Implementing Xception and Yolov5 transfer learning

1st Yao An Lee
Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611 USA
danielyaoanlee@gmail.com

***Abstract*—Over the years, Machine Learning(ML) has been widely used in several industries and usually need to deal with large data set. This study proposes two different neural network models over flower species dataset and car detection dataset to experiment their performance. In my study, I introduce Xception [1] and Yolov5 [2] and implement transfer learning based on these models. Xception is introduced to classify flower species and Yolov5 is introduced to detect car images. The metric to compare model performance is determined by the accuracy of classification, the classification report, confusion matrix, precision value and precision-recall curve. All models are tuned using hyperparameter tuning techniques to maximize performance of each model.**

***Index Terms*—Machine Learning, Artificial Neural Network, Xception, Yolov5, Transfer Learning.**

I. Introduction

Machine learning has developed into a widely used framework for approximating statistical and algebraic relations over a given set of data for decades. Since 1986, the introduction of backpropogation [3] has allowed people to be able to train MLP (Multi-Layer Perceptrons) and thus lead to the development of deep neural network and convolution neural network. With the combination of different complex architectures, Machine Learning is capable of solving difficult prediction and classification problems. Another huge advantage of artificial neural network is to recognize images and detect objects, in which its application has benefit human in multiple places.

A. Data

Two dataset are used in this study. One is flower species dataset (Fig.1) and another is car detection dataset (Fig.2). Flower species dataset contains 1678 images from 10 classes and each RBG image is of size 300×300×3. Car detection dataset contains 1001 training images with labeled annotations for 559 training samples and 175 test images without any label. These car images are collected from dash cam video and thus several images have no car in it.

B. Xception

Xception [1], stands for Extreme version of Inception, is an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operationis. They both use depthwise separable convolution because it does not need to perform convolution across all channels, which makes the number of connections fewer and the model lighter. Compared with Inception-v3(also by Google), which use original depthwise separable convolution, Xception has modified depthwise separable convolution and change the order of operations to make pointwise convolution followed by a depthwise convolution with residual connections and eliminate intermediate activation function to increase accuracy and mean average precision. These modification make it even better than Inception-v3 for both ImageNet ILSVRC and JFT datasets.

Fig. 1. Flower species

Fig. 2. Car detection

C. Yolov5

YOLO [2] an acronym for 'You only look once', is an object detection algorithm trained on the COCO dataset that divides images into a grid system, and includes simple functionality for Test Time Augmentation (TTA), model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite. The reason I select yolov5 medium (Fig.3) is because it does almost the same performance compared to yolov4 but with much lower computational time. Moreover, compared with the newest yolov7 and the largest yolov5 x, although it does not perform as good as them, yolov5 medium has smaller number of parameters, higher computational efficiency and only slightly lower performance which also outperform EfficientDet and several RCNN.

D. Transfer Learning

It is very computational expensive to train a very large deep neural network architecture from scratch. Specially if you have limited computational resources and/or a small training dataset. As a result, by using transfer learning, we should find an existing neural network that accomplishes a similar task to

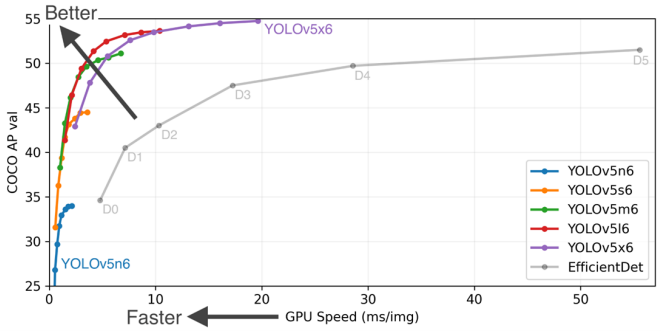

Fig. 3. Yolov5

the one you are trying to tackle and reuse the lower layers of this network. It will not only speed up training considerably but will also require much less training data.

II. Experiment

A. Data preprocessing

For flower species classification, since the data is in pixel value between 0 to 255, all I need to do is divide it by 255 to normalize it and transfer it to constant tensor as the preparation for transfer learning.

As for car detection, since I'm using yolov5, several data preprocessing techniques need to be implemented:

- Transfering the bounding box format from minimum and maximum of x,y-axis to center of x,y-axis, width and height and normalize it.
- Create a new dataframe to include the unlabeled training images and assign the class number.
- Combine the duplicate image with several bounding box labels, rename training images, create label text file and split training data into train and valid.
- Rename test images and create my own test label from MakeSenseAI.
- Create a yaml file to direct my train, valid, and test dataset.

B. Hyperparameter tuning

For flower species classification, I first compare between pooling and flatten as output layer (Fig.4), then compare between different optimizers (SGD, Adam, Nadam, RMSprop), last compare between different batch size (online learning/ mini-batch learning). The best hyperparameters are pooling layer as output layer with Nadam as optimizer and batch size 1.

For car detection, I first compare between different optimizers (SGD, Adam, AdamW) (Fig.5), then compare between different batch size (online learning/ mini-batch learning) (Fig.6). The best hyperparameters are optimizer AdamW and batch size 1. The reason why I didn't select more hyperparameters is because yolov5 was initially trained on several objects including cars, there is no need for me to change too many combination.

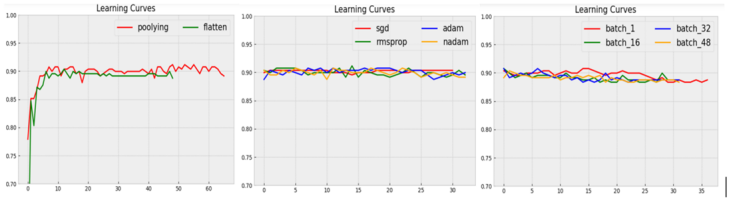

Fig. 4. Learning curve of pooling and flatten, optimizers, batch sizes

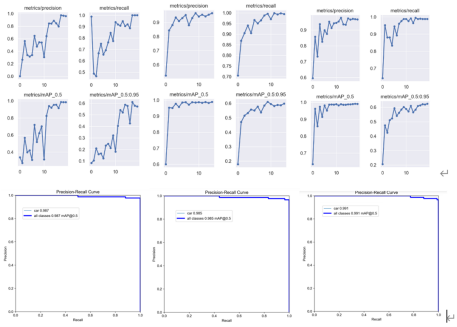

Fig. 5. Learning curve between different optimizers

C. Xception

Xception is already a built-in application in Keras library. By using transfer learning, I only need to create a resizing input layer as (300, 300, 3) on top and specify pooling and output layer.

D. Yolov5

Yolov5 has its built-in functions that we can make use of. By implementing the train.py function, we can direct to our own data directory and specify different hyperparameters we are interested in tuning. After completing the training, train.py function will automatically create a best.py function in our own directory. We can then implement val.py or detect.py function to validate the performance of test dataset or predict the results. When using val.py and detect.py function, we have to specify the best.py function we just trained and store the results to our own directory.

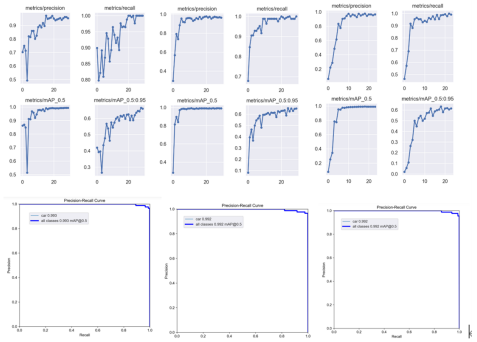

Fig. 6. Learning curve between different batch size

TABLE I
PERFORMANCE FOR FLOWER SPECIES CLASSIFICATION

dataset	Train	Validation	Test
Accuracy	100 percent	90 percent	88 percent
precision	1	N/A	0.88
recall	1	N/A	0.88
f1-score	1	N/A	0.88

E. Validation for unlabeld test set

Since I'm using yolov5, I decide to make use of its built-in validation function to validate the performance between the prediction and my own labels. This built-in function provide mAP when IoU (Intersection over union) in thresholds (0.5, 0.5 0.95), precision, recall, f1-score and precision-recall curve.

As for the images without a car, the way yolov5 evaluate detection performance is to compare the percision and recall to form precision-recall curve. Because yolov5 will return nothing when no car exists in the image, if I set fix target label ([0,0,0,0]) for all images without a car within it, the False Negative will increase and thus decrease recall. As a result, I leave the target blank for images without a car within it.

Last, for region of interest, where the exact bounding box location is not particularly the target. Yolov5 introduce the concept of IoU (Intersection over union) and select different thresholds (0.5, 0.5 0.95) to defined the correctness of detection.

III. RESULTS AND DISCUSSION

A. Flower species classification

By using transfer learning with Xception, we get 100 percent accuracy in training dataset and about 90 percent accuracy for validation set (Table.1). As for precision, recall and f1-score, we get all 1 for training dataset and 0.88 for validation and test set. In confusion matrix (Fig.7), only a few samples are misclassified in test set. we can see It's a little overfit but the accuracy 90 percent and 0.88 in precision, recall, f1-score is already a good classification.

```
[41,  0,  1,  0,  0,  2,  1,  1,  0,  2]
[ 0, 42,  1,  0,  0,  0,  0,  1,  0,  0]
[ 1,  1, 35,  0,  2,  0,  2,  0,  0,  5]
[ 0,  0,  0, 33,  0,  1,  0,  0,  2,  0]
[ 2,  0,  5,  0, 36,  0,  0,  0,  0,  2]
[ 1,  0,  0,  0,  0, 38,  0,  0,  1,  0]
[ 1,  1,  0,  0,  0,  0, 41,  0,  0,  0]
[ 0,  1,  0,  0,  0,  0,  1, 35,  0,  0]
[ 0,  0,  0,  0,  0,  0,  0,  1, 29,  2]
[ 3,  0,  2,  0,  1,  1,  5,  0,  0, 32]
```

Fig. 7. Confusion Matrix

B. Car detection

After validating the test set with trained best.py function, results get precision 0.961, recall 0.884, mAP0.5 = 0.943 and mAP0.5-0.95 = 0.54 with precision-recall curve in Fig.8. Moreover, by implementing the detect.py function, the following predicitons will be shown (Fig.9). Despite that performance is a little lower than training data, we can still conlcude that when IoU with threshold = 0.5, performance is good and mAP can be over 0.94. The reason why I select IoU threshold to 0.5 is because every test label is create by MakeSenseAI and thus all bounding box are influenced by mistakes human may encounter.

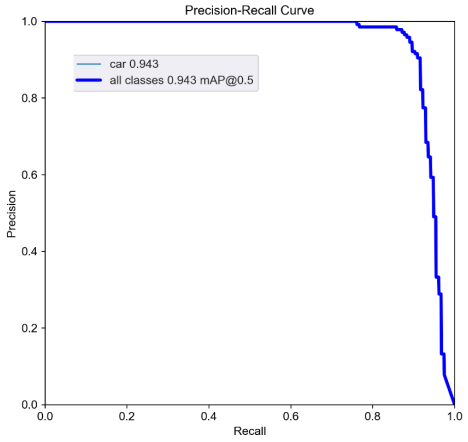

Fig. 8. Precision-Recall curve

Fig. 9. Bounding box predicitons

ACKNOWLEDGMENT

The authors would like to thank Dr. Catia Silva for providing this opportunity to experiment Artificial Neural Network implementation.

REFERENCES

[1] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

[2] https://github.com/ultralytics/yolov5

[3] Rumelhart, D., Hinton, G. Williams, R. Learning representations by back-propagating errors