OPEN ACCESS Freely available online

International Journal of Advancements in Technology

Research Article

Efficient Detection of Obstacle Objects Using AI

Ashwin Prabou*

Department of Information Technology, Vellore Institute of Technology, Tamilnadu, India

ABSTRACT

With the rise of automation and machine learning in our world, we have to work with a mindset to use it for our safety and for others' safety. This research paper aims to utilize two different machine learning models-K-nearest neighbors and neural networks to try to find what is the best for vehicles to detect any oncoming dogs on the road. Through building several different models and comparing their overall accuracy, this research paper will answer the questions about the differences between the two models, and which one would be the most reliable for vehicles. The accuracy of the models are measured through several tests taking in 32 × 32 px images of both roads and dogs, and training and testing the different models.

Keywords: K-nearest neighbors, Neural networks, Accuracy, Automation, Vehicles

INTRODUCTION

Nowadays we are seeing a surging trend with the use of automation technology in our daily lives: the cars we drive, the phones we use, and factory machines which create lots of our goods and services. Alongside this new wave of technology, safety of the consumers must also be a priority. And when safety is mentioned, it is usually guided towards humans-we are the consumers of these goods, after all. But one certain group of life that we don't tend to think about are our animals. Approximately 48 million American households own a dog, making it the most common pet in the US, but unfortunately, 1.2 million dogs are killed on the roads each year in the United States. In order to significantly reduce this quantity, we need to help train our AI to also recognize dogs when we want to keep them safe on the streets. To do so, we want to try several different modules and find which one will be the most effective in keeping our dogs safe [1-4].

The two models which are trained and tested in this research are the K-nearest neighbors and neural networks algorithms and models. But before we dive deeper into the differences of these models, we need to understand how the training process works: First, we set up our dataset (provided by Inspirit AI) of 1200 32 × 32 px images of dogs and roads, split into 600 each. When creating a simple machine learner, we want to split our images up into two parts; we select 20% of the images, at random, for the testing part, and 80% of the images, also at random, for the training. Although we can simply train our modules with images, we must also confirm that it can work as it is intended through tests-it is like training a student in a specific subject, and testing that student to measure its understanding on said subject (Figures 1 and 2).

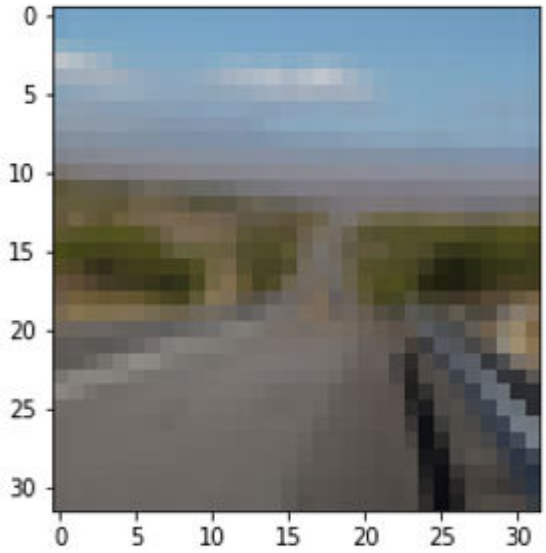


Figure 1: Label: Road.

Correspondence to: Ashwin Prabou, Department of Electrical Engineering, Vellore Institute of Technology, Tamilnadu, India; E-mail: ashwinprabou16@gmail.com

Received: 26-Oct-2022, Manuscript No. IJOAT-22-19827; **Editor assigned:** 28-Oct-2022, PreQC No. IJOAT-22-19827 (PQ); **Reviewed:** 11-Nov-2022, QC No. IJOAT-22-19827; **Revised:** 10-Jan-2023, Manuscript No. IJOAT-22-19827 (R); **Published:** 17-Jan-2023, DOI: 10.35248/0976-4860.23.14.230

Citation: Prabou A (2023) Efficient Detection of obstacle objects using AI. Int J Adv Technol. 14:230.

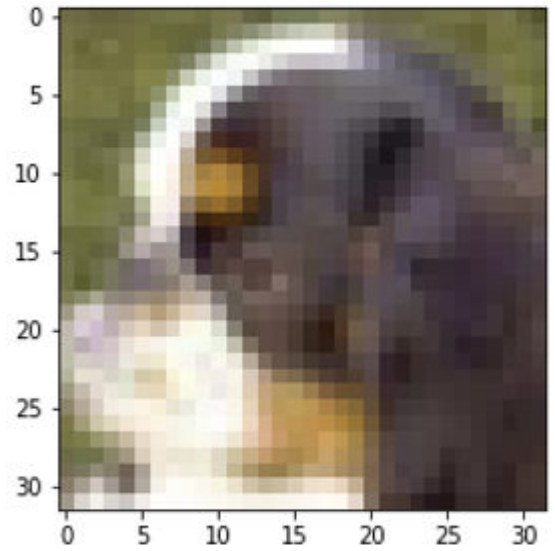


Figure 2: Label: Dog.

MATERIALS AND METHODS

We blur the images in our dataset because it is better for image processing; by smoothing edges and removing noise from an image, or random variation of brightness or color information in images. With blurry images, we can perform thresholding or edge detection [5-9].

K-nearest neighbors

The K-nearest neighbours algorithm is the first one we will be training and testing. The way a KNN algorithm works is through a proximity based learning- KNN uses proximity to make classifications KNN is currently being used in the medicine and agricultural fields, but is used a lot more in finance; forecasting the stock market, credit rating, and loan management (Figure 3).

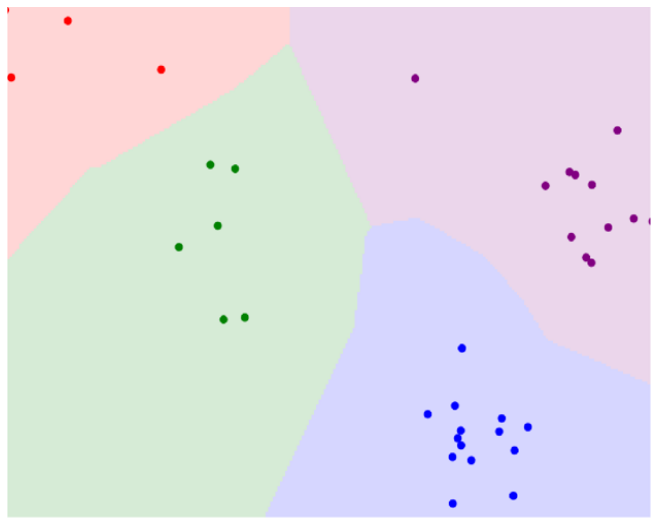


Figure 3: Demo of how KNN classification algorithm works. Each point in the plane is colored with the class that would be assigned to it using the K-nearest neighbors algorithm.

RESULTS AND DISCUSSION

When building this KNN model, you want to reach the highest percentage of accuracy, and one way you can play around with reaching the highest accuracy is through the variations of the number of neighbors. The number of neighbors, or the K value, is used to classify what group the test value belongs to (Figure 4).

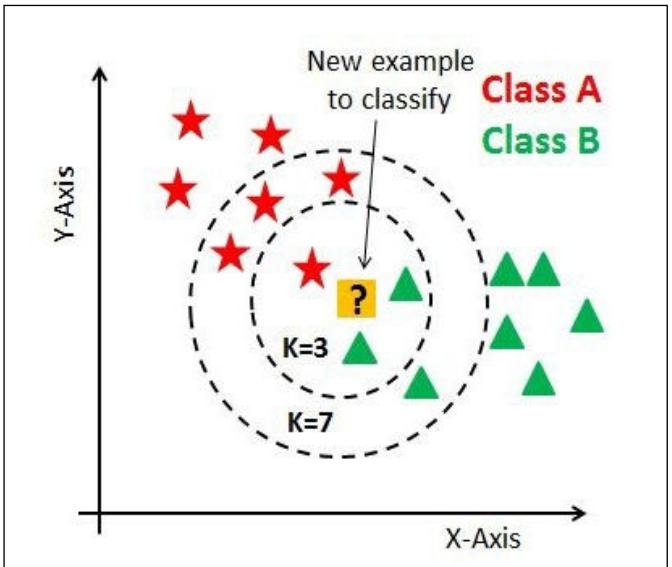


Figure 4: Classification of example.

If K=3, then the example classifies as class B since the number of neighbors which is greatest within the area are class B variables. If we were to use K=7, the number of neighbors which is greatest within the area would now be class A.

When writing and testing our models, I used Google's colaboratory system to proficiently write and execute Python in my browser. Before we run our model, we must select randomly 20% of the data for the test, and 80% for the train.

```
[ ] X_train, X_test, y_train, y_test = model_selection.train_test_split(data, labels, test_size=0.2)
```

Now it's time for the testing and training. For this part, I use the K Neighbors Classifier and train our model with fit, giving it the training inputs and labels. For this experiment, we will be testing various K values ranging from 1-10 and comparing the accuracies of each and concluding what would be the best value.

A conclusion we can make from KNN models is that with K values ranging from 1-10, the accuracy seen was above 82%, with the highest accuracy seen at 89% with a K-value of. When comparing images such as dogs and roads, a K-value will result in the highest accuracy, and is best for autonomous machines to help prevent more dogs from being run over.

A reason for an accuracy like such may be due to the nature of a KNN algorithm; they are very sensitive to small, irrelevant features which may throw the modules off and result in an accuracy like the one scored with the test. Because of such errors, we as humans must take on the responsibility of taking away these irrelevant features before moving on to modules.

Neural networks

Neural network is another technique for building a computer program that learns from data. And as you could tell by the name, it is a program which is loosely based on how the human brain works: It includes a collection of connected software "neurons" which helps convey messages to each other. Neural networks train their problem solving methods by attempting the problem over and over again, strengthening its connections that lead to success and diminishing those that lead to failure.

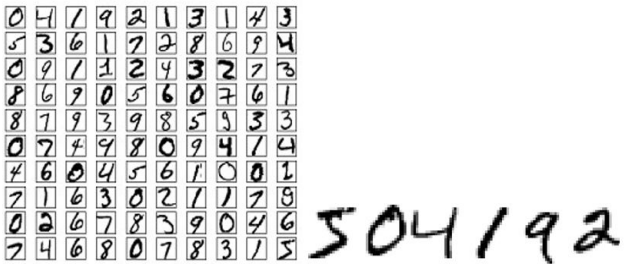


Figure 5: An example of how neural networks perform can be shown here.

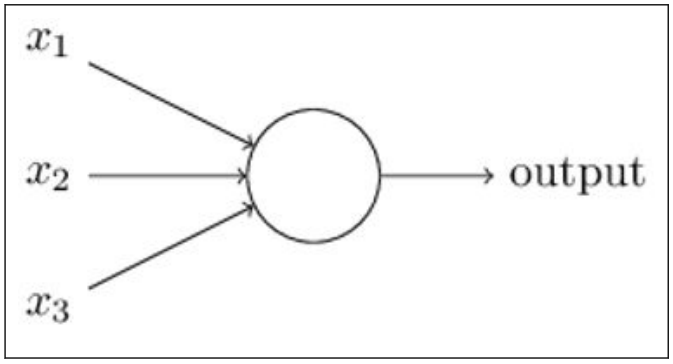
The image on the left represents the training data, while the image on the right represents the test. Neural networks repeatedly go over the handwritten numbers to understand them and achieve a high accuracy when it comes to solving the testing data.

The human brain can repeat the numbers presented on the right as we know the symbols for numbers and what differentiates a '4' from a '9', or a '1' from a '7'. In essence, our brain is like a supercomputer with over 100's of millions of neurons and 10's of billions of connections between the neurons. Computer programs have no prior knowledge, however, and depend on repetition of trial and error to learn.

When building our neural network program, I used MLP classifier from scikit-learn. MLP classifier makes neural network programs simple for our problem which we want to solve.

MLP, which stands for multi-layer perceptron classifier, consists of multiple layers and each layer is fully connected to the following one. The nodes of the layers are neurons using nonlinear activation functions, except for the nodes of the input layer. There can be one or more non-linear hidden layers between the input and the output layer.

To more clearly explain how neural networks and perceptions work together, we can use the image below:



A way you can think about the perceptron is that it's a device that makes decisions by weighing up evidence. In this case, x1, x2, x3 represent multiple factors which pass through a singular layer which then results in an output. The way that outputs are computed are with weights: Weights (w_1, w_2...) are real numbers which express the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum_{jwjxj}$ is less than or greater than some threshold value. This is a basic explanation as to how a perceptron works. In our study, we will pass our data of varied images through not just one, but multiple layers of multiple hidden-layer networks.

Using MLP classifier, we can easily create a neural network with 2 hidden layers, such as this one of 3 neurons and one of 4 neurons.

```
nnet = MLPClassifier(hidden_layer_sizes=(3, 4))
```

Unlike the previous experiment with KNNs, we also utilized 'for loops' to make it less repetitive, and just have the code run through the different parameters. For this experiment, we want to test our neural network with 10 different configurations; 5 with one hidden layer and 5 with two hidden layers: (1), (2), (2), (4), (5), (2,1), (4,2), (6,3), (8,4), (10,5). We kept the 1000 unchanged throughout the tests. After 3 separate runs of the

programs since results may vary, we take the average accuracy (in %) of the 10 configurations (Table 1).

Table 1: Results and average accuracy.

Neurons 1st layer	Neurons layer 2nd	Run 1 accuracy	Run 2 accuracy	Run 3 accuracy	Average accuracy
1	N/A	48.33333	48.33333	51.66667	49.44444
2	N/A	51.66667	51.66667	51.66667	51.66667
3	N/A	48.33333	48.33333	48.33333	48.33333
4	N/A	51.66667	51.66667	48.33333	50.55556
5	N/A	51.66667	51.66667	48.33333	50.55556
2	1	48.33333	51.66667	51.66667	50.55556
4	2	48.33333	51.66667	48.33333	49.44444
6	3	51.66667	51.66667	87.08333	63.47222
8	4	88.33333	51.66667	48.33333	62.77778
10	5	48.33333	51.66667	86.66667	62.22222

Based on our conclusions from the runs, the highest accuracy that we got was 63.47% with a neural network of 2 layers (6,3). However, based on this being the highest accuracy, we can conclude that neural networks may not work best when it comes to having a life at risk.

CONCLUSION

In conclusion, between the two types of models trained and tested, it would be more efficient in using a K-nearest-neighbors model. The KNN model test showed a high level of consistency along with a higher level of scores, compared to the neural networks, which seemed to show only 2 higher than average scores, but still proved to be highly inconsistent. These models still have their issues which include small factors in imaging which may throw these models off. The ideal solution to preventing further deaths of animals on the road may be to select a similar but better performing model to train and test with, such as a convolutional neural network. In the end, we were able to understand how these models tend to work in their unique ways and pick up knowledge on how to further improve these studies.

REFERENCES

1. Khosla P, Volpe R. Superquadric artificial potentials for obstacle avoidance and approach. In Proceedings. 1988 IEEE International conference on robotics and automation, Philadelphia, PA, USA. 1998;1778-1784.
2. Mostofi Y. Cooperative wireless-based obstacle/object mapping and see-through capabilities in robotic networks. IEEE trans mob comput. 2012;12(5):817-829.
3. Chapman CS, Goodale MA. Missing in action: The effect of obstacle position and size on avoidance while reaching. Exp Brain Res. 2008;191(1):83-97.
4. Yang J, Zhang B, Zhang H. The factorization method for reconstructing a penetrable obstacle with unknown buried objects. SIAM J Appl Math. 2013;73(2):617-635.
5. Lenser S, Veloso M. Visual sonar: Fast obstacle avoidance using monocular vision. In proceedings 2003 IEEE/RSJ international conference on intelligent robots and systems, 27-31 October, Las Vegas, NV, USA. 2003;886-891.
6. Lee J, Cho Y, Nam C, Park J, Kim C. Efficient obstacle rearrangement for object manipulation tasks in cluttered environments. In 2019 International conference on robotics and automation, 20-24 May, Montreal, QC, Canada. 2019;183-189.
7. Shrivastava AK, Verma A, Singh SP. Distance measurement of an object or obstacle by ultrasound sensors using P89C51RD2. Int J Comput Sci Eng. 2010;2(1):64-68.
8. Erdmann M, Lozano-Perez T. On multiple moving objects. Algorithmica. 1987;2(1):477-521.
9. Qu F, Yang J, Zhang B. Recovering an elastic obstacle containing embedded objects by the acoustic far-field measurements. Inverse Probl. 2017;34(1):015002.