mathematics

MDPI

Article

A New Loss Function for Simultaneous Object Localization and Classification

Ander Sanchez-Chica [1], Beñat Ugartemendia-Telleria [1], Ekaitz Zulueta [1,*], Unai Fernandez-Gamiz [2] and Javier Maria Gomez-Hidalgo [3]

[1] System Engineering and Automation Control Department, University of the Basque Country (UPV/EHU), Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain
[2] Department of Nuclear and Fluid Mechanics, University of the Basque Country (UPV/EHU), Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain
[3] MERCEDES BENZ España, Las arenas 1, 10152 Vitoria-Gasteiz, Spain
* Correspondence: ekaitz.zulueta@ehu.eus

Abstract: Robots play a pivotal role in the manufacturing industry. This has led to the development of computer vision. Since AlexNet won ILSVRC, convolutional neural networks (CNNs) have achieved state-of-the-art status in this area. In this work, a novel method is proposed to simultaneously detect and predict the localization of objects using a custom loop method and a CNN, performing two of the most important tasks in computer vision with a single method. Two different loss functions are proposed to evaluate the method and compare the results. The obtained results show that the network is able to perform both tasks accurately, classifying images correctly and locating objects precisely. Regarding the loss functions, when the target classification values are computed, the network performs better in the localization task. Following this work, improvements are expected to be made in the localization task of networks by refining the training processes of the networks and loss functions.

Keywords: image classification; object detection; deep learning; deep convolutional neural networks; computer vision; custom training loop

MSC: 49J05; 49J15

Citation: Sanchez-Chica, A.; Ugartemendia-Telleria, B.; Zulueta, E.; Fernandez-Gamiz, U.; Gomez-Hidalgo, J.M. A New Loss Function for Simultaneous Object Localization and Classification. *Mathematics* **2023**, *11*, 1205. https://doi.org/10.3390/math11051205

Academic Editors: Debiao Meng and Shui Yu

Received: 25 January 2023
Revised: 23 February 2023
Accepted: 25 February 2023
Published: 1 March 2023

1. Introduction

Nowadays, robots are essential for the manufacturing industry. The use of robots has helped the manufacturing industry to manufacture products more efficiently, saving both costs and time. Despite the fact that an increasing need for robots has been observed in all industrial sectors in recent years, the electronics industry has been the main customer of industrial robots since 2020, when it overtook the automotive industry. However, the latter still demands 80,000 robots a year; hence, it is still an important sector for robot manufacturers. Industrial robot manufacturers are making every effort to design and develop safe and human-friendly robots. This is spurred on by the fact that small- and medium-sized companies are increasing their use of industrial robots due to the availability of affordable solutions and easy-to-use collaborative robots. Hence, collaborative solutions, where humans and robots work together, are becoming the new frontier in industrial robotics [1,2]. The use of collaborative robots is also supported by the current trend of automation and data exchange in manufacturing industries, also called Industry 4.0 [3].

In the case of the automotive industry, robots are used mainly in the manufacturing process. At the beginning of the 20th century, when chain production was introduced by the Ford Model T, cars were handmade. Nowadays, this process is mainly automatic. However, there are still tasks where humans need to intervene. In this context, collaborative robots can help workers improve the efficiency and reduce the manufacturing faults of production

lines [4]. Robots also help in other areas, such as neurosurgery [5]. Since collaborative robots need to be aware of their surroundings, computer vision can help robots to detect their environment and object positions.

Image classification, object detection and semantic segmentation are the main tasks in computer vision. Since LeCun et al. [6] proposed LeNet-5 in 1998 for document recognition, convolutional neural networks (CNNs) have become the state-of-the-art techniques for these tasks. The explosion of deep learning has led to the need for high-quality, diverse and structured image datasets. In 2010, ImageNet [7] was presented as a solution to this problem. Following the path that started LeNet-5, in 2012, AlexNet [8] won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [9] by outperforming previous image-classifying state-of-the-art techniques. This success led researchers to improve the performance of CNNs. Following this trend of making deeper and larger networks, Zeiler et al. [10] proposed ZFNet to understand and improve the results obtained by Krizhevsky et al. [8]. In 2014, Simonyan et al. [11] proposed a CNN with very small convolution filters to evaluate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting.

To tackle the difficulty of training very deep neural networks, He et al. [12] presented a residual learning framework called Residual Network (ResNet), which allows for an increase in the depth, while maintaining the complexity of the network. ResNet cross-layer connections try to solve the gradient diffusion problem that very deep convolutional networks, such as VGGNet and ZFNet, have. However, training very deep residual networks is extremely costly. Thus, Zagoruyko et al. [13] conducted an experimental study on the architecture of ResNet blocks. Based on this study, they proposed a novel architecture, where they decreased the depth and increased the width of residual networks. The resulting network structures, called wide residual networks (WRNs), overperformed their thin and very deep counterparts.

GoogLeNet [14], also called Inception-v1, combines inception modules with conventional convolution modules to improve the utilization of the computing resources inside the network, hence increasing the depth and width of the network, while keeping the computational budget constant by varying the sizes of the convolution filters. Inception-v4 [15] aggregates residual connections to the inception architecture to accelerate training, while maintaining the accuracy of similarly expensive inception networks. Huang G. et al. [16] proposed the Dense Convolutional Network (DenseNet), which connects each layer to every other layer downstream, reusing the features of all previous layers to strengthen feature propagation and reduce the vanishing gradient problem. Following the trend of dense connectivity, with CondenseNetV2, Yang et al. [17] ensured that each layer simultaneously learned to carry out the following:

1. Selectively reuse the set of the most important features from preceding layers;
2. Actively update the set of preceding features to increase their utility for later layers, achieving promising performance in image classification (ImageNet) and object detection (MS COCO) in terms of both theoretical efficiency and practical speed.

In recent years, large advancements have been made in image classification tasks [18]. Therefore, CNNs have great value when there is a need to identify images. However, normally, this feature is not useful when it is used alone. It can be combined with a region proposal network (RPN) and perform traditional object detection.

The traditional object detection method consists of generating region proposals first using an RPN and then classifying each proposal into different object categories [19]. This is the case of R-CNN [20]. Nevertheless, this process is normally very computationally costly. In order to tackle this issue, different iterations of R-CNN have been proposed. Girshick et al. [21] improved their original R-CNN to be faster and more accurate. Ren et al. [22] improved this by introducing an RPN that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. The Faster R-CNN architecture has achieved good results in object detection tasks. For example,

Fu et al. [23] and Song et al. [24] used the Faster R-CNN based on ZFNet and VGG16, respectively, to detect kiwifruits in order to enable robots to pick them up.

The other object detection method with regard to the task of regression or the classification problem adopts a unified framework to achieve the final results (categories and locations) directly. Redmon et al. [25] predicted bounding boxes and their associated class probabilities directly from full images in one evaluation. They called this new approach to object detection You Only Look Once (YOLO). The Single-Shot Multibox Detector (SSD) [26] discretizes the output space of bounding boxes into a set of default boxes, adjusting them by the scores generated for the presence of each object category in each default box in order to better match the object shape. CenterNet, proposed by Duan et al. [27], presents an efficient solution based on the detection of each object as a triplet of key points rather than a pair, improving both precision and recall.

However, each of these methods has its own issues: the traditional object detection techniques require a high computational power, whereas the single-stage methods do not have the same level of accuracy as the traditional techniques. In 2017, Li et al. [28] proposed a two-stage object detector based on ResNet-101 [12] to address the shortcomings of these types of detectors, that is, the slow speed of these networks due to their heavy-head designs. In 2018, Zhang et al. [29] proposed a novel single-shot-based detector that achieves a better accuracy than the two-stage methods and maintains an efficiency comparable to that of the one-stage methods. Examples of the advancements that have been made in object detection tasks in recent years are in reference [30].

In 2019, EfficientDet [31] proposed a new family of object detectors based on EfficientNet backbones and optimized the weighted bi-directional feature pyramid network (BiFPN) and the compound scaling method. In particular, the model EfficientDet-D7 achieved state-of-the-art results at MS COCO. Another example of this appeared in 2022, when Liu et al. [32] presented a network called ConvNeXts, constructed entirely from standard ConvNet modules. These modules are ResNet modules modernized towards the design of a vision transformer, and they compete favorably with transformers in terms of accuracy and scalability.

Additionally, the networks observed in the literature focus on single-task problems: image classification, object detection, image recognition, etc. To the best of our knowledge, there are no or very few examples of CNNs that have been used to simultaneously perform different tasks. Therefore, we see the need for exploring image classification and object localization tasks using the same CNN. The objective of this article is to determine whether both tasks can be performed accurately with a single CNN. Therefore, we propose a custom evaluation loop that merges the cross-entropy loss (Ex) for the classification task and the half mean square error (mse) for the regression task (object localization). We also compare two different loss functions using different Ex and mse loss proportions and determine which method is the best.

2. Materials and Methods

2.1. Convolutional Neural Network

A CNN is a type of deep neural network that uses convolutional layers to extract feature maps from the input image. Usually, the network consists of one input layer, one or more convolutional layers, one fully connected layer and one output layer [33]. In this case, the network has two fully connected layers at the end of the convolutional layers, separating each one from the main branch. This allows the network to perform two different tasks using the same convolutional layers. At the end of one fully connected layer, a softmax layer is connected. This branch performs the classification task, while the other performs the detection task. In Figure 1, the structure of the network can be seen.

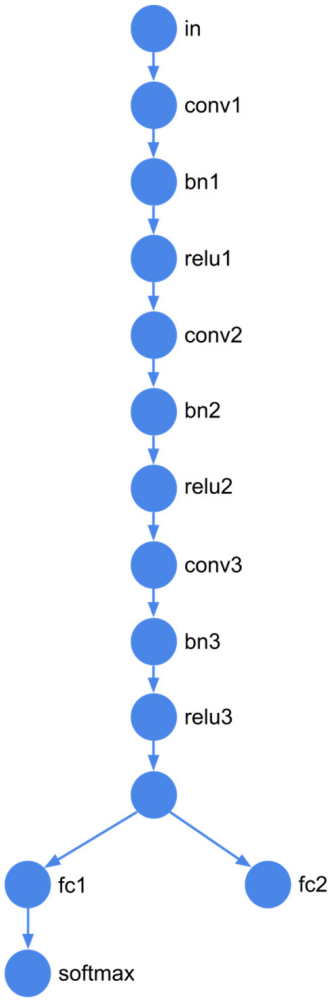

Figure 1. Structure of the proposed CNN.

The input layer in has a dimension of 100 × 100 × 1. Therefore, the input data consist of a single matrix with dimensions of 100 × 100, which contain the value of each pixel in gray-scale from 0 (black) to 255 (white).

The convolutional layer is the specific layer of the CNN. The convolutional equation used is that shown in Equation (1):

$$y_j = \sum(w_{ij} \cdot x_j) + b_j \tag{1}$$

where y_j is the result, w_{ij} is the filter matrix, x_j is the input of the convolutional layer, and b_j is the bias term. In this case, the network features three convolutional layers. The first one has 16 filters with a 5 × 5 size. The second one has 32 filters with a 3 × 3 size. Finally, the third one also has 32 filters with a 3 × 3 size, although it has a stride of one, instead of two like the second layer. Furthermore, the output of this network uses a nonlinear activation function (ReLU), as shown in Equation (2):

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

In order to speed up the training and reduce the sensitivity to network initialization, a batch normalization layer is included between each convolutional layer and the ReLU layer. This is achieved by normalizing a mini-batch of data across all observations for each channel independently. The parameters of the model are listed in detail in Table 1.

Table 1. Parameters and output shapes of the proposed CNN model.

Layer Name	Previous Layer	Function	Weight Filter Size /Kernels	Padding	Stride	Output Tensor Size	Learnable Parameters
in	-	-		-	-	$100 \times 100 \times 1$	-
conv1	in	conv2d	$5 \times 5 \times 1/16$	same	1	$100 \times 100 \times 16$	416
bn1	conv1	-		-	-	$100 \times 100 \times 16$	32
relu1	bn1	ReLU		-	-	$100 \times 100 \times 16$	-
conv2	relu1	conv2d	$3 \times 3 \times 16/32$	same	2	$50 \times 50 \times 32$	4608
bn2	conv2	-		-	-	$50 \times 50 \times 32$	64
relu2	bn2	ReLU		-	-	$50 \times 50 \times 32$	-
conv3	relu2	conv2d	$3 \times 3 \times 32/32$	same	1	$50 \times 50 \times 32$	9216
bn3	conv3	-		-	-	$50 \times 50 \times 32$	64
relu3	bn3	ReLU		-	-	$50 \times 50 \times 32$	-
fc1	relu3	-		-	-	$1 \times 1 \times 2$	160 k
softmax	fc1	softmax		-	-	$1 \times 1 \times 2$	-
fc2	relu3	-		-	-	$1 \times 1 \times 2$	160 k

The proposed neural network contains 334.4 k parameters and has a model size of 1.22 MB after training. This network was used because it showed good results in similar tasks. It was considered valuable to use other types of convolutional neural networks, such as VGG-16 and ZFNet, but these networks had too many parameters for this application, and, thus, training would take too long to evaluate the performance of the proposed custom training loop with a custom loss function.

We analyzed the basic structure of the proposed convolutional neural network; in the next subsection, the learning process of the network is discussed.

2.2. Learning Process

The learning process of a deep learning network consists of three steps: data acquisition, data preparation and model training. In the current work, the first step consists of capturing images of the surroundings of the pin. The images are taken using a camera that captures images of 612×512 pixels. The images are in gray-scale and are saved as a tiff file.

The second step consists of preparing the data to train the network. The first task is to label the images. After manually identifying the pin in each image, the data are used to generate images starting from the seed images. The identification is made by drawing a rectangle surrounding the pin. The center pixel of the marked rectangle is taken as the location of the pin, which is then used in the training process as the target value. Then, from each seed image, 5 images are obtained. In these images, the position of the pin is the same, but the contrast and the brightness of the images are randomly modified using Equations (3)–(5):

$$\text{Contrast factor}: \ Cf = 1 - 0.2 \cdot rand, \tag{3}$$

$$\text{Brightness factor}: Bf = 0.3 \cdot (rand - 0.5), \tag{4}$$

$$I_{ij} = Cf \cdot Is_{ij} + Bf \tag{5}$$

where $rand$ is a random value between 0 and 1, I_{sij} is the seed pixel value, and I_{ij} is the resulting pixel value. This is applied to all seed images to obtain 1620 images.

These images, however, still have a size of 612×512. In order to train the network, the images need to be transformed so that their size is 100×100. Therefore, each image receives a random transformation, where a 100×100 size region is chosen from each image. This is carried out by randomly selecting whether the image has a pin, the chance of which is 50/50. At the end of the transformation, there are 810 images with a pin and 810 without a pin.

The final step of the training consists of the model training itself. In this case, a custom training loop is used. MATLAB is the software chosen to develop the different algorithms

that are involved in this work. This software has different tools to develop and train deep neural networks. One of these functionalities is to train custom training loops, updating the learnable parameters of the network using different solvers. In this case, the Adam (adaptative moment estimation) solver is used [34].

In this process, each mini-batch of data is evaluated using the modeloGradients function. The modeloGradients function takes the following as inputs: the network and a mini-batch of input data, with the corresponding targets T1 and T2 containing the labels and positions, respectively. Then, it returns the gradients of the loss with respect to the learnable parameters, the updated network state and the corresponding loss.

The loss for each mini-batch θ is calculated by adding the cross-entropy loss of the classification task and the half mean squared error, with the latter multiplied by factor $\lambda = 0.1$, following Equation (6):

$$loss_{\theta} = loss_{Ex,\theta} + \lambda \cdot loss_{mse,\theta} \tag{6}$$

The cross-entropy loss (Ex) for each mini-batch θ is calculated using Equation (7):

$$loss_{Ex,\theta} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} t_{ni} \ln y_{ni} \tag{7}$$

where N is the number of samples, K is the number of classes, t_{ni} is the indicator showing that the n^{th} sample belongs to the i^{th} class, and y_{ni} is the output for sample n for class i. That is, y_{ni} is the probability that the network associates the n^{th} input with class i.

The half mean squared error (mse) operation computes the half mean squared error loss between the network predictions and target values for regression tasks. The loss for each mini-batch θ is calculated using the following Equation (8):

$$loss_{mse,\theta} = \frac{1}{2N}\sum_{i=1}^{M}(X_i - T_i)^2 \tag{8}$$

where X_i is the network prediction, T_i is the target value, M is the total number of responses in X (across all observations), and N is the total number of observations in X.

Afterwards, the calculated gradients are used to update the learnable parameters of the network. This process continues until the training ends, which is when the training reaches 200 epochs. Each mini-batch consists of 60 elements. Therefore, 5400 iterations are performed. The parameters of the Adam solver are listed in Table 2.

Table 2. Parameter values of the Adam solver.

Parameter	Value
Learn rate	0.001
Gradient decay factor	0.9
Squared gradient decay factor	0.999
Epsilon *	10^{-8}

* Small constant for preventing divide-by-zero errors.

During the training, a validation evaluation is performed. This is carried out to ensure that the training is performing well and that the results are converging. To perform this task, a new dataset is created following the same steps as those used for the training data. In this case, 3 images are obtained from each seed image in order to speed up the validation process. This dataset is evaluated as the training dataset in groups of 60 data samples. At the end of each training epoch, all the validation data are evaluated, and the average loss value is returned by the algorithm.

The first loss function is based on a constant ratio between the two different losses. Regarding the second loss function, we only want to perform the localization task when the network detects an object in order to evaluate whether this approach improves the

effectiveness of the network. This new loss function is also based on the cross-entropy loss of the classification task and the half mean square error of the regression task. However, the combination of both is not a simple constant ratio, as with the first loss function. At first, we thought that the loss function only needed to take into account the cross-entropy loss when the classification was not performed correctly, because trying to locate a pin in an image that does not have one would not be correct. Therefore, the loss function that was proposed included the target values of the classification task, as well as the network prediction. However, the use of the predictions to calculate the loss led the network to classify all images in one group due to the learnable parameters being related to the predictions. Because of this, it was decided that the network predictions should not be used. Consequently, only the target values of the classification task are used. In the images where there is no pin, only the cross-entropy loss is used to calculate the overall loss. In the other case, the half mean squared error is also computed. This is carried out with the objective of only taking into account the localization task when there is a pin to locate. All this is performed in each image μ of the mini-batch θ using Equation (9):

$$loss_{\theta} = loss_{Ex,\theta} + \frac{1}{N}\sum t_{pc1}^{\mu} \cdot loss_{mse,\mu} \tag{9}$$

where t_{pc1}^{μ} is the target probability that image μ contains a pin, $loss_{Ex,\theta}$ is the cross-entropy loss of the mini-batch θ (Equation (7)), $loss_{mse,\mu}$ is the half mean square error loss of the image μ (Equation (8)), and N is the number of images in the mini-batch θ.

As with the first proposed loss in this article, this loss is used to calculate the gradients of the loss with respect to the learnable parameters in order to update the latter to improve the predictions of the network. The same base network is used to compare the obtained results.

After finalizing the training, the same validation data are used to evaluate the training. At this point, 10 randomly selected images are chosen to evaluate the network performance. The same images are used to evaluate the training of the second loss function. Therefore, both results are directly comparable and allow one to conclude whether the proposed method is effective and which loss function has the best performance.

3. Results

In this section, the results of the investigation are presented. First, the network is trained using the presented loss function. The loss during the training and the average validation loss are presented in Figure 2. The quick drop that appears in the first iterations suggests that the classification of the images is optimized early in the training. The values obtained at the end of the training are collected in Table 3.

Table 3. Training properties values.

Property	Value
Epoch	200
Iteration	5400
Training time	54 min 37 sec
Loss	2.11
Classification loss	0
Regression loss	21.087
Validation loss	1.87
Validation classification loss	2×10^{-4}
Validation regression loss	18.699

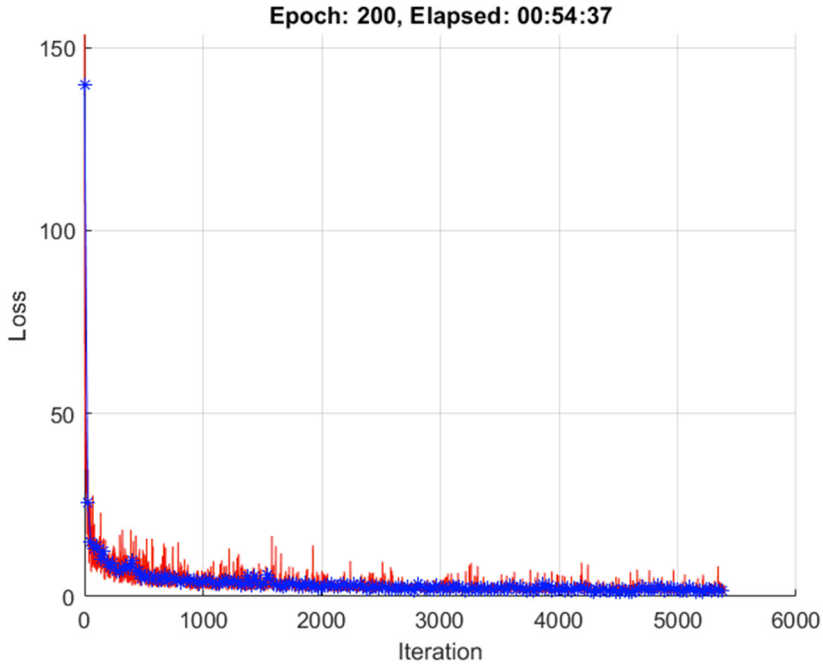

Figure 2. The loss during the training process (red) and the average validation error obtained for each epoch of the training data (blue) of the first proposed loss function. In this figure, the downward trend of the loss value can be seen. The loss value continues to drop at a low but constant rate after the quick drop of the early iterations.

Looking at the results in Table 4, it can be seen that the network achieves very good results in the classification task, labeling most of the images correctly. After analyzing all the images in the validation dataset, 10 images were randomly selected to expose the training results. Regarding the pin localization, the results can be improved. Most of the time, the network is able to locate the pin with decent precision. However, the localization task fails when there is no pin in the image, for example, as shown in images 816, 165 and 836 in Figure 3. It can also be noted that image 357 is not classified correctly, although the localization task is performed accurately.

Table 4. The values of the analyzed validation images.

Image Index	T1		Y1		T2		Y2		Loss	Classification	Regression
	No Pin	Pin	No Pin	Pin	x	y	x	y			
774	0	1	0	1	47	25	44.19	26.49	0.505	0	5.053
34	1	0	1	0	0	0	−0.40	−2.47	0.314	0	3.142
816	1	0	1	0	0	0	29.24	33.41	98.58	0	985.8
869	0	1	0	1	43	55	37.63	47.26	4.438	0	44.384
11	0	1	0	1	60	66	55.52	64.16	1.173	0	11.728
165	1	0	1	0	0	0	10.94	15.25	17.624	0	176.24
836	1	0	1	0	0	0	10.53	7.06	8.036	0	80.364
357	0	1	1	0	31	25	21.04	16.74	44.416	36.044	83.724
697	0	1	0	1	28	31	33.59	40.84	6.401	0	64.014
827	0	1	0	1	28	29	21.69	22.37	4.193	1.59×10^{-4}	41.93

T1 and T2 are the target values of each image. Y1 and Y2 are the predictions made by the network. Loss is the total loss; Classification is the cross-entropy error; Regression is the half mean square error. All the values smaller than 10^{-4} are considered null.

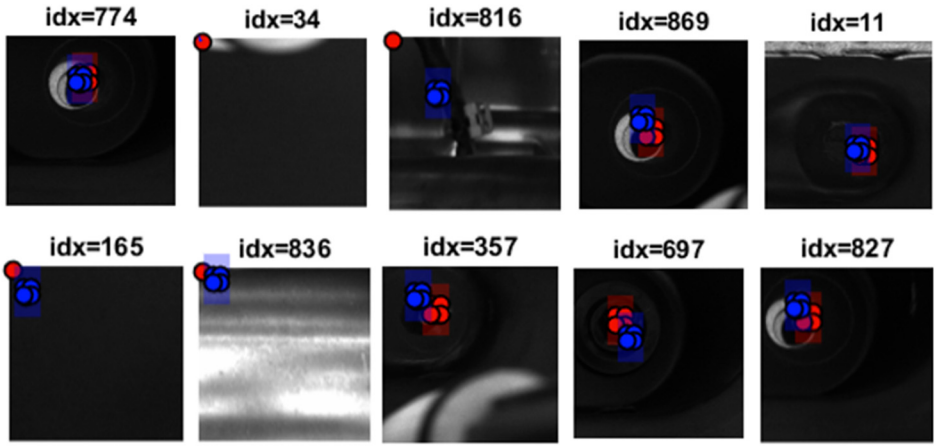

Figure 3. Images used to analyze the performance of the network. The red marking shows the real position of the pin (manually labeled), whereas the blue marking shows the prediction of the network.

The same observation can be made with the second proposed loss. In Figure 4, the loss during the training and the average validation loss are presented.

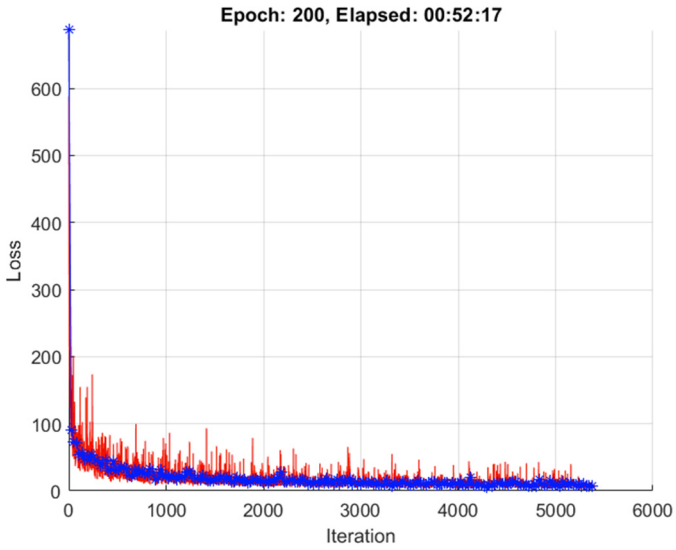

Figure 4. The loss during the training process (red) and the average validation error obtained for each epoch of the training data (blue) of the second proposed loss function. In this figure, the downward trend of the loss value can be seen. Comparing the values with those in Figure 2, the loss value is higher at the beginning, although at the end of the training process, the values converge, as can be seen in Table 5.

Table 5. Training properties values.

Property	Value
Epoch	200
Iteration	5400
Training time	52 min 17 sec
Loss	4.497
Classification loss	0
Regression loss	8.431
Validation loss	6.461
Validation classification loss	0
Validation regression loss	12.922

After analyzing all the images in the validation dataset, 10 images were randomly selected to present the training results. Looking at the results in Table 6, as well as those of the first network, this network performs well on the classification task, thus improving its performance in the location task. In images 816 and 836 in Figure 5, it can be seen that the network predicts the positions of the pins, although they are not in the images. However, in the other images, the network predicts the positions of the pins more accurately than the previous network. This can be the result of giving the regression task more influence when there is a pin. This can also provide an explanation for the overall *loss* values being higher than those in the first network.

Table 6. The values of the analyzed validation images.

Image Index	T1		Y1		T2		Y2		Loss	Classification	Regression
	No Pin	Pin	No Pin	Pin	x	y	x	y			
774	0	1	0	1	47	25	41.62	23.61	15.414	0	15.414
34	1	0	1	0	0	0	3.35	1.84	0	0	7.295
816	1	0	1	0	0	0	17.07	21.18	0	0	370.07
869	0	1	0	1	43	55	40.28	45.80	46.019	0	46.019
11	0	1	0	1	60	66	59.09	65.53	0.521	0	0.521
165	1	0	1	0	0	0	−2.39	3.31	0	0	8.315
836	1	0	1	0	0	0	10.77	4.42	0	0	67.727
357	0	1	0.991	0.009	31	25	17.13	13.61	196.74	35.736	161
697	0	1	0	1	28	31	25.48	28.55	6.170	0	6.170
827	0	1	0	1	28	29	25.91	29.84	2.533	0	2.533

T1 and T2 are the target values of each image. Y1 and Y2 are the predictions made by the network. Loss is the total loss; Classification is the cross-entropy error; Regression is the half mean square error. All the values smaller than 10^{-4} are considered null.

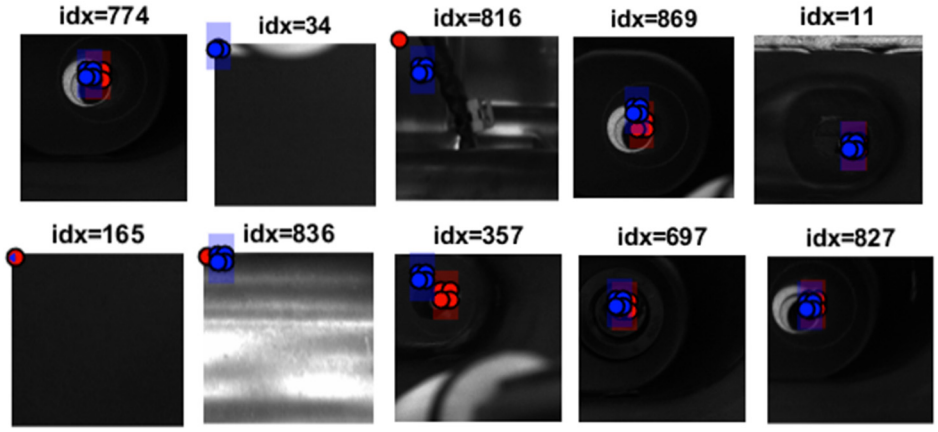

Figure 5. Images used to analyze the performance of the network. The red marking shows the real position of the pin (manually labeled), whereas the blue marking shows the prediction of the network.

4. Discussion

The improvements of convolutional neural networks in image classification [18] and object detection [19,30] have positioned CNNs as the best solution to perform these tasks. Most of the reviewed literature suggests that a single task is performed by each network. Although single-stage object detectors, such as YOLO [25] and SSD [26], perform both the region proposal task and the detection task as a single task, they do not classify the input image. In this work, the objective was to first classify the input image into two groups, namely, images that contain a pin and images that do not contain a pin, and to then position the pin inside the image, all while using the same simple network.

In this work, we proposed two different approaches to perform a custom training loop. The two approaches differ from each other in the loss function that is used in each case. After analyzing the results, we could see that the classification task was performed

accurately in both trained networks. Regarding the localization of the pin, the second network achieved better results. The differences in the approaches of the two methods led us to think that the second network performed better in the localization task because it gives more weight to the half mean squared error when it needs to perform the localization task. However, both networks underperformed in the localization task when there was no pin in the image. Therefore, in following iterations of this network, we will attempt to improve the localization task of the network by refining the training processes of the network and the loss function.

5. Conclusions

The objective of this research is to determine whether image classification and object localization tasks can be performed using a single CNN. In this work, two new loss functions are added to a custom training loop. Both loss functions combine the cross-entropy loss (Ex) for the classification task and the half mean square error (mse) for the regression task (object localization). The main differences in both networks are as follows:

1. The first network always computes the loss by adding the classification task loss and the localization task loss (Equation (6)), whereas the second only takes into account the localization task loss when there is a pin in the image (Equation (8)).
2. In the first network, the localization task loss is multiplied by factor λ (Equation (6)), reducing the importance that this loss has in the overall loss of the network.

The first loss function is very simple, allowing for the results to be more easily analyzed and for the network to be finetuned more accurately. However, the second loss function introduces differentiation between the two types of losses that computes the total loss depending on the image localization task. By doing this, only the half mean square error is computed when an object is detected in the image. This is the key aspect of this second loss, because the first loss does not make any differentiation in the computing of the total loss, which, in our humble opinion, is the biggest contribution of this work. Based on the results, this approach shows better performance than the first loss function, paving the way forward for future research.

These results show that computer vision can benefit the manufacturing industry. The tasks that require some type of visual recognition, detection or classification can now be performed efficiently using neural networks. This can lead to more automated manufacturing processes. Therefore, workers can focus less on automatic tasks and more on their efforts in other tasks, such as problem solving and organizational tasks.

Author Contributions: Conceptualization, E.Z., A.S.-C. and J.M.G.-H.; methodology, B.U.-T. and E.Z.; software, E.Z., A.S.-C. and B.U.-T.; validation, B.U.-T., E.Z. and U.F.-G.; formal analysis, B.U.-T., E.Z.; investigation, B.U.-T. and E.Z.; resources, B.U.-T. and E.Z.; writing—original draft preparation, B.U.-T.; writing—review and editing, B.U-T. All authors have read and agreed to the published version of the manuscript.

Funding: The current study was sponsored by the Government of the Basque Country-ELKARTEK21/10 KK-2021/00014 ("Estudio de nuevas técnicas de inteligencia artificial basadas en Deep Learning dirigidas a la optimización de procesos industriales") research program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors want to thank the technical support provided by Javier Loredo.

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

References

1. International Federation of Robotics. Available online: https://ifr.org/img/worldrobotics/Executive_Summary_WR_Industrial_Robots_2021.pdf (accessed on 10 December 2022).
2. Probst, L.; Frideres, L.; Pedersen, B.; Caputi, C.; Luxembourg, P. *Service Innovation for Smart Industry Human-Robot Collaboration Business Innovation Observatory Contract No 190/PP/ENT/CIP/12/C/N03C01*; European Commission: Brussels, Belgium, 2015.
3. Forschungsunion. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0 April 2013 Securing the Future of German Manufacturing Industry Final Report of the Industrie 4.0 Working Group*; Forschungsunion: Berlin, Germany, 2012.
4. Villani, V.; Pini, F.; Leali, F.; Secchi, C. Survey on Human–Robot Collaboration in Industrial Settings: Safety, Intuitive Interfaces and Applications. *Mechatronics* **2018**, *55*, 248–266. [CrossRef]
5. Inziarte-Hidalgo, I.; Uriarte, I.; Fernandez-Gamiz, U.; Sorrosal, G.; Zulueta, E. Robotic-Arm-Based Force Control in Neurosurgical Practice. *Mathematics* **2023**, *11*, 828. [CrossRef]
6. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2323. [CrossRef]
7. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Kai, L.; Li, F.-F. *ImageNet: A Large-Scale Hierarchical Image Database*; Institute of Electrical and Electronics Engineers (IEEE): New York, NY, USA, 2010; pp. 248–255.
8. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
9. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
10. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In Proceedings of the Computer Vision–ECCV 2014 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Springer International Publishing: New York, NY, USA, 2014; pp. 818–833.
11. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2014**, arXiv:1409.1556.
12. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
13. Zagoruyko, S.; Komodakis, N. Wide Residual Networks. *arXiv* **2016**, arXiv:1605.07146.
14. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; IEEE Computer Society: Washington, DC, USA, 2015; pp. 1–9.
15. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
16. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
17. Yang, L.; Jiang, H.; Cai, R.; Wang, Y.; Song, S.; Huang, G.; Tian, Q. CondenseNet V2: Sparse Feature Reactivation for Deep Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021.
18. Sultana, F.; Sufian, A.; Dutta, P. Advancements in Image Classification Using Convolutional Neural Network. In Proceedings of the 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), Kolkata, India, 22–23 November 2018. [CrossRef]
19. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object Detection with Deep Learning: A Review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef] [PubMed]
20. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; IEEE Computer Society: Washington, DC, USA, 2014; pp. 580–587.
21. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; IEEE: Washington, DC, USA, 2015; pp. 1440–1448.
22. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
23. Fu, L.; Feng, Y.; Majeed, Y.; Zhang, X.; Zhang, J.; Karkee, M.; Zhang, Q. *Kiwifruit Detection in Field Images Using Faster R-CNN with ZFNet*; Elsevier B.V.: Amsterdam, The Netherlands, 2018; Volume 51, pp. 45–50.
24. Song, Z.; Fu, L.; Wu, J.; Liu, Z.; Li, R.; Cui, Y. Kiwifruit Detection in Field Images Using Faster R-CNN with VGG16. *IFAC Pap.* **2019**, *52*, 76–81. [CrossRef]
25. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
26. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the Computer Vision–ECCV 2016 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016.
27. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.

28. Li, Z.; Peng, C.; Yu, G.; Zhang, X.; Deng, Y.; Sun, J. Light-Head R-CNN: In Defense of Two-Stage Object Detector. *arXiv* **2017**, arXiv:1711.07264.
29. Zhang, S.; Wen, L.; Bian, X.; Lei, Z.; Li, S.Z. Single-Shot Refinement Neural Network for Object Detection. In Proceedings of the Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; IEEE Computer Society: Washington, DC, USA, 2018; pp. 4203–4212.
30. Wu, X.; Sahoo, D.; Hoi, S.C.H. Recent Advances in Deep Learning for Object Detection. *Neurocomputing* **2020**, *396*, 39–64. [CrossRef]
31. Tan, M.; Pang, R.; Le, Q.V. EfficientDet: Scalable and Efficient Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020.
32. Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; Xie, S. A ConvNet for the 2020s. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
33. Teso-Fz-Betoño, D.; Zulueta, E.; Sánchez-Chica, A.; Fernandez-Gamiz, U.; Saenz-Aguirre, A. Semantic Segmentation to Develop an Indoor Navigation System for an Autonomous Mobile Robot. *Mathematics* **2020**, *8*, 855. [CrossRef]
34. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.