VOL. 16, NO. 10, MAY 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DESIGN AND IMPLEMENTATION OF TRAFFIC VIOLATION DETECTION SYSTEMS WITH DEEP LEARNING TO SUPPORT ELECTRONIC TRAFFIC LAW ENFORCEMENT (e-TLE)

Kusworo Adi[1], Catur Edi Widodo[1] and Aris PujiWidodo[2]
[1]Department of Physics, Faculty of Sciences and Mathematics, Diponegoro University, Indonesia
[2]Department of Informatics, Faculty of Sciences and Mathematics, Diponegoro University, Indonesia
E-Mail: kusworoadi@lecturer.undip.ac.id

ABSTRACT

In this paper we investigate a method based on machine learning to identify types of motor vehicle traffic violations with image processing techniques to support electronic traffic law enforcement. Machine learning is a machine learning method that performs well when applied to data outside of the training set. In this study, data collection was carried out using a camera with a resolution of 13 megapixels. This data processing carries out a training process so that it requires high quality hardware such as a laptop / computer to get a result that can predict objects well. In this system it uses the CPU to train a model where the data is relatively small. The output in this system is a traffic violation image taken by using a camera. This system will work if a vehicle violates traffic, such as a violation of not wearing a helmet for motorcyclists and passing road markings. The system formed is a training result model with a total of 100 000 steps with 32 and 16 batch sizes, namely in the form of an inference graph consisting of a checkpoint file, frozen_inference_graph.pb, and there are 3 ckpt-model files. The accuracy rate of the model obtained ranges from 70% to 96%.

Keywords: traffic, image, detection, machine learning, convolution neural network.

1. INTRODUCTION

Traffic accidents can be caused by many things, including bad vehicle drivers, careless pedestrians, inappropriate roads such as potholes, vehicle damage, vehicles that are no longer suitable for use, motorists who violate traffic rules. One type of traffic violation is not wearing a helmet and running red lights or passing road markings. This violation is very dangerous for himself and other road users.

Visual detection of traffic violations is an important effort to minimize accidents at road intersections. Visual detection in this system is inseparable from an intelligent traffic control system which is part of an effort to regulate driving order at road junctions (Girschick, 2014 and Jalled *et al*, 2016). If a violation occurs at a crossroads, the system can respond to passing vehicles by visually detecting them through surveillance cameras at each section at the intersection (LeCun, 2010). Because there are so many traffic violations at the crossroads, we need to classify them so that we can find out what violations have occurred.

Classification is the process of finding a model or function that can distinguish classes, with the aim of being able to estimate the class of an object whose label is unknown. The classification process is usually divided into two phases, namely the learning phase and the test phase. In the learning phase, some of the data that the data class has known is fed to form an approximate model. In the test phase, the model that has been formed is tested with some other data to determine the accuracy of the model. If the accuracy is sufficient, this model can be used to predict unknown data classes (Han *et al*, 2006).

Deep Learning is a new area of Machine Learning research, which has been introduced with the aim of moving Machine Learning closer to Artificial Intelligence. Convolutional Neural Network (CNN) is a deep learning method that can be applied to perform image classification (Arabi, 2019; Deng *et al*, 2014 and Perkovic, 2012). This method has been used, among others, in image recognition, computer vision, and Natural Language Processing. The purpose of this study is to implement a deep learning method to classify traffic violation images using a convolutional neural network algorithm. Research on the detection of helmet use has been carried out by other researchers using raspberry pi and machine learning methods as an effort to enforce traffic laws. In addition, other studies use surveillance cameras to detect helmet use (Perkovic, 2019; Dahiya *et al*, 2016; Raj *et al*, 2018; Vishnu *et al*, 2017; Wonghabut *et al*, 2018, and Zhang *et al*, 2016). In addition, the use of surveillance cameras which are used to detect moving objects has been carried out research (Wang et *al*, 2004). Then the development of research on the use of surveillance cameras and deep learning to detect cellular phone use when driving has been carried out by several researchers (Seshadri *et al*, 2015 and Le *et al*, 2016). In addition, several researches to detect traffic violations have been carried out by several researchers using surveillance cameras and deep learning methods (Ibadov *et al*, 2019; Christian *et al*, 2016, Zhang *et al*, 2019 and Kumar *et al*, 2015).

2. THEORY

The application of artificial intelligence has now entered human life. Artificial intelligence or Artificial Intelligence (AI) is a field of science that studies how to build computer systems that apply intelligence in several ways. Currently, there are many studies on artificial intelligence, one of which is machine learning. Machine Learning itself is a branch of Artificial Intelligence which focuses on systems that sweep learning from data.

Machine learning was first defined by Arthur Samuel in 1959. According to Arthur Samuel, machine learning is a field of study that provides learning capabilities on computers without being explicitly programmed. Deep learning is part of machine learning that centers on modeling algorithms for high-level abstraction in data sets, using several layers of nonlinear transformations. The most well-known and used group of deep learning algorithms is the convolutional neural network because of its powerful application in recognizing different patterns, especially in detecting objects in digital images (Khoma *et al*, 2018).

Convolutional Neural Network is a multilayer neural network with a supervised learning architecture that consists of two parts, namely extractor features and classifier that can be trained. The feature extractor contains map layers and retrieves different features from the raw image through two operations, namely local receptive field and shared weights. Classifiers and weights were studied in local extractor features trained by back-propagation (Liang *et al*, 2019). In the feature extraction layer, there are two layers, namely the convolution layer and the pooling layer. In the convolution layer, convolution operations are carried out while the pooling layer is usually carried out by maxpooling operations. At the classifier layer, there is a fully connected layer that is connected to all data input. CNN has a high network depth and is often applied to image data so that it is included in the Deep Neural Network as shown in Figure-1 (Nielsen, 2018).

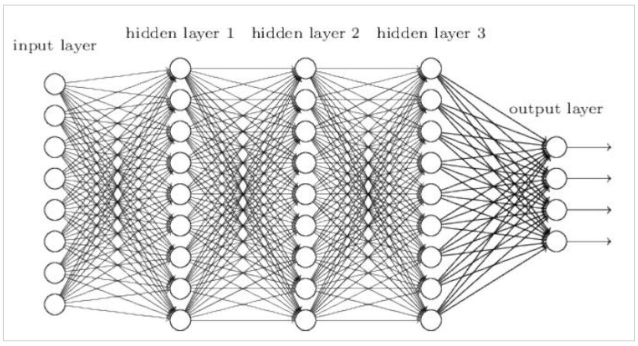


Figure-1. CNN network, Nielsen (2018).

Pooling layers are a typical technique widely used by image processing schemes. Pooling layers aim to reduce feature resolution. The pooling layer operation is carried out through the two-dimensional α × α operation of the convolutional network by taking the average or maximum value from the matrix as shown in Figure-2 (Hijazi *et al*, 2015 and Kim, 2017).

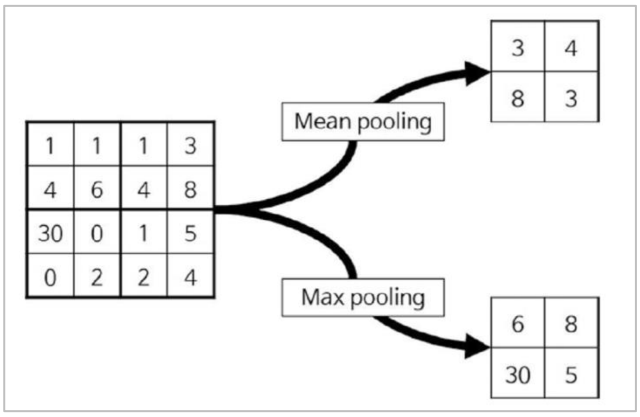


Figure-2. Pooling layer generated using two methods Simonyan *et al* (2015).

$$\text{Mean pooling} = \frac{\text{Number value submatrix}}{\text{Number matrix}} = \frac{1+1+4+6}{4} = 3 \quad (1)$$

$$\text{Max pooling} = \text{ Nilai maksimum submatrik} = 6 \quad (2)$$

After all the images have been added, it will form a new layer with the image size of 2x2 (Simonyan *et al*, 2015) Figure 2.10 shows the mean and max pooling values.

The fully connected layer is often used as the final CNN layer. This layer mathematically sums the weights of the previous feature layer which determines the calculation result of each element of each target output feature specifically (Hijazi *et al*, 2015). Figure-3 illustrates the fully connected layer L. The L-1 layer has two features, each of which is 2x2, that is, it has four elements. Layer L has two features, each of which has a single element.

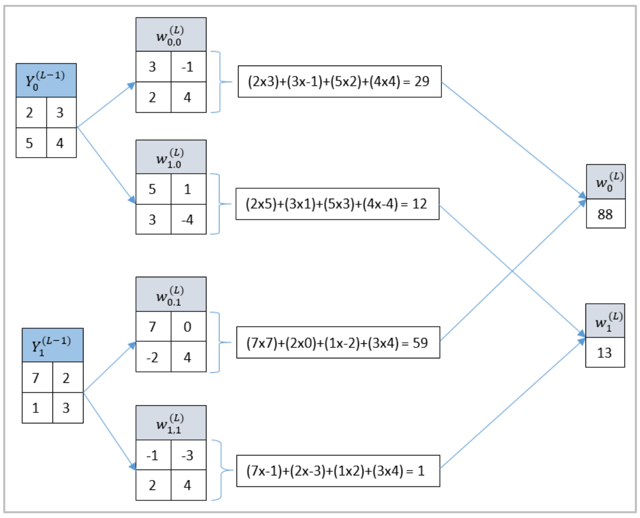


Figure-3. Fully connected layers, Simonyan *et al* (2015).

The classification is obtained from several interconnected layers of the image based on extracted features (Hijazi *et al*, 2015 and Simonyan *et al*, 2015). After using a combination of convolution and pooling layer, the output can be fed to the fully connected layer for efficient classification. The activation function in this layer is called softmax (Talo *et al*, 2019) which can be expressed as:

$$\mathrm{Sofmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)} \quad (3)$$

Softmax's function is to provide a classification output in excess of the range (0.1), which ensures neuron activation.

Tensorflow is a library that is used for deep learning implementation and uses the python programming language. Tensorflow has a feature to run model train using CPU and GPU. In this research, tensorflow is used as the back-end of Keras. This means that in this CNN process, low-level operations such as tensor and convolution will use the tensorflow library and the search for the right hyperparameter is used by the hard library. This is because tensorflow is included in the low-level, whereas hard is a high-level building for developing deep learning.

3. MATERIAL AND METHODS

This research was conducted by implementing deep learning for the classification of traffic violations using the Convolutional Neural Network algorithm. Capture and collection of data using a camera with a resolution of 13 megapixels. Data processing is carried out through a training process so that it requires a high-quality computer to get a result that can classify objects properly. The general working principle can be explained as follows. This system will work when a vehicle violates traffic, such as a violation of not wearing a helmet for motorcyclists and passing road markings for motorbikes and cars. This system is placed at a crossroads as shown in Figure-4.

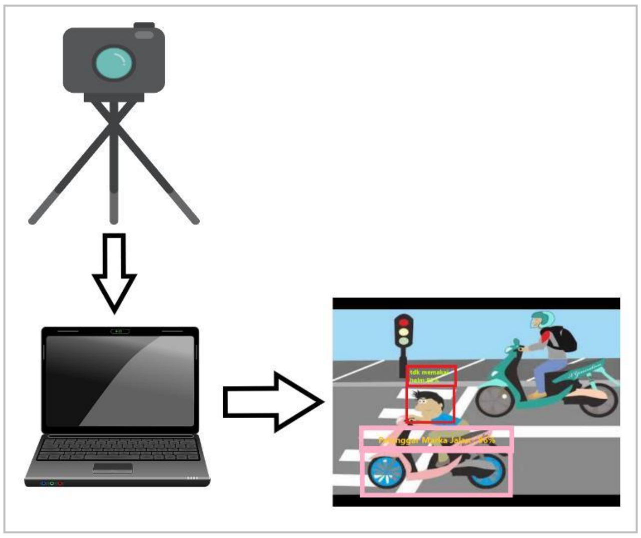
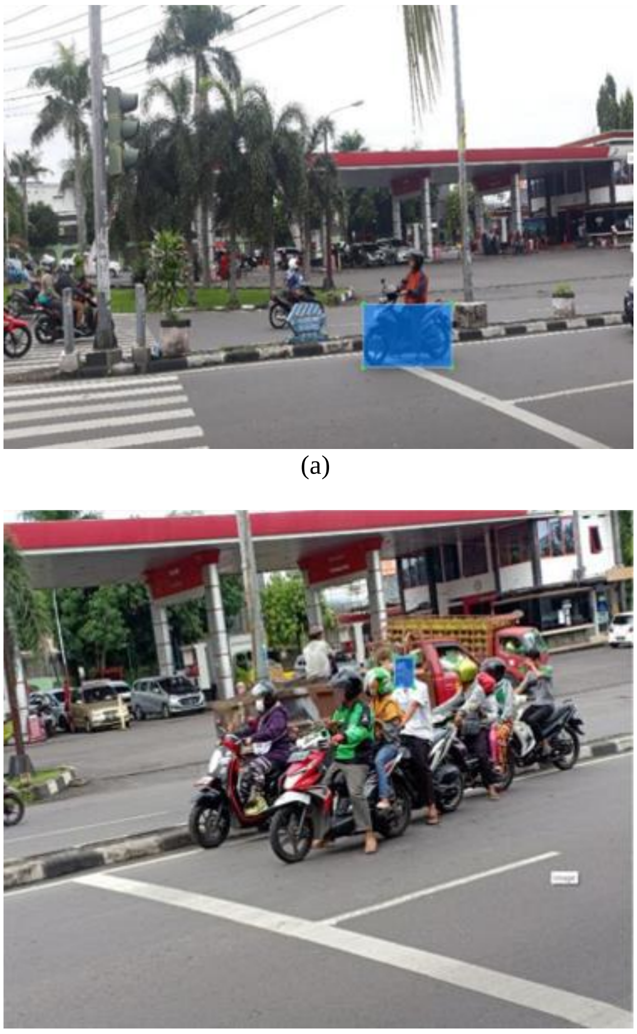


Figure-4. System design.

The data consisted of 2,265 images of which 1,812 (80%) were used for training and 453 (20%) were used for testing. Furthermore, on the training data, labeling is done with the label "Not wearing a helmet" and "road marking offenders". After that the data is entered into the neural network architecture for the training process (training). At this stage the Neural Network is trained to learn patterns that produce object detection with a high degree of accuracy. This training process will generate automatic checkpoints by Tensorflow in the form of tensor graphs to store information on the training process. When the training process is complete, then export the graph tensor so that it becomes a model that will be ready to be used to detect objects. The detection of objects will result in an image that is carried out in the testing process which is taken through the camera and displayed on a computer. The labeling process is carried out manually on each training image. Figure-5 shows an example of labeling.

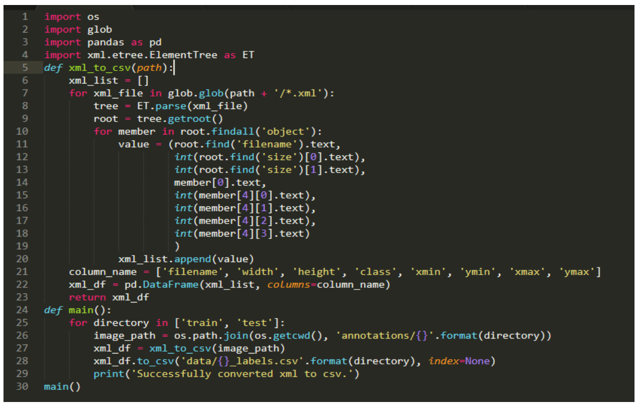


(a)

(b)

Figure-5. Example of labeling. a) labeling for helmet wearing violations b) labeling for road marking violations.

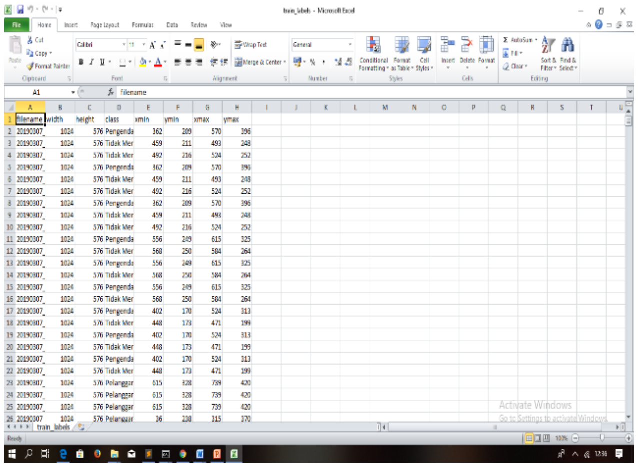
After labeling, we convert from XML to CSV (Comma Sepparated Value) to convert the dataset to TFRecord. The XML to CSV data conversion code can be seen in Figure-6.



```python
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
def main():
    for directory in ['train', 'test']:
        image_path = os.path.join(os.getcwd(), 'annotations/{}'.format(directory))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv('data/{}_labels.csv'.format(directory), index=None)
        print('Successfully converted xml to csv.')
main()
```

Figure-6. XML to CSV conversion code.

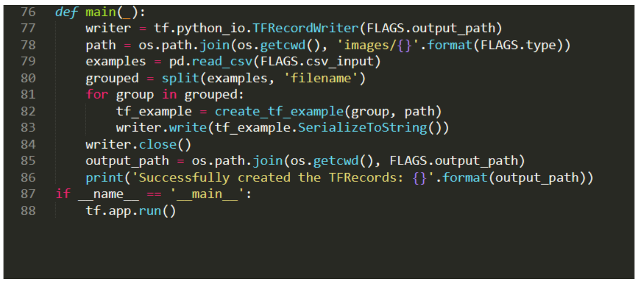
While the data display of the conversion results from xml to csv can be seen in Figure-7.



filename	width	height	class	xmin	ymin	xmax	ymax
20190307_	1024	576	Pengenda	362	209	570	396
20190307_	1024	576	Tidak Mer	459	211	493	248
20190307_	1024	576	Tidak Mer	492	216	524	252
20190307_	1024	576	Pengenda	362	209	570	396
20190307_	1024	576	Tidak Mer	459	211	493	248
20190307_	1024	576	Tidak Mer	492	216	524	252
20190307_	1024	576	Pengenda	362	209	570	396
20190307_	1024	576	Tidak Mer	459	211	493	248
20190307_	1024	576	Tidak Mer	492	216	524	252
20190307_	1024	576	Pengenda	556	249	615	325
20190307_	1024	576	Tidak Mer	568	250	584	264
20190307_	1024	576	Pengenda	556	249	615	325
20190307_	1024	576	Tidak Mer	568	250	584	264
20190307_	1024	576	Pengenda	556	249	615	325
20190307_	1024	576	Tidak Mer	568	250	584	264
20190307_	1024	576	Pengenda	402	170	524	313
20190307_	1024	576	Tidak Mer	448	173	471	199
20190307_	1024	576	Pengenda	402	170	524	313
20190307_	1024	576	Tidak Mer	448	173	471	199
20190307_	1024	576	Pengenda	402	170	524	313
20190307_	1024	576	Tidak Mer	448	173	471	199
20190307_	1024	576	Pelanggar	615	328	739	420
20190307_	1024	576	Pelanggar	615	328	739	420
20190307_	1024	576	Pelanggar	615	328	739	420
20190307	1024	576	Pelanggar	36	238	315	370

Figure-7. The results of the XML to CSV conversion.

After the XML to CSV conversion process then the conversion to TFRecord is used for data feeding in the training process, the TFRecord creation code can be seen in Figure-8.



```python
def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), 'images/{}'.format(FLAGS.type))
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords: {}'.format(output_path))
if __name__ == '__main__':
    tf.app.run()
```

Figure-8. TFRecord creation code.

The next step is to set some parameters in the object detection training pipeline configuration. Tensorflow'sapi Object detection uses protobuf files to configure the training and evaluation process. The configuration file is divided into 5 parts, namely:

a) **Model:** defines what type of model to train (i.e. meta-architecture, feature extractor)

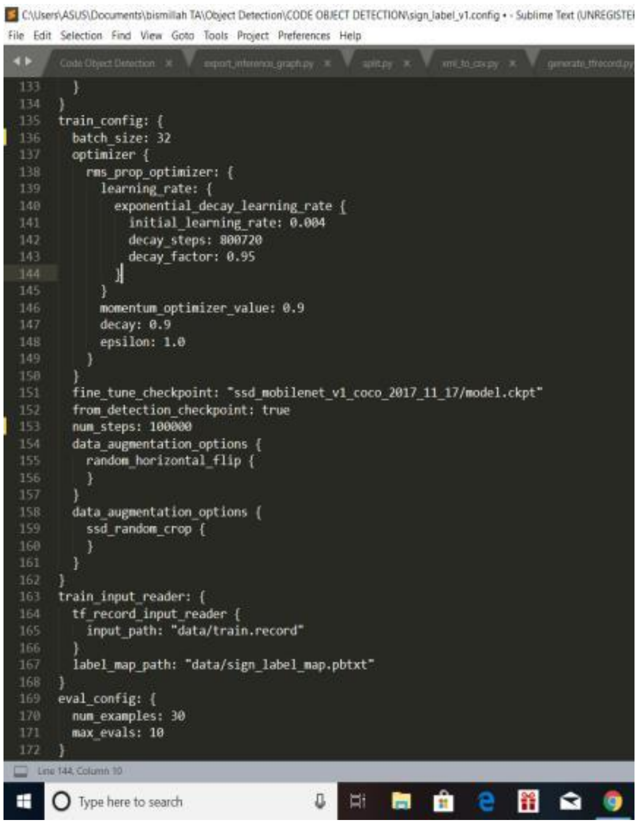
b) **Training_config:** specifies what parameters should be used to train model parameters (e.g. SGD parameters, input preprocessing and feature extractor initialization values).

c) **Eval_config:** determines what measurement matrices will be reported for evaluation.

d) **Train_input_config:** defines what dataset the model should train on.

e) **Eval_input_config:** defines what dataset the model will evaluate. Usually this should be different from the input dataset for training.

The pipeline configuration code can be seen in Figure-9.



```
  }
}
train_config: {
  batch_size: 32
  optimizer {
    rms_prop_optimizer: {
      learning_rate: {
        exponential_decay_learning_rate {
          initial_learning_rate: 0.004
          decay_steps: 800720
          decay_factor: 0.95
        }
      }
      momentum_optimizer_value: 0.9
      decay: 0.9
      epsilon: 1.0
    }
  }
  fine_tune_checkpoint: "ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
  from_detection_checkpoint: true
  num_steps: 100000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    ssd_random_crop {
    }
  }
}
train_input_reader: {
  tf_record_input_reader {
    input_path: "data/train.record"
  }
  label_map_path: "data/sign_label_map.pbtxt"
}
eval_config: {
  num_examples: 30
  max_evals: 10
}
```

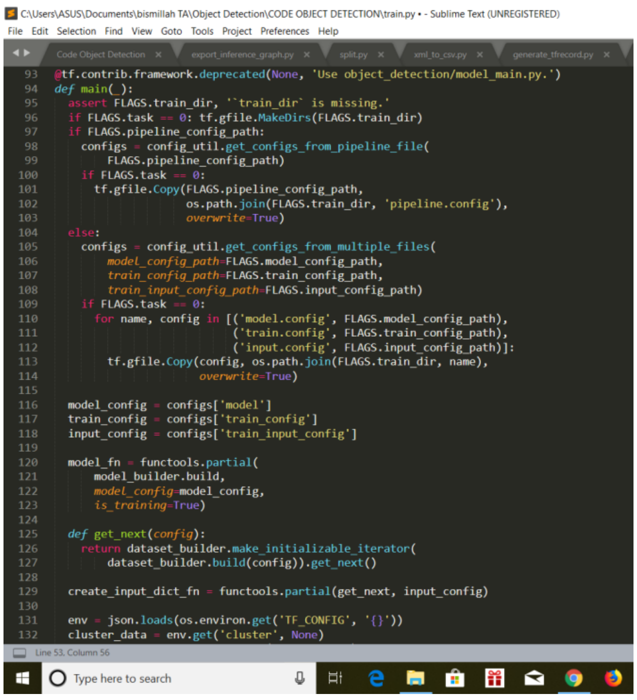
Figure-9. Pipeline configuration code.

The pipeline configuration used is a mobilenet SSD. The following are the parameters that must be set as in Table-1.

Table-1. Object detection pipeline configuration parameters.

Parameters	Value	Information
num_classes	2	Number of classes
batch_size	32	Number of data processed at each step
learning_rate	0,004	Learning rate
decay_steps	800720	Schedule decay steps, i.e. lowering the level of learning by a factor over time
num_steps	100.000	Number of steps in the process training
kernel_size	3	Number of layers to count and detect a pattern used during the convolution process.

The next stage is the Neural Network training stage, at this stage the Neural Network is trained to learn patterns that produce object detection recognition with high accuracy. The program code in the training process can be seen in Figure-10.


C:\Users\ASUS\Documents\bismillah TA\Object Detection\CODE OBJECT DETECTION\train.py • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
Code Object Detection
export_inference_graph.py
split.py
xml_to_csv.py
generate_tfrecord.py
Line 53, Column 56
Type here to search

```python
@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
  assert FLAGS.train_dir, '`train_dir` is missing.'
  if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
        FLAGS.pipeline_config_path)
    if FLAGS.task == 0:
      tf.gfile.Copy(FLAGS.pipeline_config_path,
                    os.path.join(FLAGS.train_dir, 'pipeline.config'),
                    overwrite=True)
  else:
    configs = config_util.get_configs_from_multiple_files(
        model_config_path=FLAGS.model_config_path,
        train_config_path=FLAGS.train_config_path,
        train_input_config_path=FLAGS.input_config_path)
    if FLAGS.task == 0:
      for name, config in [('model.config', FLAGS.model_config_path),
                           ('train.config', FLAGS.train_config_path),
                           ('input.config', FLAGS.input_config_path)]:
        tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                      overwrite=True)

  model_config = configs['model']
  train_config = configs['train_config']
  input_config = configs['train_input_config']

  model_fn = functools.partial(
      model_builder.build,
      model_config=model_config,
      is_training=True)

  def get_next(config):
    return dataset_builder.make_initializable_iterator(
        dataset_builder.build(config)).get_next()

  create_input_dict_fn = functools.partial(get_next, input_config)

  env = json.loads(os.environ.get('TF_CONFIG', '{}'))
  cluster_data = env.get('cluster', None)
```

Figure-10. Code in the training process.

4. RESULTS AND DISCUSSIONS

Monitoring the training process using a Tensorboard in which there is a graphic. Tensorboard calls the checkpoint files generated during the training process in the training folder. Tensorboard can be opened in a web browser by typing the address and port of the tensorboard at the command prompt. The results on the Tensorboard can be seen in Figure-11 and Figure-12.

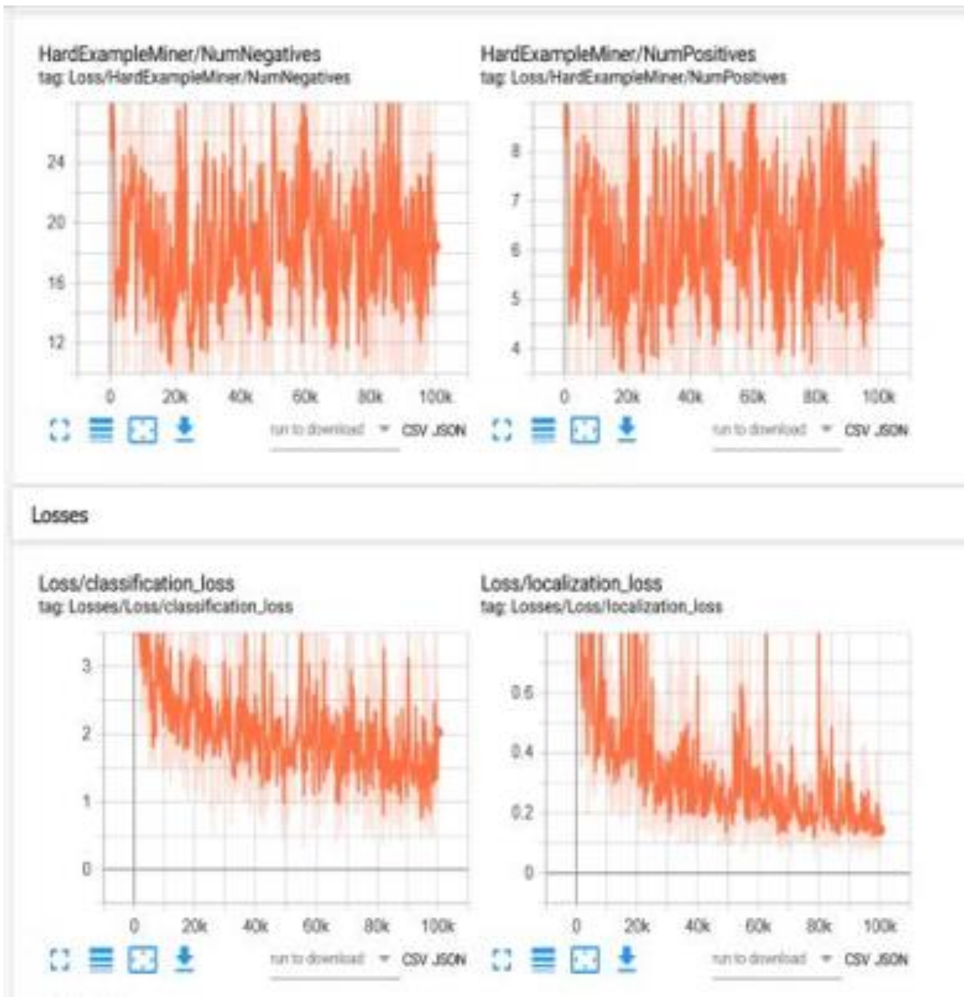

HardExampleMiner/NumNegatives
tag: Loss/HardExampleMiner/NumNegatives
HardExampleMiner/NumPositives
tag: Loss/HardExampleMiner/NumPositives
Losses
Loss/classification_loss
tag: Losses/Loss/classification_loss
Loss/localization_loss
tag: Losses/Loss/localization_loss
CSV JSON

Figure-11. Tensorboard view.

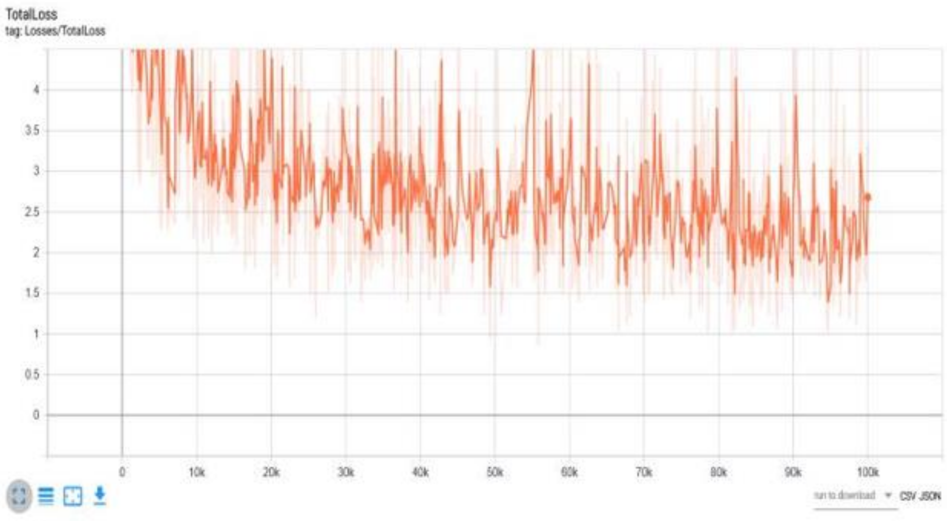


Figure-12. Graph of classification loss.

The output on the Neural Network that represents the data flow in the form of a graph, the graph that has been previously trained is a checkpoint during the training process then exported to the inference graph. The results of the detection of the type of violation without wearing a helmet are shown in Figure-13.

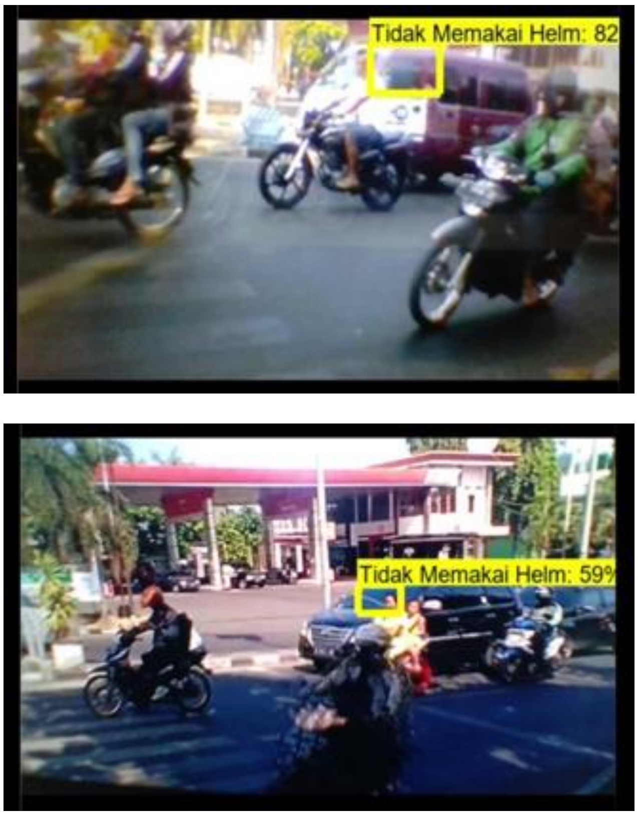
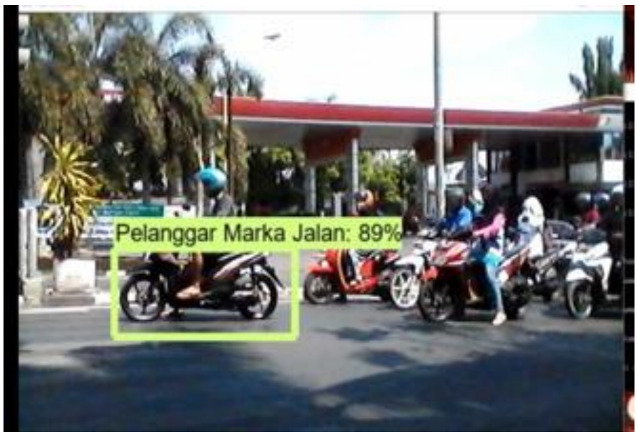


Figure-13. Example of violation detection of not wearing a helmet.

Meanwhile, the results of violation testing passing road markings are shown in Figure-14. As shown in the figure above, the system can detect the types of road markings violations. So that overall 2 types of violations can be detected by the system, namely violations of not using helmets and road markings for 2-wheeled vehicles.



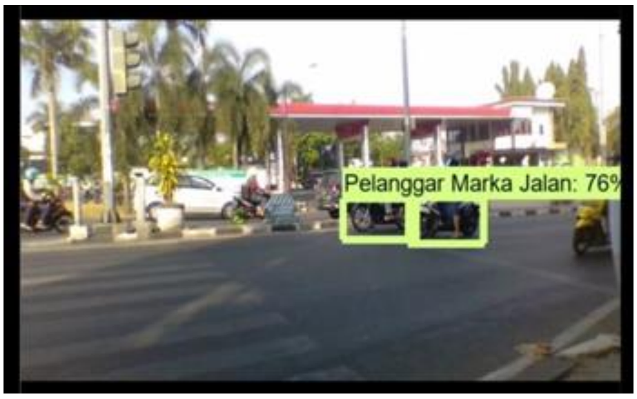


Figure-14. Example of road marking violation detection.

Whereas in the classification loss graph, it can be seen that in the training process CNN can read or not classify what we use in the dataset and in the localization loss in the CNN training process can read or not a position we use in the dataset and for total loss is the result of all losses that can be obtained during the training process. After seeing the total loss in the training process, then extract the model from the checkpoints generated during the training process. After that, a real time webcam call can be made to run the detection program. After the experiment on the camera in real time is detected, the next step is to call the video that has been programmed. The following (Table-2 and Table-3) is an example of a table on camera detection results that is carried out in real time and the level of accuracy in detecting traffic violations.

Table-2. Example of detection results on camera.

No	Method	Visual	Result
1	Violators of road markings	Violators of road markings	Correct
2	Violators of road markings	Violators of road markings	Correct
3	Violators of road markings	Violators of road markings	Correct
4	Not wearing a helmet	Wrong	Wrong
5	Not wearing a helmet	Wrong	Wrong
6	Not wearing a helmet	Wrong	Wrong

Table-3. Accuracy on traffic violation classification.

No	Classification	Accuracy
1	Violators of road markings	96%
2	Violators of road markings	88%
3	Violators of road markings	76%
4	Not wearing a helmet	82%
5	Not wearing a helmet	59%
6	Not wearing a helmet	an invisible

5. CONCLUSIONS

In this study we succeeded in making a training model with a total of 100 000 steps with 32 batch sizes. The results of detecting the classification of traffic violations on a digital image using a Convolutional Neural Network can be considered to be working well. The accuracy level of the model obtained from the detection of traffic violation image classification in a digital image using the Convolutional Neural Network ranges from 70% to 96%. Based on the research that has been done, suggestions can be given to add the number of datasets and the variety of objects to the image to train the model and achieve high accuracy. Object recognition needs to be developed with a focus on detecting or recognizing vehicle classes. If possible, we can increase the specifications for higher devices by using a computer with a high Random Access Memory (RAM) and using the Graphics Processing Unit (GPU) to speed up the training process.

ACKNOLEDGEMENT

This research was funded by the Development and Implementation Research Scheme, Diponegoro University in 2020.

REFFERENCES

Arabi S. 2019. Adeeplearning based solution for construction equipment detection: from development to deployment. Department of Civil, Construction and Environmental Engineering Iowa State University.

C. Vishnu, D. Singh, C. K. Mohan and S. Babu. 2017. Detection of motorcyclists without helmet in videos using convolutional neural network. In International Joint Conference on Neural Networks (IJCNN), pp. 3036-3041. IEEE.

Deng L. & Yu D. 2014. Deep Learning: Methods and Applications, Foundations and Trendsin Signal Processing. 7, 3-4, pp. 197-387.

G. Umapathy., H. Yuvaraja., S. Vijaya Kumar, S. Parameshwaran and M. Ramya Princess. 2019. Violation Prevention of Traffic Rules and Helmet Detection using Machine Learning, International Journal of Engineering Science and Computing. 9(3).

Girschick R. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. USA: IEEE.

Han J. & Kamber M. 2006. Data mining: Concepts and Techniques, San Fransisco: Elsevier.

Jalled F. & Voronkov I. 2016. Object Detection Using Image Processing, Ithaca, New York: Cornell University Library.

K. Dahiya, D. Singh and C. K. Mohan. 2016. Automatic detection of bikeriders without helmet using surveillance videos in real-time, in Proc. Int. Joint Conf. Neural Networks (IJCNN), Vancouver, Canada, 24-29 July 2016, pp. 3046-3051.

K. D. Raj, A. Chairat, V. Timtong, M. N. Dailey and M. Ekpanyapong. 2018. Helmet violation processing using deep learning, In International Workshop on Advanced Image Technology (IWAIT). pp. 1-4.

K. He, X. Zhang, S. Ren and J. Sun. 2016. Deep residual learning for image recognition. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778.

Khoma V., Pelc M., Khoma Y. and Sabodashko D. 2018. Outlier Correction in ECG - Based Human Identification, Proceedings of the 3rd International Scientific Conference on Brain-Computer Interfaces, Poland: 13 - 14 March 2018, pp. 11-21.

Kim Phil. 2017. MATLAB Deep Learning MATLAB Deep Learning. ed. Kezia Endsley Todd Green, Steve Anglin, Matthew Moodie, Mark Powers. Apress.

Le Cun Y. 2010. Convolutional networks and applications in vision, ISCAS2010-2010 IEEE International Sympo siumon Circuits and Systems: Nano-Bio.

Le T. H. N., Zheng Y., Zhu C., Luu K. and Savvides M. 2016. Multiple Scale Faster-RCNN Approach to Driver's Cell-phone Usage and Hands on Steering Wheel Detection, in CVPR.

Liang W., Zhang H., Zhang G., dan Cao H. 2019. Rice Blast Disease Recognition Using a Deep Convolutional Neural Network, Scientific Reports. 9(1): 2869.

Nielsen M. 2018. Neural Networks and Deep Learning. In Artificial Intelligence.

P. Wonghabut, J. Kumphong, T. Satiennam, R. Ungarunyawee and W. Leelapatra. 2018. Automatic helmet-wearing detection for law enforcement using CCTV cameras. In IOP Conference Series: Earth and Environmental Science, volume 143. IOP Publishing, 2018.

P.M., Ashok Kumar, V., Sathya and V., Vaidehi. 2015. Traffic Rule Violation Detection in Traffic Video Surveillance. International Journal of Computer Science and Electronics Engineering (IJCSEE) 3(4), ISSN 2320-4028 (Online).

Perkovic L. 2012. Introductionto Computing Using Python: An Application Development Focus. London: Wiley.

P. U., Aron Christian, *et al*. 2016. Automated traffic violation apprehension system using genetic algorithm and artificial neural network, 2016 IEEE Region 10 Conference (TENCON). pp. 2094-2099.

Samer Hijazi, Rishi Kumar and Chris Rowen, IP Group, Cadence. 2015. Cadence Using Convolution Neural Networks for Image Recognition. United States: Cadence Design Systems.

Seshadri K., Juefei-Xu F., Pal D. K., Savvides M., Thor C. 2015. Driver Cell Phone Usage Detection on Strategic Highway Research Program (shrp2) Face View Videos, In Proc. IEEE Conf. CVPRW. pp. 35-4.

Simonyan Karen and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings: 1-14.

S.R. Ibadov, B.Y. Kalmykov, R.R. Ibadov and R.A. Sizyakin. 2019. Method of Automated Detection of Traffic Violation with a Convolutional Neural Network, EPJ Web of Conferences 224, 04004 (2019).

Talo Muhammed *et al*. 2019. Convolutional Neural Networks for Multi-Class Brain Disease Detection Using MRI Images. Computerized Medical Imaging and Graphics. 78: 101673.

W. Hu, T. Tan, L. Wang and S. Maybank. 2004. A survey on visual surveillance of object motion and behaviors, IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews. 34(3): 334-352.

Z. Zhang, C. Trivedi and X. Liu. 2018. Automated detection of grade-crossing-trespassing near misses based on computer vision analysis of surveillance video data, Safety science. 110 276-285.