

Deep Neural Network for Visual Localization of Autonomous Car in ITS Campus Environment

Rudy Dikairono

Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
rudydikairono@ee.its.ac.id

Hendra Kusuma

Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
hendraks@gmail.com

Arnold Prajna

Department of Electrical Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
arnold.18071@mhs.its.ac.id

Abstract— Intelligent Car (I-Car) ITS is an autonomous car prototype where one of the main localization methods is obtained through reading GPS data. However the accuracy of GPS readings is influenced by the availability of the information from GPS satellites, in which it often depends on the conditions of the place at that time, such as weather or atmospheric conditions, signal blockage, and density of a land. In this paper we propose the solution to overcome the unavailability of GPS localization information based on the omnidirectional camera visual data through environmental recognition around the ITS campus using Deep Neural Network. The process of recognition is to take GPS coordinate data to be used as an output reference point when the omnidirectional camera takes images of the surrounding environment. Visual localization trials were carried out in the ITS environment with a total of 200 GPS coordinates, where each GPS coordinate represents one class so that there are 200 classes for classification. Each coordinate/class has 96 training images. This condition is achieved for a vehicle speed of 20 km/h, with an image acquisition speed of 30 fps from the omnidirectional camera. By using AlexNet architecture, the result of visual localization accuracy is 49-54%. The test results were obtained by using a learning rate parameter of 0.00001, data augmentation, and the Drop Out technique to prevent overfitting and improve accuracy stability.

Keywords — catadioptric camera, deep neural network, GPS, intelligent car (i-car) ITS, visual localization

I. INTRODUCTION

ITS intelligent Car (I-Car) is an autonomous car prototype that relies on GPS data as one of its main localization methods. However, the accuracy of GPS readings can be influenced by various factors such as weather conditions, signal blockage, and land density, leading to unreliable location information. To address the challenge of unavailability and inconsistency of GPS localization, this paper proposes a solution based on using omnidirectional camera visual data and Deep Neural Network (DNN) for environmental recognition around the ITS campus.

Visual localization has traditionally been applied indoors, but this research aims to explore its effectiveness in outdoor environments, specifically for the I-Car ITS autonomous car. By integrating an omnidirectional camera with the I-Car, the camera captures the surrounding environment during operation. The captured images serve as a dataset, which is processed along with the GPS coordinates readings. This combined information is then compared to determine the I-Car's position at a given time, serving as an alternative or backup to GPS-based localization.

To place this research in context, outdoor visual localization has been a subject of interest in the field of autonomous vehicles. Several studies have explored the application of visual data for accurate localization in outdoor environments. For instance, Lin et al. [1] utilized deep learning techniques based on ConvNet Descriptor and Global Optimization to achieve robust visual localization for assisted navigation. In order to attain a reliable image representation, they assess different layers obtained from five commonly used Convolutional Neural Networks (ConvNets) based on their ability to withstand diverse environmental variations. Their findings indicate that the robustness of ConvNet-based descriptors does not exhibit a direct correlation with object classification accuracy or filter size. Another relevant study by Yun et al. [2] they have developed a new optical navigation sensor capable of measuring a mobile robot's transverse distance accurately, unaffected by external influences. Leveraging this optical navigation sensor, they have devised a novel localization algorithm that combines Inertial-Measurement-Unit (IMU) and GPS data. The algorithm has been thoroughly tested in urban environments characterized by frequent GPS signal interruptions and rough ground surfaces, which pose significant disturbances.

In the context of ITS campuses, outdoor visual localization has garnered attention due to its potential to complement existing non-visual localization methods. LIDAR, odometry, and GPS are commonly employed in ITS environments, but visual localization offers an additional layer of redundancy and accuracy assessment. By comparing the accuracy of all non-visual sensors with the visual localization approach, the I-Car can enhance its positioning capabilities and provide a more reliable and comprehensive autonomous driving experience.

II. METHODS

The method used for visual localization in this research is a classification of environmental images based on coordinates obtained from GPS. Figure 3.1 shows that there are a total of 200 points which are the coordinates of the points on the path determined by GPS. The distance between points is about 5 meters based on point picks with an average vehicle speed of 20km/hour. This is in accordance with the GPS resolution used. Environmental images are captured by the omnidirectional camera at a rate of 30 fps.

It aims to match each point of GPS coordinates with images obtained from omnidirectional cameras. For a speed

of 20km/hour, the average images obtained is 28 images (frames) per point of GPS coordinates.

$$\text{SPEED} = 20\text{km/hour} = 20000 :3600 = 5.5 \text{ m/s}$$

$$5.5\text{m/s} \cong 30 \text{ fps}$$

So for every distance of 5.5 meters, represented by 30 images (frames). Therefore every 5 meters (1 point GPS coordinates) is proportional to:

$$\frac{30}{5.5} \times \frac{\text{frame}}{\text{5meter}} = 277.272727 \text{ frame/coordinate}$$

$$\cong 28 \text{ frame/point}$$

Because the length of the path taken produces 200 points, this image classification is classified into 200 classes (multi-class image classification). Each class represents one coordinate point. Here is a picture of the ITS I-CAR trajectory in the ITS campus environment:



Fig. 3.1. The Process of Pointing Points on OpenStreetMap Results of Collecting Datasets around the ITS Campus Environment

LEARNING PROCESS (TRAINING) MODEL

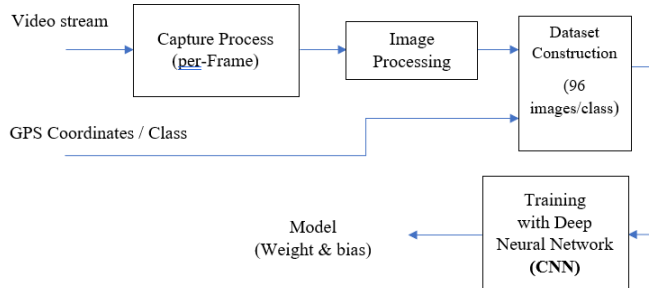


Fig. 3.2. Visual Localization Training Process Block Diagram

A. Image frame capture process (Image Acquisition)

Fig. 3.3 is the process of taking image data by recording the road around ITS using an omnidirectional camera, precisely around the Mannarul area, ITS rectorate, and alumni park. The original image resolution of the video is (1024px by 768px). The resulting frame rate is 30 fps. The recording process was taken using a private vehicle (motorcycle) because, during the data collection process, I-CAR ITS still couldn't run. The speed of the vehicle is 20km/hour.

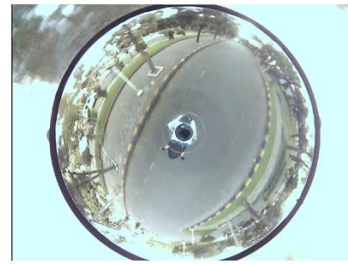


Fig. 3.3. Data collection process (1024 x 768 Pixel)

B. GPS Coordinate Data Collection Process



Fig. 3.4. Process of reading GPS coordinates via Q-Ground Control

Fig. 3.4. is the process of collecting GPS coordinates data using a GPS module (U-Blox ZED-F9P Series) which is equipped with RTK (real-time kinematic) to produce high accuracy up to centimeter level; RFD 900X telemetry for serial communication between GPS Base Station and vehicle-mounted GPS Rover Station; Hex Cube Black flight controller (Pixhawk 2.1) used to read NMEA GPS data from the GPS module using the Q-Ground Control application. Through Q-Ground Control, a data log regarding the results of the trip will be created and stored in the form of a csv file.

gps.lat	gps.lon	clock.currentTime	
-7.2828414	112.7933837	nan	
-7.2828432	112.7933835	15:11:57	
-7.2828423	112.7933838	15:11:58	
-7.2828435	112.7933854	15:11:59	
-7.2828475	112.7933853	15:12:00	
-7.2828432	112.7933862	15:12:01	
-7.282839	112.7933872	15:12:02	
-7.2828357	112.7933867	15:12:03	
-7.2828357	112.7933873	15:12:04	2022-05-31 15-11-57 vehicle1
-7.2828372	112.7933865	15:12:05	2022-05-31 15-19-13 vehicle1
-7.2828399	112.7933863	15:12:06	2022-06-04 07-43-00 vehicle1
-7.2828399	112.7933848	15:12:07	
-7.2828413	112.7933846	15:12:08	2022-06-04 08-21-02 vehicle1

Fig. 3.5. Data log Coordinate reading of Q-Ground Control

C. Determination of the GPS reading data log through the Folium Library Visualization

The folium library serves as a map visualization using Open Street Map. First of all, the GPS data log will be read through pandas.read_csv ('file_name.CSV) to be able to display the coordinates of the location where the location is located, Latitude and Longitude data are needed from the data log of trips that have been made.

The following are the results of the four data logs generated during the dataset retrieval

process.



Fig. 3.6. take_1.csv (a), take_2.csv (b), take_3.csv (c), take_4.csv (d)

After the tracking results are obtained as shown in Fig. 12, then one of the most accurate tracking results will be selected as the identity of the class.

D. Video Editing & Generalization Process

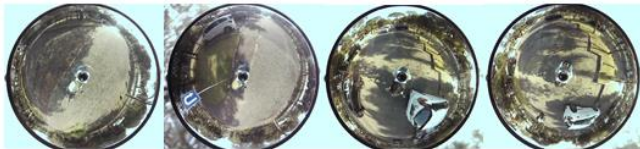


Fig. 3.7. taking pictures with variations in position and time differences

The recording process is in Fig. 3.7. conducted using an omnidirectional camera is four times where each recording is done at a different time and date (morning and evening) as well as on different sides (right and left sides of the road). In order for the length of the image data from the recording and the results of the GPS reading to be the same, it is equated through the time that is read (clock.currenttime) in the data log. Several lengths of duration between videos are also different, so it is necessary to edit the duration cuts and the similarity of the position of the starting point of the journey so that a good and clear dataset can be obtained.

E. Automatic Class Folder Processing and Labeling for Preprocessing Data

At this step, programming will be carried out to generate a folder creation system and class labeling from video image data and GPS coordinate data log automatically. The library used is OS to process the directories contained in the computer; cv2 for visual image processing; NumPy for numerical calculations and so on; pandas to read GPS csv data files; glob to return all file paths associated with a specific pattern.

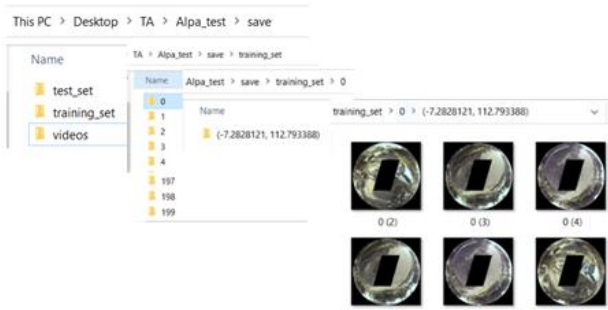


Fig. 3.8. Automatic Class Folder Processing and Labeling

Number of classes produced is based on the travel time during the dataset retrieval process with one round in the area around the ITS rectorate. Each coordinate has training data and testing/validation data with a total of 96 images on the training data and 24 images on the testing data.

F. Region Of Interest (ROI) Image Data

The frames that are made will be made several changes to eliminate unnecessary outliers so that they do not enter into distortion inputs for image feature extraction.



Fig. 3.9. ROI generation on every Frame

In fig. 3.9. there is a trapezoid in the middle of the image created using cv2.fillConvexPoly which functions as a virtual visualization of the length and width of the ITS-I-CAR top surface. The side of the image is made black using cv2.bitwise_and which serves to remove outliers or edges that are not needed as feature input from the image. ROI is created by defining the width - a height that you want to attach to the original image (frame [w1: w2, h1: h2]).

G. Preprocessing Input Data

At this step, the input data for each image will be re-scaled to 1./255 to produce a value that can be read by the system and resized to reduce the original image resolution (1024px by 768px) to (224px by 224px) so that it can be entered into the deep model. learning. Then a shuffle was also carried out to randomize the order of images so that when the batch size iteration was carried out, the images that entered the architectural model varied. This process runs in python with the help of the TensorFlow library. There are 23,868 total data, which is divided into 80% for training data and 20% for testing data.

H. Augmentation Image Data

Another step that will be carried out during the image input preprocessing process is the image augmentation process. This technique aims to enrich the diversity of data in the training data so that the developed Deep Neural Network model can recognize the pattern (pattern) of each identity/feature of each image better.

I. Creating a Deep Learning Model

After the pre-processing step has been completed, the data is ready to be entered into the deep learning model. In the process of making this architectural model, there are main layers such as:

- Convolutional layer (filter, strides, kernel size)
- Pooling layer (max-min pool/average pool)
- Regulation/Batch normalization
- Flatten Layer
- Dense Layer
- Drop Out

The model that has been created will produce a dense layer output with a total of 200 units/class with softmax activation.

J. Using an existing DNS Architecture

In this research project, we aim to compare the results of accuracy and loss with the existing architecture. Here is the architecture we want to test:

- ResNet50
- AlexNet

```
Model: "AlexNet"
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
conv0 (Conv2D)	(None, 54, 54, 96)	34944
bn0 (BatchNormalization)	(None, 54, 54, 96)	384
activation (Activation)	(None, 54, 54, 96)	0
max0 (MaxPooling2D)	(None, 26, 26, 96)	0
conv1 (Conv2D)	(None, 26, 26, 256)	614656
bn1 (BatchNormalization)	(None, 26, 26, 256)	1024
activation_1 (Activation)	(None, 26, 26, 256)	0
max1 (MaxPooling2D)	(None, 12, 12, 256)	0
conv2 (Conv2D)	(None, 12, 12, 384)	885120
bn2 (BatchNormalization)	(None, 12, 12, 384)	1536
activation_2 (Activation)	(None, 12, 12, 384)	0
conv3 (Conv2D)	(None, 12, 12, 384)	1327488
bn3 (BatchNormalization)	(None, 12, 12, 384)	1536
activation_3 (Activation)	(None, 12, 12, 384)	0
conv4 (Conv2D)	(None, 12, 12, 256)	884992
bn4 (BatchNormalization)	(None, 12, 12, 256)	1024
activation_4 (Activation)	(None, 12, 12, 256)	0
max2 (MaxPooling2D)	(None, 5, 5, 256)	0
flatten (Flatten)	(None, 6400)	0
fc0 (Dense)	(None, 4096)	26218496
fc1 (Dense)	(None, 4096)	16781312
fc2 (Dense)	(None, 200)	819400

```

Total params: 47,571,912
Trainable params: 47,569,168
Non-trainable params: 2,752
    
```

Fig. 3.10. AlexNet Architecture Program

K. Transfer Learning

Transfer learning is a technique for using a pre-trained model (knowledge about the weight and bias value of a parameter) that architecture already has in intend to make the image of our dataset that will enter a model better recognized for its pattern to increase the accuracy of the output. The transfer learning that will be tested is ResNet50.

L. Dataset input training process into the DNN model

Several hyper tuning parameters need to be done at the model learning step to produce the desired accuracy. In fig. 3.11. shows the compile step parameters, there are hyper parameters regarding the optimizer you want to use and the type of loss used in the problem to be solved. The optimizer that will be used is 'Adam' with a variation of the learning rate, which is 0.001 (default) and also 0.00001 to see how it affects the accuracy value. The loss used is categorical_crossentropy type because the classification type is multi-class. At the fit step, input parameters in the form of 'x' which represents learning data which amounts to 80% of the dataset and 'validation_data' which represents testing data which amounts to 20% of the dataset.

```

Epoch 27: saving model to cechk_alex_0.0001/cp.cpkt
299/299 [=====] - ETA: 0s - loss: 4.6173e-04 - accuracy: 1.0000 - val_
l_loss: 1.6584 - val_accuracy: 0.4956
Epoch 28/30
299/299 [=====] - ETA: 0s - loss: 3.8886e-04 - accuracy: 1.0000
Epoch 28: saving model to cechk_alex_0.0001/cp.cpkt
299/299 [=====] - 301s 1s/step - loss: 3.8886e-04 - accuracy: 1.0000 - val_
l_loss: 1.6764 - val_accuracy: 0.4925
Epoch 29/30
299/299 [=====] - ETA: 0s - loss: 3.2854e-04 - accuracy: 1.0000
Epoch 29: saving model to cechk_alex_0.0001/cp.cpkt
299/299 [=====] - 322s 1s/step - loss: 3.2854e-04 - accuracy: 1.0000 - val_
l_loss: 1.6884 - val_accuracy: 0.4992
Epoch 30/30
299/299 [=====] - ETA: 0s - loss: 2.7977e-04 - accuracy: 1.0000
Epoch 30: saving model to cechk_alex_0.0001/cp.cpkt
299/299 [=====] - 319s 1s/step - loss: 2.7977e-04 - accuracy: 1.0000 - val_
l_loss: 1.7270 - val_accuracy: 0.4956
    
```

Fig. 3.11. Compile and Fit Functions for the Training Process

TESTING PROCESS (DEPLOYMENT) MODEL

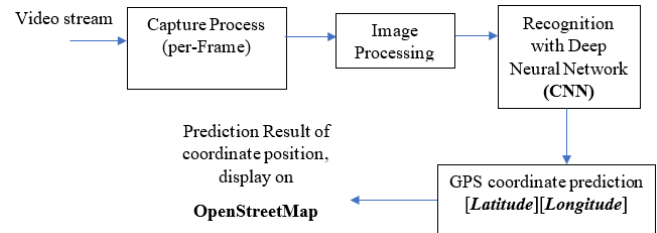


Fig. 3.12. Visual Localization Deployment Process Block Diagram

In the model testing process, the initial steps are the same as in the model training process, that is image data retrieval by recording the previously studied path, which is around the Mannarul area, ITS rectorate, and ITS alumni park. During the recording process, image processing also takes place where the image is made to remove outliers, and also virtual visualization of the length and width of the I-Car, and resolution changes (224px by 224px). From this step, the data is ready to be entered into the DNN model that has been built using previously trained weights and biases.

M. Creating a Map with Folium from Output Picture

In case the training process has been completed, then the next step is to test the predictions in the previous area to prove whether the predicted output displayed is by the actual data. The predicted output results are in the form of index numbers 0 to 199 which represent the class of each GPS coordinate that has been taken previously. This index will then be accessed first to go to the actual directory to get the Latitude and Longitude GPS coordinate label attached to each folder. Then the label in the form of Latitude and Longitude coordinates will be included in the program assisted by the Folium library to display the point of the GPS coordinates in the form of an Open Street Map.

```

1/1 [=====] - 0s 24ms/step
(-7.282783, 112.7933972)
1/1 [=====] - 0s 19ms/step
(-7.282783, 112.7933972)
1/1 [=====] - 0s 14ms/step
(-7.2828098, 112.7933867)
1/1 [=====] - 0s 18ms/step
(-7.282784, 112.7933981)
1/1 [=====] - 0s 14ms/step
(-7.2828117, 112.7933863)
1/1 [=====] - 0s 14ms/step
(-7.282783, 112.7933972)
1/1 [=====] - 0s 16ms/step
(-7.2828116, 112.7933863)
    
```

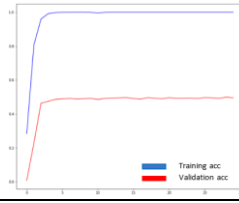
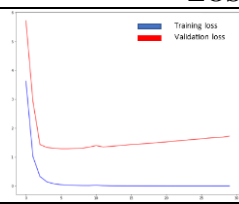
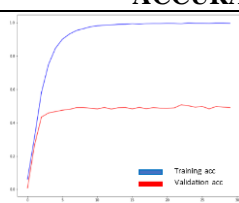
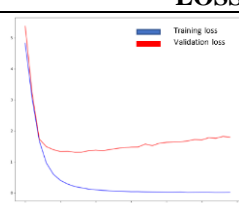
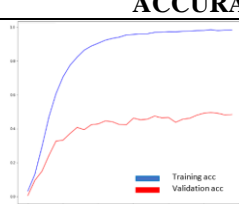

Fig. 3.13. Image Prediction Output Coordinates Results

III. RESULTS AND DISCUSSION

In this final project, several testing procedures were carried out to determine the reliability of the developed system. The reliability developed is the level of accuracy of the system predicting the coordinates when the omnidirectional camera is run which will capture every recorded frame. The location of the test is the location of the training, that are the area around the Mannarul Mosque, ITS Alumni Park, and ITS Rectorate. During the testing process, the model that was trained previously during the training process will be recalled so that the system can recognize each coordinate of the recorded image.

A. Testing using DNN based on architecture AlexNet

In this procedure, a dataset of 24000 images is used with a comparison of each for training and validation (80-20%).

VARIABLE	ACCURACY
-Without Drop Out	 <p>Max val_Acc = 49.92%</p> <p>Max Acc = 100%</p>
	 <p>Max val_Loss = 1.6884%</p> <p>Max Loss = 3.28 e-04%</p>
-Without Augmentation Data	 <p>Max val_Acc = 50.75%</p> <p>Max Acc = 99.36%</p>
	 <p>Max val_Loss = 1.6476%</p> <p>Max Loss = 0.0301%</p>
-With Drop Out	 <p>Max val_Acc = 49.68%</p> <p>Max Acc = 98.43%</p>
	 <p>Max val_Loss = 1.0161%</p> <p>Max Loss = 0.0719%</p>

The parameters used are as follows:

Table 1. Optimizer = Adam (learning rate = 0.001); Epochs = 30; Batch size = 64

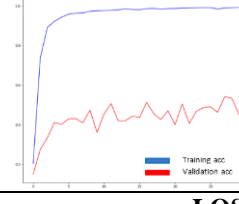
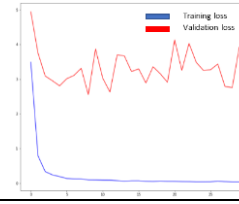
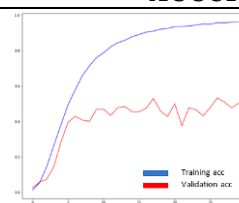
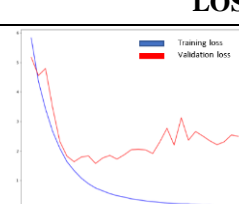
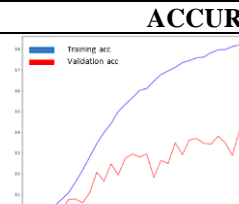
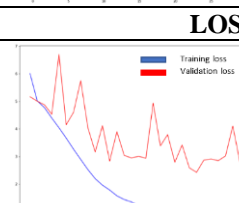
VARIABLE	ACCURACY
-Without Drop Out	 <p>Max val_Acc = 54.21%</p> <p>Mac Acc = 98.96%</p>
	 <p>Max val_Loss = 2.7896%</p> <p>Max Loss = 0.0458%</p>
-With Drop Out	 <p>Max val_Acc = 53.31%</p> <p>Max Acc = 95.80%</p>
	 <p>Max val_Loss = 2.2121%</p> <p>Max Loss = 0.135%</p>
-Without Augmentation Data	 <p>Max val_Acc = 40.54%</p> <p>Max Acc = 88.23%</p>
	 <p>Max val_Loss = 2.7823%</p> <p>Max Loss = 0.5202%</p>

Table 2. Optimizer = Adam (learning rate = 0.00001); Epochs = 30; Batch size = 64

B. Testing using DNN based on ResNet50 architecture

This procedure uses a dataset of 23868 images with a comparison of each for training and validation (80-20%). The parameters used are as follows:

Table 3. Optimizer = Adam (learning rate = 0.0001); Epochs = 110;

Batch size = 64

VARIABLE	ACCURACY	
-Without Drop Out		Max val_Acc = 38.31%
		Max Acc = 97.73%
-With Augmentation Data		Max val_Acc = 38.31%
		Max Acc = 97.73%
LOSS		
-Without Drop Out		Max val_Loss = 5.52%
		Max Loss = 0.0752%
-With Augmentation Data		Max val_Loss = 5.52%
		Max Loss = 0.0752%

Table 4. Optimizer = Adam (learning rate = 0.001); Epochs = 110; Batch size = 64

VARIABLE	ACCURACY	
-Without Drop Out		Max val_Acc = 19.92%
		Max Acc = 46.98%
-With Augmentation Data		Max val_Acc = 19.92%
		Max Acc = 46.98%
LOSS		
-Without Drop Out		Max val_Loss = 3.1176%
		Max Loss = 4.5033%
-With Augmentation Data		Max val_Loss = 3.1176%
		Max Loss = 4.5033%

C. Image Prediction Process Time Table

Table 5. Comparison of Prediction Period Between Models

MODEL	TIME
AlexNet	±15 ms
ResNet50	±20 ms

D. Prediction Output Localization Visualization Results

The results of the visualization of the prediction output from Fig. 4.1 were obtained with a validation accuracy level of 49.64% and a validation loss of 1.4899%. The total number of predicted classes is 200, which represent GPS coordinates. The data retrieval was done using a vehicle with a speed of about 20 km/h and an image acquisition speed of 30 fps. The error can be observed in the image, where there are unreadable or unknown road coordinates.



Fig. 4.1. Visualization of coordinate prediction output through Folium

E. Discussion

Based on the test results presented in Table 1, it is evident that the model (AlexNet) achieves a maximum validation accuracy of 54.21% with the following hyperparameters: Optimizer = Adam (learning rate = 0.001), Epochs = 30, Batch size = 64. Additionally, a validation accuracy of 50.75% is attained with the hyperparameters Optimizer = Adam (learning rate = 0.00001), Epochs = 30, Batch size = 64.

The graph depicting the variation of learning rates shows that a learning rate of 0.001 yields unstable accuracy per iteration compared to a learning rate of 0.00001. This instability arises from the impact of larger learning rates on gradient descent, resulting in greater weight renewal and a tendency towards finding local and global minimum points. Consequently, overfitting can occur. Since the first learning rate uses a value of 0.00001, it is observed that the maximum validation accuracy level is lower due to the longer weight renewal process. However, by increasing the number of epochs with a learning rate of 0.00001, it is possible to achieve better and more stable accuracy results, thereby preventing overfitting.

The next variation to be tested involves the utilization or exclusion of the Dropout layer, as well as the use or omission of data augmentation, to determine the highest accuracy level. Analyzing the graph in section 4.1, it is evident that the variations without data augmentation and Dropout layers, as well as those without data augmentation but with Dropout layers, exhibit high accuracy results. However, the graph indicates that the variations with data augmentation do not experience overfitting, while the other two variations show slight overfitting. This is attributed to the fact that data augmentation enhances the model's ability to recognize image patterns within a class by introducing a variety of class representations in the dataset.

Regarding the graphs with variations using Dropout, it is evident that the validation graph exhibits overfitting compared to those without Dropout. This suggests that the inclusion of Dropout in this study, which eliminates approximately 25-30% of neurons during the learning process in the Fully-Connected layer, hampers the model's

ability to recognize patterns in an image. It is possible that reducing the percentage of neurons eliminated could enhance the model's performance. Numerous studies have shown that Dropout can increase accuracy levels by disconnecting neurons that possess outlier features (undesirable characteristics).

IV. CONCLUSION

Based on the conducted project research thus far, it can be concluded that tackling the problem of Image Classification with a large number of classes, such as the 200 classes in this case, remains quite challenging. The maximum achievable accuracy using the AlexNet model ranges between 49-54%. The limitations in achieving higher accuracy levels may be attributed to the relatively small dataset size used in the research. Currently, the dataset consists of 24,000 images, with an 80% portion allocated for training data and a 20% portion for testing data. Expanding the dataset by collecting a larger number of diverse images could potentially improve the accuracy performance of the model.

Indeed, the selection of hyperparameters plays a crucial role in determining the final accuracy level of the model. Through the utilization of image augmentation techniques, overfitting can be mitigated. However, it is important to note that achieving higher accuracy results often requires increasing the number of epochs. This is because a more complex image pattern recognition process is necessary for the model to learn and generalize well from the augmented data. By allowing the model to go through more epochs, it has more opportunities to refine its understanding of the augmented images and improve its overall accuracy. Thus, finding the right balance between hyperparameters, such as the number of epochs, and employing image augmentation techniques can lead to better accuracy results in image classification tasks.

When employing the Dropout technique, overfitting is observed compared to not using Dropout. Additionally, a smaller learning rate prolongs the learning process and updates the weights gradually. However, using a small learning rate offers the advantage of preventing overfitting.

ACKNOWLEDGMENT

The authors gratefully acknowledge financial support from the Institut Teknologi Sepuluh Nopember for this work, under project scheme of the Publication Writing and IPR Incentive Program (PPHKI).

REFERENCES

- [1] Lin, Shufei, Ruiqi Cheng, Kaiwei Wang, and Kailun Yang. 2018. "Visual Localizer: Outdoor Localization Based on ConvNet Descriptor and Global Optimization for Visually Impaired Pedestrians" *Sensors* 18, no. 8: 2476.
- [2] Y. Yun, J. Jin, N. Kim, J. Yoon and C. Kim, "Outdoor localization with optical navigation sensor, IMU and GPS," 2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Hamburg, Germany, 2012, pp. 377-382, doi: 10.1109/MFI.2012.6343007.
- [3] D. Scaramuzza, "Omnidirectional Camera," 2014, doi: 10.5167/UZH-106115
- [4] H.-Y. Lin and C.-H. He, "Mobile Robot Self-Localization Using Omnidirectional Vision with Feature Matching from Real and Virtual Spaces," *Appl. Sci.*, vol. 11, no. 8, p. 3360, Apr. 2021, doi: 10.3390/app11083360.
- [5] J. A. Larcom and H. Liu, "Modeling and characterization of GPS spoofing", 2013 IEEE International Conference on Technologies for Homeland Security (HST), pp. 729-734, 2013.
- [6] A. Mulla, J. Baviskar, A. Baviskar and A. Bhovad, "GPS assisted Standard Positioning Service for navigation and tracking: Review *& implementation", 2015 International Conference on Pervasive Computing (ICPC), pp. 1-6, 2015.
- [7] Tianhao Bai, Bingqing Mei, Long Zhao, Xiaodong Wang, "Machine Learning-Assisted Wireless Power Transfer Based on Magnetic Resonance", *Access IEEE*, vol. 7, pp. 109454109459, 2019.
- [8] Jingjie Xin, Xin Li, Yongjun Zhang, Lu Zhang, Jianghua Wei, Shanguo Huang, "DNN based Multi-Faults Localization for 5G Coexisting Radio and Optical Wireless Networks", the Design of Reliable Communication Networks (DRCN) 2021 17th International Conference on, pp. 1-6, 2021.
- [9] Wei, J. (2020, September 25). AlexNet: The Architecture that Challenged CNNs. Medium. <https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951>
- [10] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25. <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>
- [11] An Overview of ResNet and its Variants | by Vincent Feng | Towards Data Science. (n.d.). Retrieved June 11, 2022, from <https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035>
- [12] Residual Networks (ResNet)—Deep Learning—GeeksforGeeks. (n.d.). Retrieved June 11, 2022, from <https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning>