IMPLEMENTATION OF IMAGE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK (CNN) ALGORITHM ON VEHICLES IMAGES

Muhammad Nurhadi[1*], Joko Purnomo[2]
Electrical Engineering Study Program, Postgraduate Masters in Technology and Engineering, Gunadarma University
*Correspondence: hadymn.official@gmail.com

Abstract
The use of surveillance cameras for most agencies only relies on video recordings and storing them for a certain time. The use of this surveillance camera can be applied to determine the type of vehicle even if the camera is not in the right position. Regarding the background of the problem, this research will use the Convolutional Neural Network (CNN) algorithm, which is part of Deep Learning with the help of Keras Library and TensorFlow, to carry out the learning process on videos captured by surveillance cameras so that it can detect images from 3 types of vehicles. The dataset used is 100 images of motorcycles, 100 images of cars, and 100 images of buses. The method used is the Image Classification Method, and the model used is the best model selected from several experiments. Researchers used training and test data distribution, namely 80% and 20%. The best results were obtained with an accuracy rate of 96.49% using epoch 100, learning rate 0.001, and batch size 32. Meanwhile, vehicle images produced image accuracy for motorcycle images when using test data from outside the dataset is 78.92%, car image is 81.71%, and bus image is 82.26%.

History:

Received: January 20, 2022
Accepted: February 12, 2022
First published online:
July 22, 2022

Keywords:

Convolutional Neural Network
Image Classification
Python
Images

1. Introduction

Cameras to monitor certain environments are very common in various agencies. This utilization can be further developed to facilitate the work of the security profession in an agency. Opening the gate manually by security officers will be easier if it is done automatically. The use of images as a source of object recognition can be found in modern shopping centers as automatic doors when people pass through certain doors can also be found in the health sector for lung disease sensors (Ker, 2018). The use of images on the camera to identify vehicles that want to enter the agency area can be used as a reference to open the main door automatically. Using vehicle images as a control system is a difficult challenge. This is due to the presence of the vehicle to be taken, various backgrounds, changing lighting conditions, different vehicle shapes, and the need for real-time execution.

Along with the times, digital image classification is needed in various fields, such as informatics, medicine, marine, agriculture, and business. Several studies have been conducted; for example, deep learning detection and classification of medical images are the main components for computer-aided diagnosis (CAD) systems to support physicians efficiently (Al-antari, 2020). The purpose of image classification is to classify input images into certain categories. Image classification is currently a problem that has been looking for a solution in computer vision for a long time. Another study (Dunnmon, 2018) focused on improving the performance of CNN classifiers by increasing the size of the dataset how to duplicate the human ability to understand digital image information so that the computer can recognize objects in the image like humans (Maurya, 2021).

Images must be pre-processed, segmented, and extracted to get optimal performance. Another Multilayer Perceptron (MLP) development that can overcome this problem is the Convolutional Neural Network (CNN). Convolutional Neural Network (CNN) is a deep learning (DL) method that can be used to detect and recognize an object in a digital image. Deep Learning is one of the sub-fields of Machine Learning. In recent years, due to the superior performance of deep learning models, many have applied deep learning methods for disease prediction (Luo, 2021).

Deep Learning implements the basic concept of Machine Learning, which implements the CNN algorithm with more layers. With the number of hidden layers that are used between the input layer and the output layer, this network can be said to be a deep neural net. In the last few years, Deep Learning (DL) is now the most advanced in most image classification tasks, representing the best research fields in this field, achieving results comparable to or even better than humans (Carneiro, 2021). This is largely due to stronger computing factors, large data sets, and techniques to train deeper networks. Based on the above background, this study applies the deep learning method using CNN to help identify the type of vehicle and determine the level of aquation in each captured image.

2. Research Method

2.1. Dataset Design

The use of the dataset in the CNN method is in the form of image data. The CNN model will perform well when using a lot of image train data. So that a model can learn to recognize the image. The dataset used in this study is an image collected through the Google search engine. The image data used this time is a picture of 3 types of vehicles to be classified, such as motorbikes, cars and buses. Collecting the dataset if done manually will take a long time. So that researchers use the image crawling method using JavaScript and python programs. The JavaScript program aims to retrieve the image URL contained in Google and the python program that performs the execution to download the image.

2.2. Network Architecture

In the Convolutional Neural Network (CNN) algorithm, the formation of the network architecture can affect the results of model accuracy.

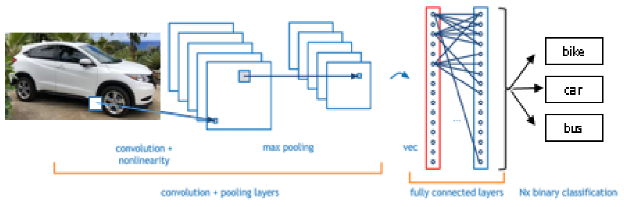

Figure 1. Network architecture

Figure 1. is a network architecture in the training process to produce an optimal model. This study uses an image input with a size of 96x96x3; the aim is to compare the accuracy value based on the image size.

2.3. Fully Connected Layer

Next is the fully connected layer. This process aims to transform the dimensions of the data so that the data can be classified linearly.

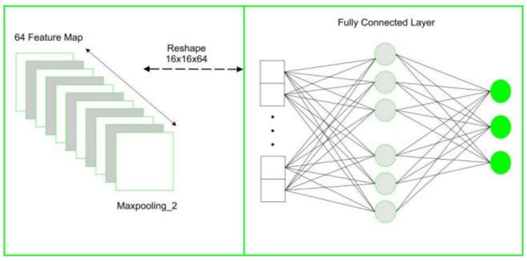

Figure 2. Fully connected layer

Figure 2 is the converting process resulting from the map max-pooling feature into flattening or vector. In this process, the input matrix values from the previous layer will be converted into vectors. This process is the same as the MLP (Multi-Layer Perceptron) Process. These networks generally use fully connected layers where each pixel is considered a separate neuron. In this process, the "dropout" method is usually applied. This method aims to disable multiple edges connected to each neuron to avoid overfitting. After that, the last step is process classification.

2.4. Testing Design

This test is carried out to evaluate the model produced by CNN. This test is carried out in two stages: training and testing. The training stage is the stage where the CNN model is tested with the training data that has been provided. The number of training data provided is 300 image data, with 100 pictures per class. The training data is divided back into training and validation, 240 training, and 60 validations. The testing stage is the stage of testing the model carried out in the training stage. The number of training data in this study was 80 pictures per class from 100 pictures overall. At this stage, the model is tested with different images with the aim of testing whether the model has produced a good performance in classifying an image.

Table 1. Dividing dataset

Data	Percentage	Total Data
Training	80%	240
Testing	20%	60

2.5. Model Training

Based on these two directories is the storage of the vehicle image dataset. Each directory has created a directory for each vehicle class. A directory was created for the storage of motorbikes, cars, and buses. After creating the directory, the next step is to create a CNN architecture.

Sample of algorithm

```
# save the model to disk
    print("[INFO] serializing network...")
          model.save(args["model"])
# save the label binarizer to disk
    print("[INFO] serializing label binarizer...")
    f = open(args["labelbin"], "wb")
    f.write(pickle.dumps(lb))
    f.close()
```

2.6. Program Testing

The Convolutional Neural Network (CNN) algorithm requires a training and testing process. This training process aims to train the CNN algorithm in recognizing the dataset and forming a model based on this training. The testing process aims to test a model that was formed during the training process. The following is the testing process in this study.

Sample of algorithm

```
# build the label and draw the label on the image
#label = "{}: {:.2f}% ({})".format(label, proba[idx] * 100)
    label = "{}: {:.2f}%".format(label, proba[idx] * 100)
    output = imutils.resize(output, width=400)
    cv2.putText(output, label, (10, 25),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.7, (0, 255, 0), 2)

# show the output image
    print("[INFO] {}".format(label))
    cv2.imshow("Output", output)
    cv2.waitKey(0)
```

3. Result and Discussion

In this study, researchers classified three classes of vehicle images, motorbikes, cars, and buses, using the Convolutional Neural Network (CNN) algorithm. The primary process in making this model begins with the training data process. This process aims to form a model that will be used for testing data testing. The parameter to measure the model's success rate is the accuracy value. The model accuracy value can be determined by testing using testing data. The training process uses the Keras packages in Python with the TensorFlow backend. Keras is a module created by Google to facilitate research on neural networks and can run on TensorFlow.

3.1. Image Pre-processing

In the script below, a simple loop will be performed by entering and resizing each image to 96x96 pixels (dimensions required by VGGNet) and adding the image array to the data list followed by extracting the label class from image Path.

Sample of algorithm

```
    # loop over the input images
    for imagePath in imagePaths:
      # load the image, pre-process it, and store it in the data list
      image = cv2.imread(imagePath)
      image = cv2.resize(image, (IMAGE_DIMS[1], IMAGE_DIMS[0]))
      image = img_to_array(image)
      data.append(image)

      # extract the class label from the image path and update the
      # labels list
      label = imagePath.split(os.path.sep)[-2]
  labels.append(label)
```

Table 2. Scenario of proportion of training data

Total Data	Proportion	Validation Accuracy
300	70% : 30%	79.49%
	75% : 25%	87.50%
	80% : 20%	96.49%

The best results were obtained using the 80%:20% scenario. This is because a system with more training data carries out the learning process, so the system will learn more about images than other scenarios. In the last line, we convert the labels to vectors using one-hot-encoding.

3.2. Create train.py

Next, the researcher will begin to train the classification of images on the dataset using deep learning Keras and Python. If the researcher had previously created a model in vggnet.py, then the researcher would then make CNN training in image classification with Keras in train.py.

Table 3. Network configuration experiment

Model Name	Epoch	Learning Rate	Batch Size	Val. Loss	Val. Acc.
Model_1	50	Le-3	32	0.3621	0.85
Model_2	100	Le-3	32	0.1355	0.96
Model_3	50	Le-4	40	0.4571	0.80
Model_4	100	Le-4	40	0.3291	0.89
Model_5	50	Le-5	50	0.6516	0.68
Model_6	100	Le-5	50	0.6114	0.70

It can be seen from the experimental Table 3 that the best model is model_2. At epoch 100, learning rate Le-3, and batch size 32. It is known that the epoch used is 100, meaning the program will learn 100 repetitions of training data. The learning rate is 0.001, where the higher the value, the greater the learning steps. However, if it is too big, it will also make the algorithm unstable. Learning Rate influences the speed of the neural network in a minimum solution. While the batch size is 32, the algorithm will take the first 32 samples (1 to 32) from the dataset and then train the network.

Furthermore, 32-second samples (32 to 64) will be taken from the dataset and trained in the network again. This will continue until the algorithm spreads (propagate) through all network samples. This is where the advantage of propagation is that the network becomes faster with smaller batches because the algorithm will update the weights after each propagation. Of the three network configurations (epoch, learning rate, and batch size) that the researchers used, the results were obtained with a loss of 0.2030. The loss function is here used to measure the performance of the neural network in predicting the charge. The loss function used by researchers here is Cross Entropy, which is commonly used for two classifications, with a value of 0.2030, which means that the neural network's performance is good in predicting the target. Then the resulting accuracy of the training data, as much as 80% of the data, is 0.9649 and loss 0.1355. Then the resulting accuracy from the testing data of as much as 20% of the data is 0.9102 and a loss of 0.2030. The following is summarized in Table 4 :

Table 4. Summary of accuracy result

Data	Percentage	Total Data	Loss Value	Acc. Value
Training	80%	80	0.1355	0.9649
Testing	20%	20	0.2030	0.9102

Then the researcher initializes the data and adds a label to the data. This data is responsible for storing the images loaded from memory along with the labels of each class.

3.3 Results of the Train Data

As previously explained, researchers used LeNet for several reasons; namely, VGGNet is a lightweight and easy-to-understand CNN collection for beginners and can easily train leNet on the researcher dataset using CPU. Researchers built this VGGNet model with the Adam optimizer.

Based on the results, Table 4 is a sample of the model execution process and produces accuracy, loss, val_accuracy, and val_loss values. In this fit model process, in addition to producing numeric output, it can also be made in the form of a plot. The plot makes it easier for researchers to see model-fit numbers as a graph from the CNN program execution results, as shown Figure 3.

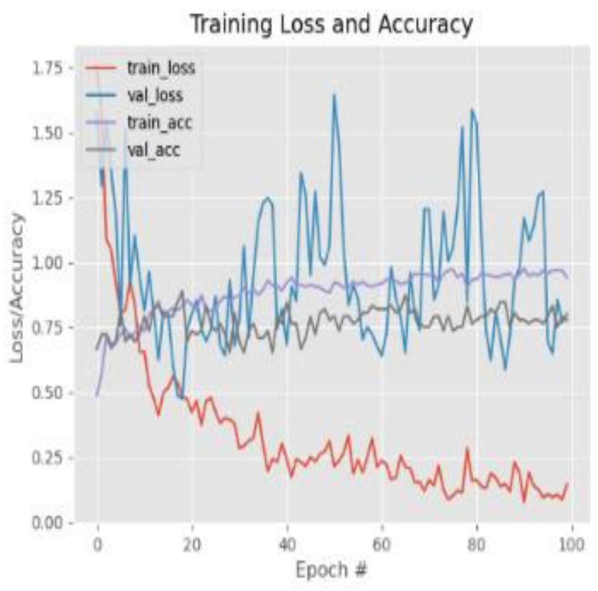

Figure 3. Graph training loss and accuracy

3.4. Program Testing

Program testing is carried out to measure the level of accuracy obtained on each image to be identified. The author prepares test data from 60 pictures of CCTV cameras, with 20 pictures for each class. For more details, the test results are summarized in Table 5.

Table 5. Summary of accuracy result

Testing	Class		
	Bike (%)	Car (%)	Bus (%)
1	63.32	82.44	90.02
2	87.35	68.55	89.55
3	76.20	87.40	86.80
4	89.45	86.05	77.36
5	78.34	86.80	87.90
6	88.26	79.00	79.94
7	66.20	78.35	92.44
8	86.22	85.34	83.88
9	76.10	87.98	64.96
10	78.50	77.02	83.78
11	84.38	83.56	60.72
12	92.04	85.78	81.74
13	90.70	70.02	81.36
14	87.80	89.00	83.70
15	83.84	89.04	78.90
16	90.00	85.12	86.94
17	66.75	84.22	77.90
18	81.95	65.80	80.08
19	90.10	79.80	86.90
20	81.05	83.04	90.42
$\bar{x}$	78.92%	81.71%	82.26%

Several factors cause inaccuracy in this study. First, there is a large difference between reference data and online image data due to differences in images when taking images of vehicles that are always moving, so they cannot maintain the same distance between the vehicle and the camera. Second, the different lighting between the sample data collection process and the test data will affect the brightness level of the color produced in the captured image. This different level of lighting is caused by the intensity of the light captured on the CCTV camera. Third, non-unique vehicle patterns will affect the vehicle pattern recognition test because the vehicle pattern will be recognized as a vehicle that is almost similar to it.

4. Conclusion

1. The best results are obtained through several experiments with different numbers of epochs, learning rates, and batch sizes, namely the model_2 model with 100 epochs, 0.001 learning rate, and 32 batch sizes.
2. Through three scenarios (60:40, 70:30, 80:20), the best proportion of data is obtained, namely 80% training data and 20% testing data, with an accuracy of 96.49%.
3. Based on the program test results using new data, as much as 20 images for each class of vehicle, the resulting average accuracy for the motorbike class is 78.92%, the car class is 81.71% and the bus class is 82.26%.

The suggestions given in this study are that future research is expected to increase the number of classification classes of all vehicles to classify more objects in an image. The image data is reproduced to train the model to achieve a higher accuracy level and developed again in an Android-based application and/or website.

References

Al-antari Mugahed A., and Kim, T.-S, 2020, “Evaluation of Deep Learning Detection and Classification towards Computer-aided Diagnosis of Breast Lesions in Digital X-ray Mammograms”, *Computer Methods and Programs in Biomedicine,* Vol. 196, No. 10, accessed February 11, 2022, *105584.* doi:10.1016/j.cmpb.2020.105584.

Carneiro, Gabriel, A., Rafaela Magalhaes, and Alexandre Neto, 2021,”Grapevine Segmentation RB Images using Deep Learning”, *Information Systems and Technologies*, Vol. 196, pp. 101-106, accessed February 11, 2022 10.1016/j.procs.2021.11.078

Dunnmon, Jared, and Yi, Darvin., 2018, “Assessment of Convolutional Neural Networks for Automated Classification of Chest Radiographs”, *Radiology,* Vol. 290, No. 2, pp. 537-544, accessed February 11, 2022, doi:10.1148/radiol.2018181422.

Ker, Justin., Lipo, Wang., J. Rao., and Tchoyoson, 2018, “Deep Learning Applications in Medical Image Analysis”, *IEEE Access.*, vol.6, pp. 9375-9389, accessed February 11, 2022, 10.1109/ACCESS.2017.2788044.

Luo, X., Gandhi, P., Zhang, Z., Shao, W., Han, Z., Chandrasekaran, V. and Huang, K, 2021, “Applying interpretable deep learning models to identify chronic cough patients using EHR data”*, Computer Methods and Programs in Biomedicine,* Vol. 210*,* accessed February 11, 2022, *106395.* doi:10.1016/j.cmpb.2021.106395

Maurya, Ritesh., Vinay, Kumar., and Malay, Kishore., 2021, “Deep Learning-based Microscopic cell images classification framework using multi-level ensemble”, *Computer Methods and Programs in Biomedicine,* Vol.211, No.10, accessed February 11, 2022, doi.org/10.1016/j.cmpb.2021.106445.