Benedictine University

Machine Learning:

Vehicle Image Processing

Angel Meraz, Juan Cisneros

CMSC 4383

Harkirat Gill

05/08/2023

How to Run

This program was written using Google's colab notebook, so as a result we first must create two folders in the "content" folder that is called "data" and inside this folder we must create two other folders, "vehicles" and "nonvehicles". After this you must upload the images from the folder we provided into its respective colab folder, and then you can run the program. If it is run in another environment, we will just have to upload the "data" folder we provided and then just edit the path directories to where they are saved within the new environment.

Vehicle Image Processing

The overall objective of our project was to make a model that is able to analyze images and determine whether the image is of a vehicle or something else. Our project could be implemented in a couple of ways in the real world. The applications of our project is that it could be used for image detection, the main use of that would be for stuff like captcha. If you have a model trained to detect vague images such as vehicles, stop signs, or crosswalks you can use that model to get past captcha verification because it would be able to get around the inherent safety factor that captcha relies on, that being vague images that would be hard for some models to distinguish. However with our project we were able to train the model to learn off of vague image sets, thus making it so that it should be able to bypass a captcha verification. However image processing models have some issues that come along with them, Image training models require a big dataset and we do not have computers made for processing huge amounts of data. Github stores a max of 1000 documents in 1 folder so our database will be limited.

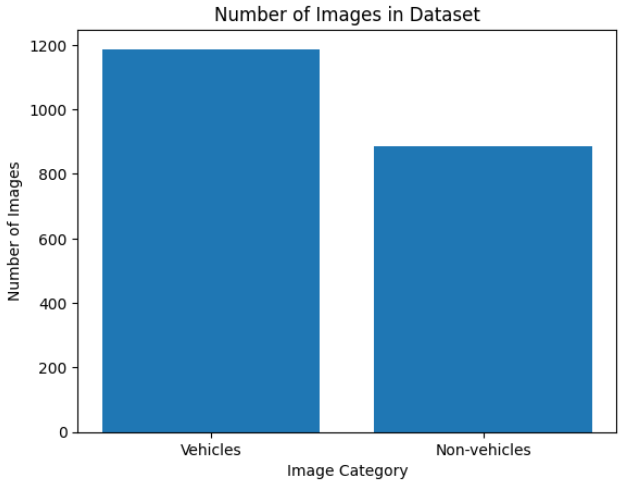
Any kind of variation in the image can cause the model to be less accurate since it might be looking for specific features that also leads to the model overfitting. In our model, the images that we used were very zoomed in on the vehicles which resulted in lower quality images, that made things harder for the model to determine if it was a vehicle or not because the amount of pixels/quality of image was very bad. In the past the types of datasets used for image processing tasks have been face datasets to distinguish people, a MIT dataset to learn what is indoors and what is not, scenery classification done by intel, and image datasets for medical purposes. There have also been tests done to see if image filtering had any effect on the efficiency of the model,

giving the model a specific trait and having it learn off that trait, and image segmentation to divide the image into portions in order to learn more accurately.

The state of the art method for image processing depends on the application and what needs to be analyzed, however Convolutional neural networks have been shown to be very effective in image processing. The reason that Convolutional neural networks have worked so well for image processing is because of its layers. The convolutional portion of the CNN goes through the image and extracts key features, The ReLu layer basically removes unimportant aspects of the image, the pooling layer decreases the image size and therefore increases processing/computational speed for the model, and the fully connected layer brings all of the previous layers together in order to have a classification. The other method that is used a lot is FixEfficientNet which combines previously used models FixRes and EfficientNet.The metrics used for model success are the accuracy, by using the ratio of success/total we can tell how successful the model was. Then there is also the Intersection over Union which measures the area of overlap/area of union.

After looking at those important aspects of image processing we went onto looking for our dataset and preparing to work with said dataset. The dataset we used was from a kaggle dataset that was for this kind of task, the dataset had about 10,000 images that were vehicles and 10,000 images that were not vehicles and those would be used to train a model accurately. However for us we did not have the benefit of being able to run all 20,000 images in 1 model so we limited it down to around 1,700 for our version, however in Github we were only able to upload 1,000 images for each (vehicle and non vehicle). However the dataset we chose did have a few issues. The amount of images we were given and the storage space that comes along with so many images was a challenge. Additionally, resolution was not the best so the model would

have some issues determining key features unlike if the images were in 4k quality. The dataset did not have a clear way for how it was collected but the way it was labeled was already divided into vehicles and non-vehicles which made it easier for us when inputting the data to the model. The key issue we could see with our dataset is that it is not too big of a dataset, compared to the task, it might end up underfitting and not being able to accurately predict the information.



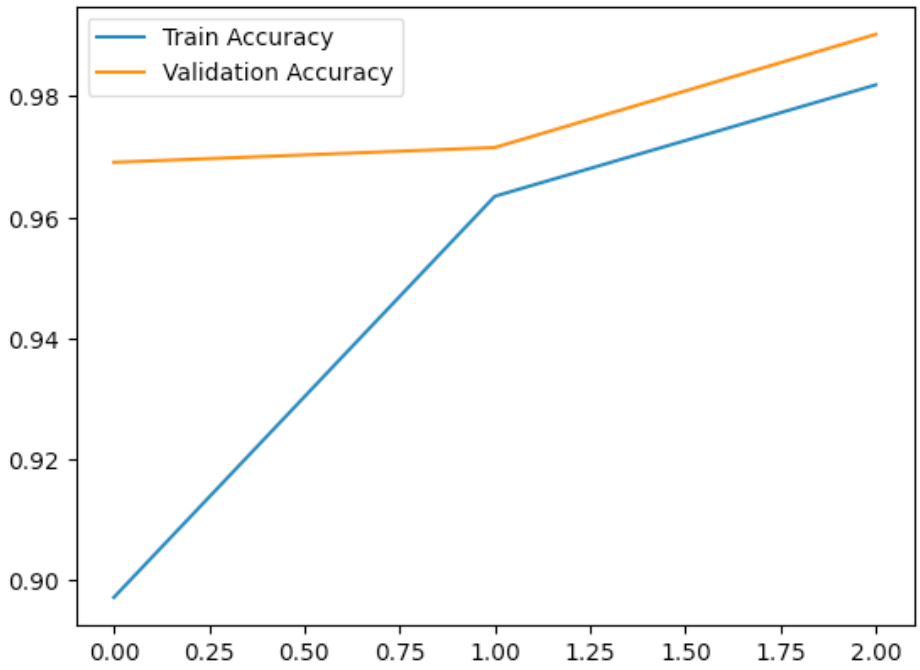
This statistics helps us determine what type of layers we are going to use, and how many in order to effectively teach the model. In order to clean our data, we didn't want to have duplicates to avoid false learning of the model, and we decided to filter out images that had low variance. We used three data transformations in our tests.The 3 data transformation methods we used were, Rescale which resizes all the images by a certain factor to reduce variance, horizonta_flip which flips the image horizontally and as a result increases the data to which the model can learn from, and zoom_range which randomly zooms into images at different scales to have increased variation. The purpose is to give the model more variations to learn from to have a better accuracy.

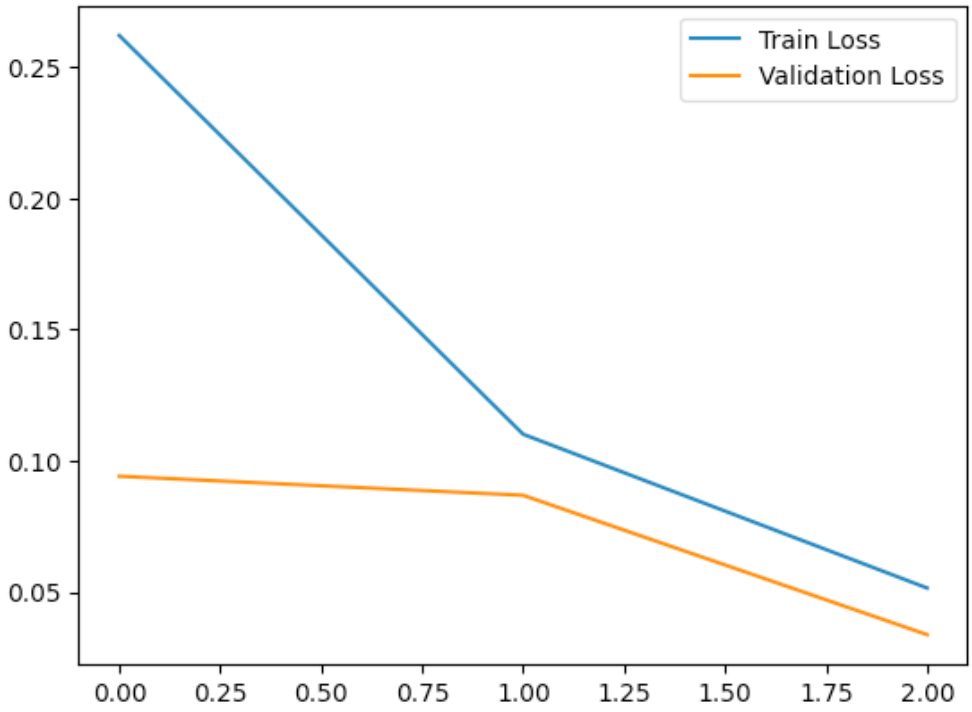
For our image classification program we used a convolutional neural network using the TensorFlow and Keras libraries. Three different variations were used in this overall program, of

which we chose the best one to use. This model is very popular in image classification, because it uses both image sets to distinguish between vehicles and non-vehicles and then tries to predict whether an image is a vehicle or not in the testing phase. The models are composed of different layers which each learns a different pattern of the data set that is presented to it and with the combination of each additional layer there is increased efficiency. In the training, the images are augmented using our data transformation to improve the ability to generalize new data. Some of the strengths of this model are that it can learn from raw images without needing manual feature engineering, can handle high dimensional images because of good spatial correlations within images. Some weaknesses include, they can require large amounts of memory to run, and can easily overfit if the model is too complex.

Some of our models overfit. When we started we began with each data set being 500 images. We then decided to add more images to counter the overfitting. We always kept the same hyperparameters, however when we saw that the validation accuracy was the same we decided to see what change deleting some hyperparameters did and it would perform even worse. Instead we tweaked our data sets.

These plots were used to determine the accuracy of our model:





These were the training times of our best model:

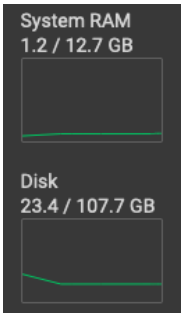
Epoch 1/3 102/102 [==============================] - 437s 4s/step -

loss: 0.2622 - accuracy: 0.8972 - val_loss: 0.0942 - val_accuracy: 0.9691

Epoch 2/3 102/102 [==============================] - 418s 4s/step - loss: 0.1102 - accuracy: 0.9635 - val_loss: 0.0869 - val_accuracy: 0.9715

Epoch 3/3 102/102 [==============================] - 416s 4s/step - loss: 0.0516 - accuracy: 0.9819 - val_loss: 0.0338 - val_accuracy: 0.9902

This was the memory usage of our best model:



The CNN model 1 was the one that outperformed all the other models. We've included these outputs in the above mentioned result plots.

After we ran the models we realized that the Convolutional Neural Network was the most efficient and gave the best result. We believe that the models that gave the best performance did so for a few reasons; a combination of having sufficient data, good architecture, and making sure the correct hyperparameters were used with the right amount of tuning was crucial to their performance. As for data transformation that gave the best improvement that would have to be Conv2d and MaxPooling. Given our previous research we did find that some of those hyperparameters would lead to high efficiency in Convolutional neural network models. If we were to continue working on the project we would continue tweaking the parameters and look into using Conv2d and Maxpooling further in order to have a more precise model. Given that our tests were done on a limited sized dataset we would have to run the model using the full 20,000 dataset in order to have a more precise model. In order to make it more interpretable there would need to be more implementation on user interface, one example could be visualization of what in

an image is the factors to which resulted in the model predicting whether or not it is a vehicle. Documenting the code even further to make sure the user understands exactly what goes on in each step would also help, and lastly, have the ability to reduce the number of layers so the user is not confused with so many layers and there is a better understanding as a result.

Works Cited

Guide, Step. "Images Classification and Object Detection Metrics." *Analytics Vidhya*, 21 June 2021, https://www.analyticsvidhya.com/blog/2021/06/evaluate-your-model-metrics-for-image-classification-and-detection/. Accessed 8 May 2023.

Hassan, Zubair. "Top 8 Machine Learning Image Classification Datasets." *Folio3.Ai*, 3 August 2022, https://www.folio3.ai/blog/ml-image-classification-datasets/. Accessed 8 May 2023.

Kundu, Rohit. "Image Processing: Techniques, Types, & Applications [2023]." *V7 Labs*, 3 August 2022, https://www.v7labs.com/blog/image-processing-guide. Accessed 8 May 2023.

Marius, Hucker. "State-Of-The-Art Image Classification Algorithm: FixEfficientNet-L2." *Towards Data Science*, 10 September 2020,

https://towardsdatascience.com/state-of-the-art-image-classification-algorithm-fixefficientnet-l2-98b93deeb04c. Accessed 8 May 2023.

"34 Amazing Image Datasets for Your Next CV Project." *Datagen*, https://datagen.tech/guides/image-datasets/image-datasets/. Accessed 8 May 2023.

"Vehicle Detection Image Set." *Kaggle*, https://www.kaggle.com/datasets/brsdincer/vehicle-detection-image-set. Accessed 8 May 2023.