Hong Kong Metropolitan University

COMP S492F

Machine Learning

Car Classification Project

Group

Chan Po Hong, 12303512

Leung Yiu Man, 11809630

Table of content

1. Description of the data preparation

The data preparation includes loading the image data from the directory using the *ImageDataGenerator* class from Keras. It rescales the pixel values from the range [0, 255] to [0, 1] by dividing each pixel value by 255. The *train_data* and *val_data* generators are created with rescaling, while the *test_datagen* generator is created with only rescaling. Those generators are used to create generator objects *train_generator*, *val_generator*, and *test_generator* respectively that enable model training and testing using the fit and evaluate functions of the Keras model.

For the training set and validation set:
The seed parameter is used to set the random seed for the data generator, which ensures that the data is shuffled consistently across different runs of the code. In this case, the seed value is set to 42, which is a commonly used value in machine learning to ensure reproducibility (the other is 123).

The interpolation parameter specifies the interpolation method used to resize the images to the target size. Here, the simplest and fastest interpolation method that simply picks the nearest pixel value by using nearest method here.

The shuffle parameter is used to randomly shuffle the order of the images in the dataset. It helps to avoid overfitting by ensuring that the model does not learn to recognize the images in a specific order.

The class_mode parameter specifies the type of label arrays that are returned by the generator. Here, the categorical mode is used, which returns one-hot encoded labels for each image. This is appropriate for multi-class classification problems where each image belongs to only one class.

2. Classification implementation

The classification implementation is using convolutional neural network (CNN) approach and using transfer learning with the VGG16 model, pre-trained on the ImageNet dataset, is used as the base model. It is effective in classifying images in the dataset by adding dense layers. The base model its pre-trained weights and which is frozen and not trainable. The top layer of the VGG16 model is removed, and the last layer is replaced with two dense layers with dropout and SoftMax activation. A Flatten layer to fatten the output from the base model. A Dense layer has 256 units with a ReLU activation function and a dropout rate of 0.5 to prevent overfitting, while the last dense layer has the same number of units as the number of classes in the dataset, which is determined by the len(train_generator.class_indices) function that will return 20 to output the predicted probabilities of each class.

The model is compiled with the Adam optimizer, categorical cross-entropy loss, and accuracy metric. The model then trained on the training data using the fit function with a defined number of epochs and early stopping callback function to stop the training when the validation loss stops decreasing. Finally, the model is saved to a file in the h5 format.

Hyperparameters with different experiments:
Epochs: 10, 30, 50, 100
Number of units in dense layer: 32, 64, 128, 256
Learning rate: 0.01, 0.001, 0.0001
Batch size: 32, 64

While using 256 number of units in dense layer, learning rate 0.0001, and batch size in 32 experiments, the larger epochs are not affect different result since the early stopping callback function will stop training after 24 epochs, and the validation loss stops decreasing. So, the epochs 50 and 100 experiment are get slightly different accuracies in the training.

3. Model testing details

The test_model.py script is used to test the trained model on a folder of images. The script accepts four arguments: model_path, image_folder_path, output_file_path, and target_size. The model_path argument is the file path of the trained model in the h5 format. The image_folder_path argument is the path to the folder containing the images to be tested. To test the model on images located in a folder should contain 20 directories that each directory represents a different class. The output_file_path argument is the path to the output text file that will contain the classification results. The target_size argument is the target size of the input image, which is set to 224 by default.
Here is the example in the terminal:

```
python test_model.py --model_path model/50_model.h5 --
image_folder_path DATA/test --output_file_path output/file.txt
```