VRDL_HW1 Report

0856054 莊紹平

Code

GitHub Link: https://github.com/Jhuangsp/VRDL_HW1 (https://github.com/Jhuangsp/VRDL_HW1)

- `main.py` : Train the model with ResNet50.
- `main_vgg.py` : Train the model with VGG16.
- `inference.py` : Inference the model.
- `crop_square.py` : Crop each image into square shape without resizing.
- `best` : Folder that contains the best situation of code(`main_vgg.py` , `inference.py`) & answer(`answer.csv`) on Kaggle.

Brief introduction

In this assignment, we have to build a deep learning model to classify the scene in the test images.
According to the Dataset provided by TA, there is a total of 2819 images for training and 1040 images for testing. All these images can be classified into 13 scenes:
(`bedroom` , `coast` , `forest` , `highway` , `insidecity` ,
`kitchen` , `livingroom` , `mountain` , `office` , `opencountry` ,
`street` , `suburb` , `tallbuilding`)
While training the model, I will split about 0.1 of training data as validation data (train:valid=2542:277), and choose the epoch which has the lowest `valid_loss` (or the highest `valid_acc`) as the result of this training.

Methodology

- **Data pre-process**
  I use `preprocess_input` function provided by Keras, which is to normalize pixel value from 0-255 to -1.0~+1.0. In this way, the weights can change more flexible (some weights decrease and others increase, rather than all weights decrease/increase together)
  I also tried to crop the images into a square shape, to avoid the distortion result from resizing function.

- **Data augmentation**
  I tried rotation, horizontal/vertical shift, shear, zoom, and horizontal flip. But according to the hyperparameter optimize method, only the horizontal flip will help the model classifying the images.

- **Model architecture (Transfer learning)**
  Because our dataset is quite small, comparing to other online datasets. It is hard to train a large model containing many parameters with a small amount of data.
  So I decide doing transfer learning using ResNet50 or VGG16 pre-trained by ImageNet as the base model and substitute the softmax classifier into my softmax classifier(used to classify into 13 classes).
  I tried 3 different softmax classifier:

  1. 1 Dense layer with output size 1024 with relu as activation
  2. 2 Dense layers with output size 1024 & 512 with relu as activations
  3. 2 Dense layers with output size 512 & 256 with relu as activations
     (Before every Dense layer, I also add Dropout layer with the same drop_rate)

  In order to avoid the overfitting, I tested the input size 256*256 & 224*224 & 200*200.
  After experience & hyperparameter optimize method, it shows that using VGG16 with the second softmax classifier and setting drop_rate to 0.25 will have the best result.

- **Hyperparameters & Hyperparameters Optimize**
  In this assignment, I use Adam to optimize the model weight due to the ease to use. The loss will calculated by categorical_crossentropy.

Fixed Hyperparameters:

- learning rate: 1e-4
- epoch: 20
- batch size: 20

Although, I train the model for 20 epochs, but I only keep the result of lowest `valid_loss` (or the highest `valid_acc`) epoch.

Besides the fixed hyperparameters, I also let some hyperparameters to be optimized by `hyperopt` TPE method. (Because of the model size of VGG is too large, I only find hyperparameters on ResNet and use the best value to train VGG)

Hyperparameters to be optimized:

- rotation_range: `[0, 30]`
- width_shift_range: `[0.0, 0.3]`
- height_shift_range: `[0.0, 0.3]`
- shear_range: `[0.0, 0.3]`
- zoom_range: `[0.0, 0.5]`
- fill_mode: `reflect`, `wrap`, `nearest`, `constant`
- horizontal_flip: `True` or `False`
- drop_rate: `[0.1, 0.3]`

Summary

Best model hyperparameters:

- Input size: 224*224
- Base model: VGG16
- Classifier: 2 Dense layers with output size 1024 & 512 with relu as activations
- Optmizer: Adam
- Learning rate: `1e-4`
- Epoch: 20
- Batch size: 20
- horizontal_flip: `True`
- drop_rate: `0.25`

Why cropping the images into a square shape doesn't work?

In this task, if we just crop the center of image, it might change the class that the data bellong to. For example, the center of `mountain` data might change into `forest`; the center of `insidecity` data might change into `street`.
So for this task cropping the images will not help the classifier.

Why most augmentation doesn't work?

In this task, our data is a large scene store into a small image, even the augmentation just slightly distort the image, the whole scene might change a lot.

Why VGG performs better than ResNet?

In most of task the ResNet architecture works better than the VGG architecture, but in this task is not. (VGG:0.91826/ResNet:0.90272)
But in this task, I found that ResNet sometimes answer differently with VGG on some ambiguous data.
For example, for this image, VGG says this is `suburb` while ResNet says this is `mountain`.

And for this image, VGG says this is `street` while ResNet says this is `insidecity`.

So, I think both of the answers of ResNet and VGG are make sense, but the VGG think more closly to the labeling person in these ambiguous situation.

Future work

In the future, I can try more model architecture (i.e. TripletNet) and find a more proper augmentation method.