] shows the main performance characteristics for 38 different models. As several architectures deal with RGB images, we decided to try to compare the MobileNet model with the VGG model. Due to virtual disk hardware limitations, we decided to investigate the EfficenNetB0 and InceptionV3 models specifically. The example introduces hyperparameters, which are included in Table 1.

Table 1. Use of hyperparameters in AI model.

Hyperparameter:	Value:
Kernel size	3×3
Image size	224×224
Batch size	256
Pool size	(2,2)
Stride	2
Optimizer	Adam
Activation function	ReLU
Learning rate	8×10^3

3.1. First Training Database—Simple Database

Each colour was duplicated four times, giving a total of 20 training images per colour. Models differ only in the number of learning epochs. Hence, with the same number of epochs, there should be a similar value for the effectiveness of the artificial neural network, meaning 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 artificial neural networks. Figure 1 shows an example of images generated by training data generator for the first database.

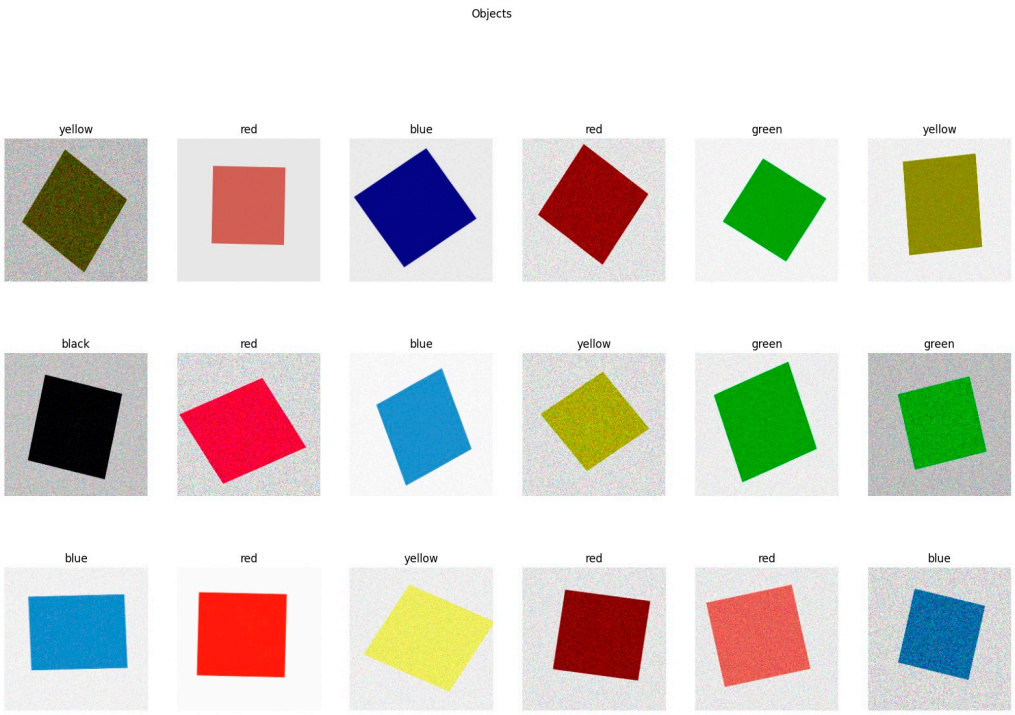

Figure 1. Example result of a training data generator—database for determining five colours.

Figure 1 shows the end result of a training image generator. After adding deviation and noise, new training images used for learning were created.

3.2. Second Training Database—Database with Cuboid

This study was intended to show how a learned model would behave, where the training database are images of blue and red cuboids, referred to as containers.

In this training database, there are actual images of two container models with a background and the object itself changing its rotation and illumination. Figure 2 shows a high accuracy model prediction with a training image base of 1000.

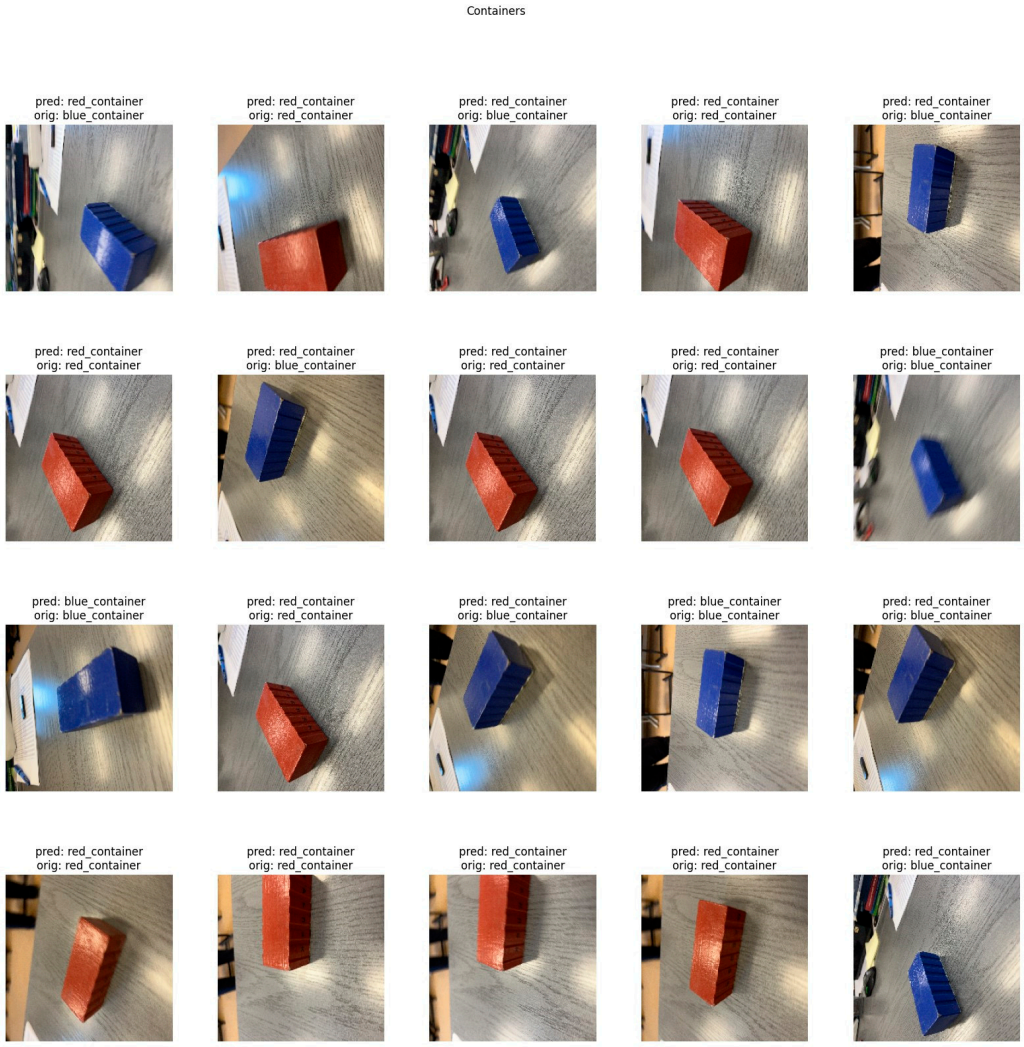

Figure 2. Examples of model predictions based on inference, where 'pred' denotes model prediction; 'orig' denotes the actual class of image.

3.3. Third Training Database—Database with Different Objects

The third training database is made up of several different objects. The model has to learn to distinguish between pictures of electrical elements and other items. This time the model has to be able to distinguish between the designated elements, electrical elements, and other coloured elements completely unrelated to the training database.

Figure 3 shows a prediction with accuracy built from 100 training images, where there are exactly 50 images of correct elements and 50 images of other elements.

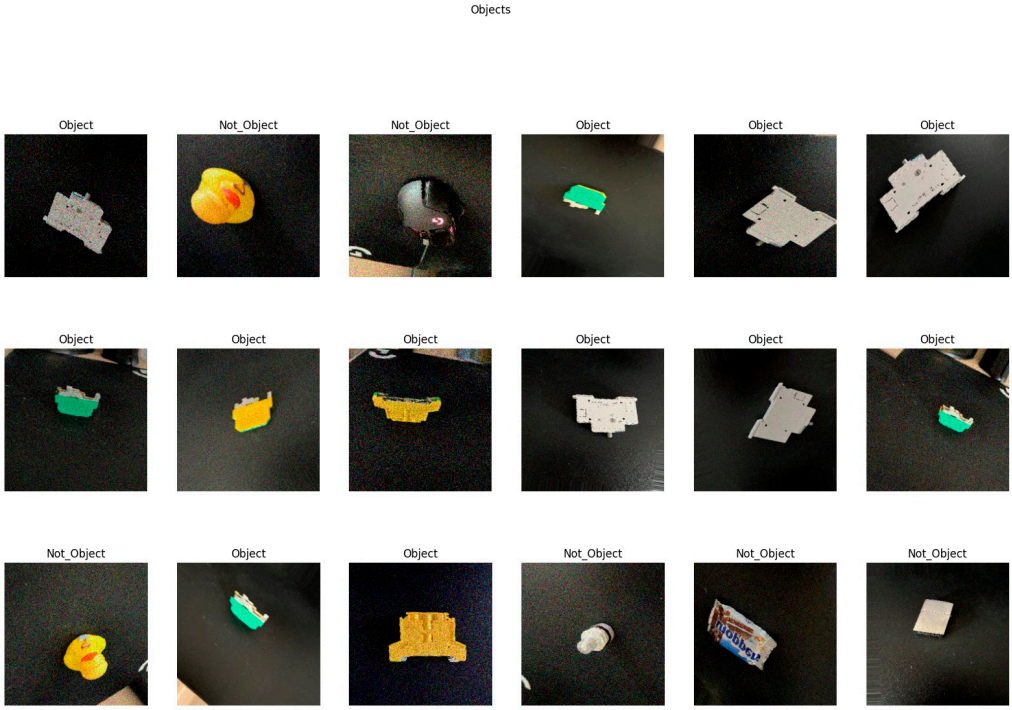

Figure 3. Example model predictions based on inference, where 'Object' denotes a valid element; 'Not_object' denotes an element outside the set.

4. Results—Verification of Tree CNN Models

The results presented here are based on three training databases. AI models used in the first two examples are based on MobileNet, while the third was built based on three different models, MobileNet, EfficienNetB0, and InceptionV3. Accuracy verification of the first two models based on MobileNet was performed first. It was next performed in the third database, where information about the object being searched for in the image was to be included with an additional indication of which object in the image was wrong. After analysing the performance of MobileNet models, we decided to perform the third verification for two additional models based on EfficienNetB0 and InceptionV3 in addition to MobileNet.

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 differences between those factors based on specific sets of images and accuracy.

4.1. Artificial Intelligence Models Built with MobileNet

4.1.1. First Training Database—Simple Data Base

The first study involved learning a model to recognise five computer-entered colours. These were not colours in actual photographs but colours entered into a computer program.

Firstly, an attempt was made to check the accuracy of the learned model for 10, 15, and 20 epochs successively. This work showed that the studied model could be configured with sufficient accuracy in as few as 15 epochs. The results are shown in Table 2 and Figure 4 below.

Table 2. Effectiveness of learned models across epochs.

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

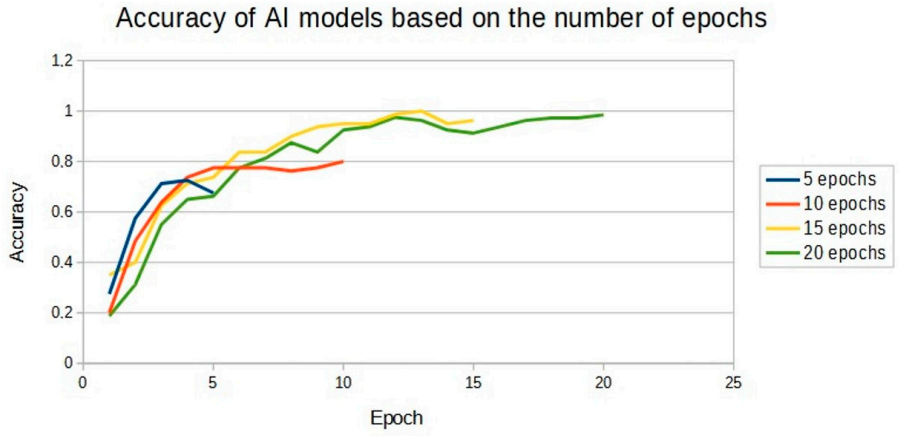

Figure 4. Accuracy graph for models based on different numbers of learning epochs.

4.1.2. Second Training Database—Database with Cuboid

Four models, each trained for 20 epochs, were created to discuss the results. The models differed in the number of training photos. Images remained the same, but the numbers were duplicated. The training sets were divided into four ranges, 10, 50, 100, and 1000 images, and are presented in Table 3.

Table 3. Effectiveness of learned models with different numbers of training images across 20 epochs.

Epoch	Accuracy of Learned Model 10 Images	Accuracy of Learned Model 50 Images	Accuracy of Learned Model 100 Images	Accuracy of Learned 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 5 shows an increasing accuracy trend, similar for all models. The accuracy value depends on both the number of epochs as well as the number of images. However, the impact of image numbers outweighs the impact of epochs but at the cost of increased learning time. It is therefore necessary to select an appropriate number of both epochs and images for each dataset. According to the graph in Figure 5, it can be seen that too few training images, in this case 10, do not allow for the neural network to learn to recognise image classes effectively regardless of the number of epochs. The accuracy of the final model, with 1000 images, reaches 100% in as few as five epochs and is the only one able to maintain such a high performance. Unfortunately, this happens at the expense of extended learning time caused by increasing the number of steps per each epoch due to the number of training images.

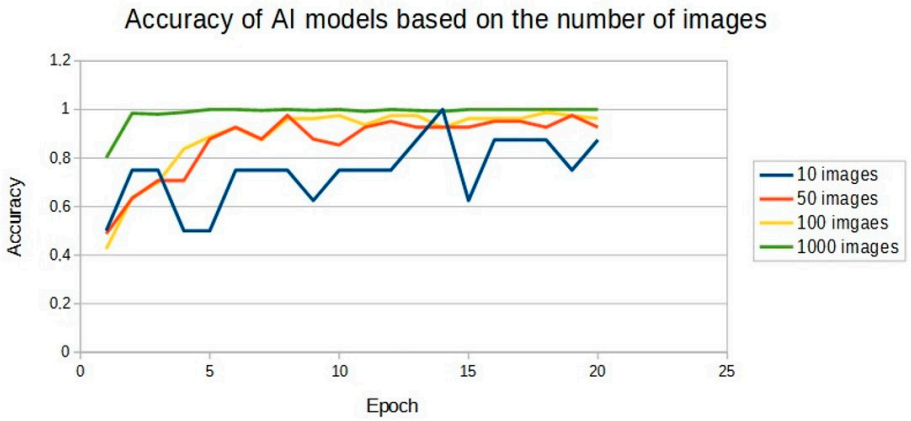

Figure 5. Accuracy graph for models based on different numbers of images.

4.2. Three Models: MobileNet, EfficientNetB0, and InceptionV3

Third Training Database—Database with Different Objects

Previous results have shown that MobileNet models have a high accuracy of 98% and even up to 100% under certain favourable conditions. Therefore, due to the scope of the tasks to be performed, we decided to select 2 more models, out of 38 of them, proposed in the Keras library [33]. In addition to MobileNet, these were the EfficientNetB0 and InceptionV3 models based on the TensorFlow and Keras library functions, such as Convolution2D, Batch Normalization, ReLU, and MaxPooling2D. The choice of models depended on the hardware parameters of the PC and virtual disk. The PC parameters were Intel(R) Core(TM) i5-9400F CPU @2.90 GHz ×64 processor; 16.0 GB installed RAM; and 64-bit Operating System. Virtual disk limitations: 10 GB RAM and 4 of 6 CPU cores used. We decided to verify the third database, built from 100 images, on 3 models across 10 epochs, as see in Table 4:

- MobileNet;
- EfficieNetB0;
- InceptionV3.

Table 4. Effectiveness of three learned models with 100 images across 10 epochs.

Epoch	Accuracy of MobileNet Model	Accuracy of EfficientNetB0 Model	Accuracy of InceptionV3 Model
1	0.4916	0.5166	0.4791
2	0.5041	0.5083	0.5833
3	0.5041	0.4791	0.5166
4	0.5000	0.4541	0.5875
5	0.5333	0.5083	0.6083
6	0.4500	0.5083	0.7000
7	0.5125	0.5166	0.7250
8	0.5916	0.5500	0.6875
9	0.5458	0.5041	0.7166
10	0.5250	0.5208	0.7458

The graph in Figure 6 shows the accuracies of the three models carried out for learning on the third dataset, namely the images of the electrical elements mixed with other objects.

Due to the interesting results obtained for the InceptionV3 model, the learning times for all three models were compared, as see in Figure 7.

It can be seen that InceptionV3 had the best accuracy of the three models, but this was at the cost of the longest learning time. While both MobileNet and EfficientNetB0 showed comparable efficiency, the former achieved this in a significantly shorter learning time.

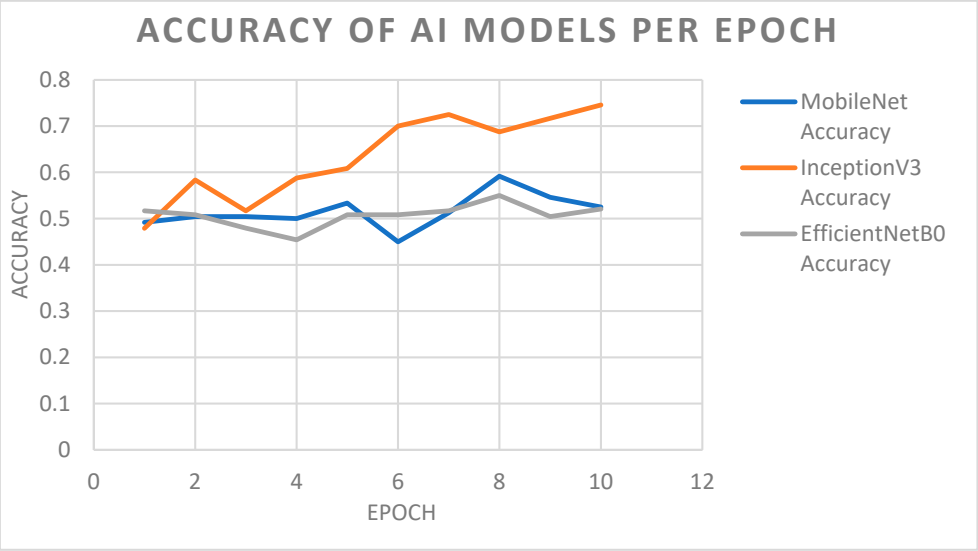

Figure 6. Accuracy graph for models based on different numbers of learning epochs.

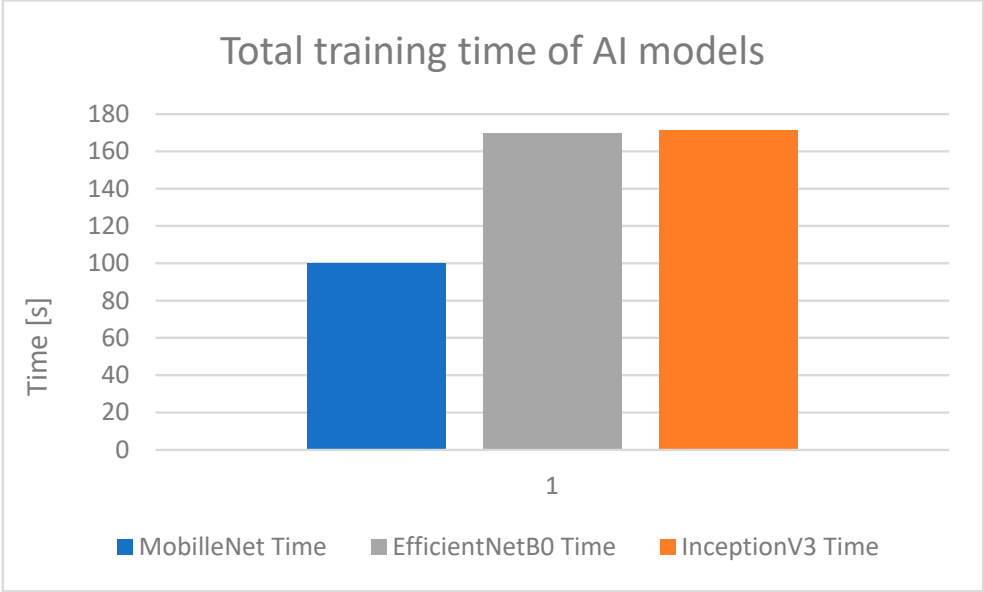

Figure 7. Total training time of three models.

4.3. Verification of InceptionV3 Model on the PLC Testbed

Teaching the model to recognise elements was implemented to the PLC testbed for further research. At first, the model was in a file format with .blob extension. The file was then saved to an SMC memory card on the PC where the model was taught. This memory card was then transferred to the NPU module and, following the steps suggested by Siemens (produced NPU module [28]), the S7-1500 family PLC reads the parametres from card [see Figures 8–10]. These parameters are not string variables, instead they are numerical information stored in the range from 0 to 1, where a result closer to 1 means that an element taught by the AI model is detected in the image.

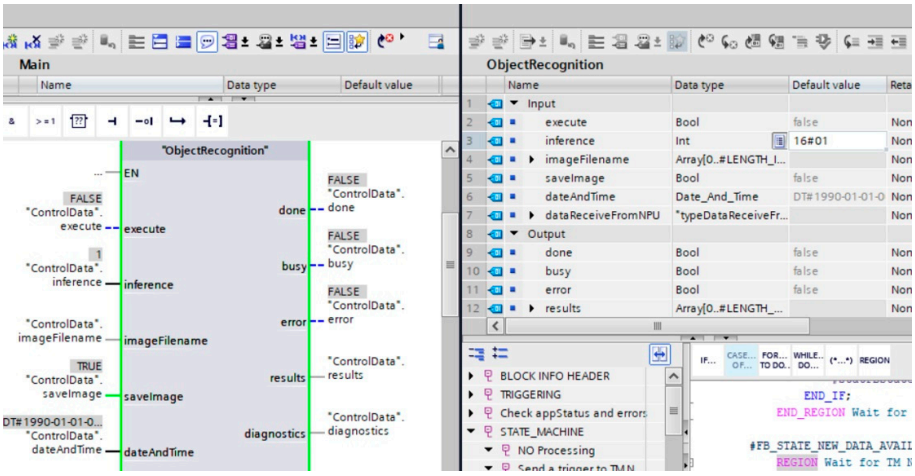

Figure 8. Part of the PLC program in LAD and SCL programming languages.

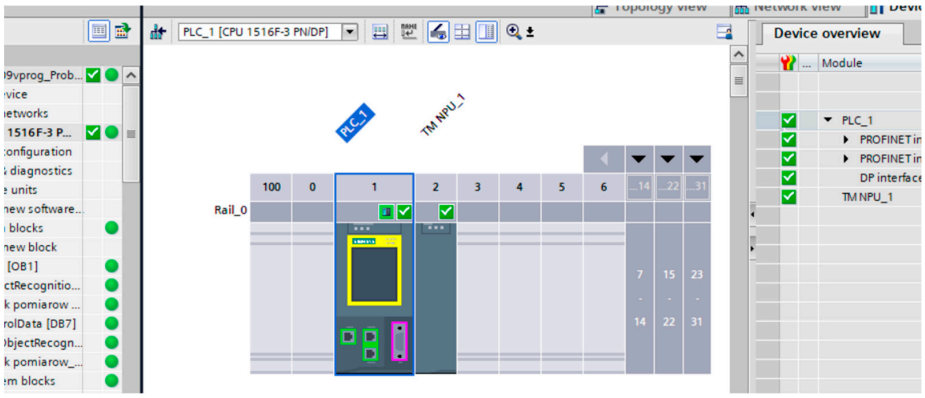

Figure 9. Configuration of S7-1516TF PLC with NPU module as shown in TIA Portal v16 software.

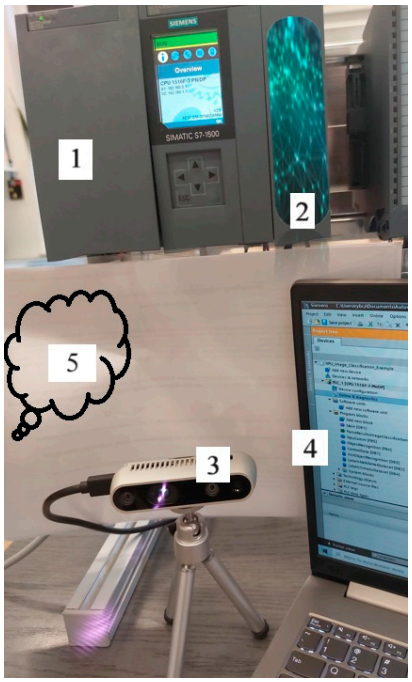

Figure 10. PLC testbed configuration: 1—S7-1516TF PLC, 2—NPU module with InceptionV3 learning model, 3—RealSense camera, 4—PC with TIA Portal software for PLC programming, 5—virtual machine with Keras toolbox.

5. Discussion

One of the goals of the research presented in this paper was to recognise the steps needed to create a model, based on the uploaded libraries, which included support for the Jupyter environment. Additionally, we were required to have the ability to input a model with .blob extension to an SMC memory card that could be inserted into an NPU module compatible with Siemens S7-1516TF PLC. One of the first problems was the lack of file transfer possibilities between the Windows host system and Linux virtual system. This problem was solved by adding a shared folder from within the Oracle VM VirtualBox virtualisation software (Version 15.x) and then adding permissions for the Linux user to access and edit this folder.

In the first study, a neural network model was built for colour recognition, and its tests results are shown in Table 2 and Figure 4. These showed that an accuracy of 98% could be achieved in as few as 15 epochs.

Study two investigated how a model was taught to recognise images of red and blue containers. The research showed that training such a model is prone to random drops in accuracy despite a single peak. In contrast, models with 50 and 100 training images retained similar, steady results, achieving more than 95% accuracy in the final epochs of training. The best accuracy was achieved by the MobileNet model with a dataset of 1000 training images. This model had a high accuracy already in the initial training epochs and maintained this for most of the learning time. Table 3 and Figure 5 show that a satisfactory accuracy of 90% was already achieved with a model based on 100 photos at 10 to 15 epochs. It is worth emphasising that the number of images and their degree of sophistication are important. Additionally, with too few epochs, the model has low accuracy, while too many epochs extend learning time but may not improve accuracy sufficiently.

The third study had its initial conditions based on the analysis of the results obtained in studies one and two. It was noted that with 100 images and 10 epochs, an accuracy of 80–90% could be obtained. On this basis, we decided to investigate a third training base with three CNN models. Due to the virtual machine PC parameters, model choices were limited to MobileNet, EfficientNetB0, and InceptionV3. The trained models were built using 50 images of a 'recognised object' and 'non-recognised object' each at 10 epochs. Promising results, with an accuracy of 75%, were obtained for the InceptionV3 model, while the accuracies of the MobileNet and EfficientNetB0 models oscillated between 50 and 60%. In comparison, model training time, which is also a relevant factor for industrial process control, was analysed. It was observed that the best learning time was achieved by the MobileNet model, while EfficientNetB0 and InceptionV3 required around 80% more training time.

6. Conclusions

The performed research was intended to show, firstly, that any model can be implemented using the Keras library, and secondly, that the learning of training models is closely related to the computer hardware on which the analysis is performed. The choice of each training model affects the accuracy of the artificial neural network and its capabilities. Depending on the desired effects and 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% accuracy as it will learn to recognize specific photos instead of the actual object. On the other hand, collecting 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 increases the progress of training the network, but at the same time it slows it down. Too few images result in randomness of learning. A large number of images improves the accuracy of an artificial intelligence model through a larger number of data samples.

The presented results showed image recognition based on one model proposed by the AI module manufacturer (MobileNet) and two models proposed by the authors (Efficient and Inception). The authors were interested in selecting appropriate models and teaching AI using a PC with relatively low parameters, as presented in Section 4.3. The results have shown that, with low computer parameters, it is possible to teach an AI to recognise specific objects at an accuracy level of up to 90% in a short time (in this example, about 180 s).

The selected models are dependent on the number of training images and epochs, and training time is a significant factor, but still, they are a very convenient tool that could have great potential in the industrial sector. This is made possible by the NPU module that is compatible with a PLC controller, where it is integrated with the learned AI model. It also shows how artificial intelligence, together with artificial neural networks, can be used to optimise production processes and is no longer just a theoretical part locked away in computer simulations but rather an opportunity to introduce it into Industry 4.0. This is demonstrated by the study carried out in this paper. Further research could show how the PLC program uses information fed from the NPU to control real objects like pneumatic actuators or robot arms.

Author Contributions: Conceptualization, M.R. and K.K.; methodology, M.R.; software, K.K; validation, K.K. and 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. and 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/2024/PZ/03, entitled "New methods of controlling the motion of an autonomous ship in open and restricted waters".

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

CNN	Convolution neural network
FPGA	Field-programmable gate array
GRU	Gated recurrent unit
HPC	High-performance computing
MAPE	Mean absolute percentage of error
MAE	Mean absolute error
LSTM	Long short-term memory
PLC	Programable Logical Control
RMSE	Root mean square error
RGB	Red, green, blue
VGG	Very deep convolution networks
SMC	Simatic memory card
SVR	Support vector regression

References

1. Available online: https://typeset.io/ (accessed on 19 October 2023).
2. Zhang, Q. Convolutional Neural Networks. In Proceedings of the 3rd International Conference on Electromechanical Control Technology and Transportation: ICECTT, Chongqing, China, 19–21 January 2018; Volume 1, pp. 434–439, ISBN 978-989-758-312-4. [CrossRef]

3. Yao, X.; Wang, X.; Wang, S.H.; Zhang, Y.D. A comprehensive survey on convolutional neural network in medical image analysis. *Multimed. Tools Appl.* **2020**, *81*, 41361–41405. [CrossRef]
4. Zhai, J.H.; Zang, L.G.; Zhang, S.F. Some Insights into Convolutional Neural Networks. In Proceedings of the 2017 International Conference on Machine Learning and Cybernetics (ICMLC), Ningbo, China, 9–12 July 2017.
5. Zhao, K.; Di, S.; Li, S.; Liang, X.; Zhai, Y.; Chen, J.; Ouyang, K.; Cappello, F.; Chen, Z. FT-CNN: Algorithm-Based Fault Tolerance for Convolutional Neural Networks. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *32*, 1677–1689.
6. Kalbande, M.; Bhavsar, P. Performance Comparison of Deep Spiking CNN with Artificial Deep CNN for Image Classification Tasks. In Proceedings of the 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 1–3 July 2022.
7. Kavitha, M. Analysis of DenseNet-MobileNet-CNN Models on Image Classification using Bird Species Data. In Proceedings of the 2023 International Conference on Disruptive Technologies (ICDT), Greater Noida, India, 11–12 May 2023.
8. Sabanci, K. Benchmarking of CNN Models and MobileNet-BiLSTM Approach to Classification of Tomato Seed Cultivars. *Sustainability* **2023**, *15*, 4443. [CrossRef]
9. Masykur, F.; Setyawan, M.B.; Winangun, K. Epoch Optimization on Rice Leaf Image Classification Using Convolutional Neural Network (CNN) MobileNet. *CESS (J. Comput. Eng. Syst. Sci.)* **2022**, *7*, 581–591. [CrossRef]
10. Rafay, A.; Hussain, W. EfficientSkinDis: An EfficientNet-based classification model for a large manually curated dataset of 31 skin diseases. *Biomed. Signal Process. Control* **2023**, *85*, 104869. [CrossRef]
11. Amruthamsh, A.; Amrutesh, A.; CG, G.B.; KP, A.R.; Gowrishankar, S. EfficientNet Models for Detection of Anemia Disorder using Palm Images. In Proceedings of the 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 11–13 April 2023; pp. 1437–1443.
12. SivaRao, B.S.S.; Rao, B.S. EfficientNet—XGBoost: An Effective White-Blood-Cell Segmentation and Classification Framework. *Nano Biomed. Eng.* **2023**, *15*, 126–135. [CrossRef]
13. ZainEldin, H.; Gamel, S.A.; El-Kenawy, E.-S.M.; Alharbi, A.H.; Khafaga, D.S.; Ibrahim, A.; Talaat, F.M. Brain Tumor Detection and Classification Using Deep Learning and Sine-Cosine Fitness Grey Wolf Optimization. *Bioengineering* **2023**, *10*, 18. [CrossRef] [PubMed]
14. Sharma, N.; Gupta, S.; Koundal, D.; Alyami, S.; Alshahrani, H.; Asiri, Y.; Shaikh, A. U-Net Model with Transfer Learning Model as a Backbone for Segmentation of Gastrointestinal Tract. *Bioengineering* **2023**, *10*, 119. [CrossRef] [PubMed]
15. Ning, W.X.; Zhang, H.X.; Wang, D.Z.; Fan, F.; Shu, L. EfficientNet-based electromagnetic attack on AES cipher chips. In Proceedings of the International Conference on Cryptography, Network Security, and Communication Technology (CNSCT 2023), Changsha, China, 12 May 2023. [CrossRef]
16. Lindow, S.E. Deep CNN-Based Facial Recognition for a Person Identification System Using the Inception Model. In Proceedings of the International Conference on Cryptography, Network Security, and Communication Technology (CNSCT 2023), Changsha, China, 6–8 January 2023. [CrossRef]
17. Amin, S.U.; Altaheri, H.; Muhammad, G.; Alsulaiman, M.; Abdul, W. Attention based Inception model for robust EEG motor imagery classification. In Proceedings of the 2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Glasgow, UK, 17–20 May 2021. [CrossRef]
18. Pradana, I.C.; Mulyanto, E.; Rachmadi, R.F. Deteksi Senjata Genggam Menggunakan Faster R-CNN Inception V2. *J. Tek. ITS* **2022**, *11*, A110–A115. [CrossRef]
19. Piratelo, P.H.M.; de Azeredo, R.N.; Yamao, E.M.; Bianchi Filho, J.F.; Maidl, G.; Lisboa, F.S.M.; de Jesus, L.P.; Penteado Neto, R.d.A.; Coelho, L.d.S.; Leandro, G.V. Blending Colored and Depth CNN Pipelines in an Ensemble Learning Classification Approach for Warehouse Application Using Synthetic and Real Data. *Machines* **2022**, *10*, 28. [CrossRef]
20. Rani, D.S.; Burra, L.R.; Kalyani, G.; Rao, B. Edge Intelligence with Light Weight CNN Model for Surface Defect Detection in Manufacturing Industry. *J. Sci. Ind. Res.* **2023**, *82*, 178–184.
21. Venkatesan, D.; Kannan, S.K.; Arif, M.; Atif, M.; Ganeshan, A. Sentimental Analysis of Industry 4.0 Perspectives Using a Graph-Based Bi-LSTM CNN Model. *Mob. Inf. Syst.* **2022**, *2022*, 5430569. [CrossRef]
22. Miao, S.; Zhou, C.; AlQahtani, S.A.; Alrashoud, M.; Ghoneim, A.; Lv, Z. Applying machine learning in intelligent sewage treatment: A case study of chemical plant in sustainable cities. *Sustain. Cities Soc.* **2021**, *72*, 103009. [CrossRef]
23. Cicceri, G.; Maisano, R.; Morey, N.; Distefano, S. A Novel Architecture for the Smart Management of Wastewater Treatment Plants. In Proceedings of the 2021 IEEE International Conference on Smart Computing (SMARTCOMP), Irvine, CA, USA, 23–27 August 2021; pp. 392–394. [CrossRef]
24. Turay, T.; Vladimirova, T. Toward Performing Image Classification and Object Detection With Convolutional Neural Networks in Autonomous Driving Systems: A Survey. *IEEE Access* **2022**, *10*, 14076–14119. [CrossRef]
25. Yadav, P.; Madur, N.; Rajopadhye, V.; Salatogi, S. Review on Case Study of Image Classification using CNN. *Int. J. Adv. Res. Sci. Commun. Technol.* **2022**, *2*, 683–687. [CrossRef]
26. Solowjow, E.; Ugalde, I.; Shahapurkar, Y.; Aparicio, J.; Mahler, J.; Satish, V.; Goldberg, K.; Claussen, H. Industrial Robot Grasping with Deep Learning using a Programmable Logic Controller (PLC). In Proceedings of the 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), Hong Kong, China, 20–21 August 2020; pp. 97–103.
27. Available online: https://docs.openvino.ai/2023.1/home.html (accessed on 2 April 2023).
28. Available online: https://cache.industry.siemens.com/dl/files/877/109765877/att_979771/v2/S71500_tm_npu_manual_en-US_en-US.pdf (accessed on 10 January 2023).

29. Rybczak, M.; Popowniak, N.; Kozakiewicz, K. Applied AI with PLC and IRB1200. *Appl. Sci.* **2022**, *12*, 12918. [CrossRef]
30. 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. [CrossRef]
31. 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. [CrossRef]
32. Janardhanan, P.S. Project repositories for machine learning with TensorFlow. *Procedia Comput. Sci.* **2020**, *171*, 188–196. [CrossRef]
33. Khan, S.; Rahmani, H.; Shah SA, A.; Bennamoun, M.; Medioni, G.; Dickinson, S. *A Guide to Convolutional Neural Networks for Computer Vision*; Morgan & Claypool Publishers: San Rafael, CA, USA, 2018.
34. Aghdam, H.H.; Heravi, E.J. *Guide to Convolutional Neural Networks*; Springer: New York, NY, USA, 2017; pp. 973–978.
35. Available online: https://keras.io/api/applications/ (accessed on 10 September 2023).