DeepLab car part segmentation – mobile solution

This DeepLab model is based on Mobilenetv2 because it aims at deploying the trained model on mobile devices. The following sections will talk about data preparation, training model and deploying the trained model on Android phones.

# Source code preparation

DeepLab is developing in tensorflow/models branch, I have cloned this branch to modify the code locally:

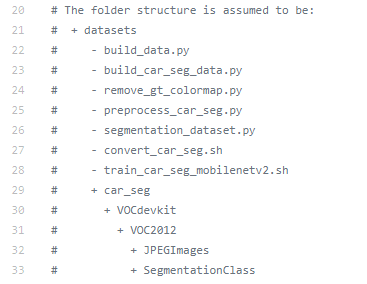
$ git clone <https://github.com/tensorflow/models.git>

After cloned the source code to local machine, the DeepLab source code lies in this path:

/models/research/deeplab

# Data preparation

DeepLab is originally trained and tested on PASCAL VOC dataset so I have preprocessed car dataset basing on that dataset. In order to preprocess data, I have written a [script](https://github.com/dohai90/models/blob/master/research/deeplab/datasets/convert_car_seg.sh) named “convert\_car\_seg.sh” to do this work. Assuming that the structure of necessary codes in “datasets” folder is as follows:



This script will create all required folders for you as the structure above, and it also calls 4 other important python3 scripts, including:

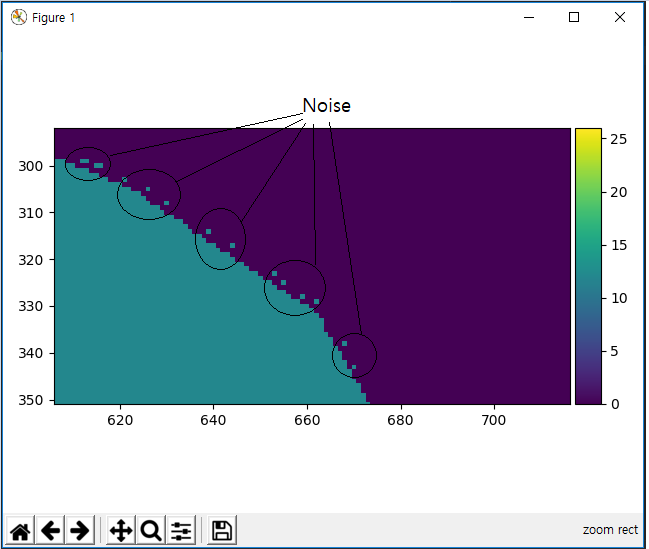
“create\_car\_seg\_datapath.py”: this script will list all background images into “backgrounds.txt” file and ground truth images into “segmentations.txt” file based on root dataset folder. For example, I have set my dataset root folder on line 54 of “convert\_car\_seg.sh” script:



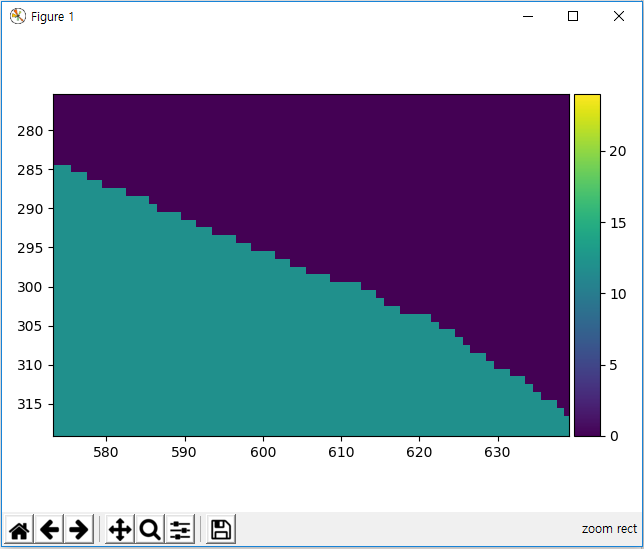
In “car\_data”, there are many pairs of background and ground truth data, background is suffixed with “\_0001\_Backgound.jpg” and ground truth is suffixed with “\_0000\_Layer 1.jpg”.

“preprocess\_car\_seg.py”: this script will read line by line in “backgrounds.txt” and “segmentations.txt” files to:

* Resizing all input images and their corresponding ground truth into max size of 769 pixels. Then save all of them into JPEGImages for background “jpeg” files and SegmentationClass for ground truth “png” file.
* Denoising ground truth images, because the ground truth is saved in color map style, however, DeepLab needs label images including category values of each class for each pixel, so I need to transform original “jpeg” ground truth into “png” ground truth. Each “png” ground truth is formatted into palette and index corresponding to our pre-defined classes. However, after transformation, the noise occurs for each ground truth as depicted as following figure:



That is a kind of “salt and pepper noise” so I have used opencv to remove those noise. The result after applying “salt and pepper denoise” is as follow:



* Splitting all background images and their ground truth images into 5 folds, I am using 4 folds to train and 1 fold to evaluate the result. The train/val/trainval files stores list of image files.

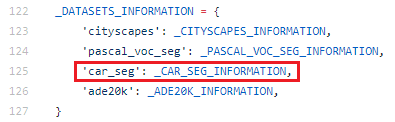
“remove\_gt\_colormap.py”: this script is written by DeepLab authors to remove the color map in ground truth annotations.

“build\_car\_seg\_data.py”: this script is based on “build\_voc2012\_data.py” script to convert background images and their annotations into TFRecords format files.

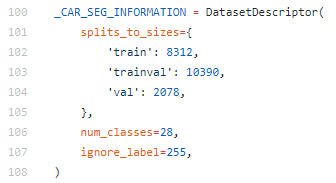
That’s all the steps to preprocess the dataset.

One more thing, in order to train/test the model, I need to modify “segmentation\_dataset.py” script in this “datasets” folder as follows:

* Create our dataset name “car\_seg”:



* And the information of our data:



The number of train/val/trainval is corresponding to the number of images are listed in “train/val/trainval.txt” files created from previous step. Our dataset has 28 categories so “num\_classes” is set to 28. In our dataset, I don’t care to labels outside the range [0, 28) so I set ignore\_label=255 (or any number outsize the range [0, 28)).

# Training, evaluating and testing DeepLab model:

## Training

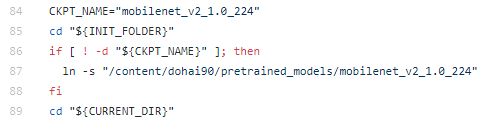
I based on “local\_test\_mobilenetv2.sh” script to write a script to train DeepLab model on our dataset, put it on “scripts” folder and named the script “train\_car\_seg\_mobilenetv2.sh”. In order to run this script, from “models/research/deeplab” directory, execute the command on one terminal:

$ sh ./scripts/train\_car\_seg\_mobilenetv2.sh

This script will create all necessary folders to stores checkpoints, evaluation results and exported model. I have used 2 NVIDIA 1080 to train the model, so I specify them in code:



Because mobilenetv2 is the backbone model so I used “mobilenet\_v2\_1.0\_224” pre-trained on ILSVRC dataset or “mobilenetv2\_coco\_voc\_trainval” pre-trained on coco and PASCAL VOC dataset. You can download the former from [here](https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet) and the latter from [here](https://github.com/dohai90/models/blob/master/research/deeplab/g3doc/model_zoo.md). You can select the pre-trained model what you want by modifying these lines:



The CKPT\_NAME variable stores the model name you want for fine tuning. However, in my experiment, the pre-trained “mobilenetv2\_coco\_voc\_trainval” model results in higher mIoU than the other. I recommend you fine-tune from “mobilenetv2\_coco-voc-trainval”.

These following lines specify the input arguments for “train.py” script:

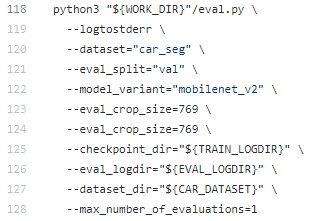


I will explain some input arguments, the others are self-explained.

* The “num\_clones” is the number of GPUs I am training the model, I train the model on 2 GPUs so I set “num\_clones=2” in this case.
* The “dataset” field is the dataset name I have set on step that I modified the “segmentation\_dataset.py” script.
* The “train\_split” is the dataset that you want to train the model, it can be “train” or “trainval”.
* The “model\_variant” is the backbone model, it can be “mobilenet\_v2” in this case or “xception\_65”.
* The “fine\_tune\_batch\_norm” is the key parameter for training the model, if you want to gain a high accuracy, you should set it to “true” as in this case, however, if you set it to “true”, you should set the “train\_batch\_size” parameter as large as possible (recommend to set it larger than 12)
* The “initialize\_last\_layer” should be set to false if you fine tune your model from the model pre-trained on PASCAL VOC dataset because it has only 21 classes, and the number of your categories may be different from PASCAL VOC dataset, that’s why you should set this parameter to false.
* The “train\_crop\_size” is the size of input images to be cropped before feeding into the network. This parameter is calculated as follow: train\_crop\_size = k \* output\_stride + 1. Where k is an integer.

## Evaluating:

A snippet of code to evaluate the model once after the model is trained as the following:



The meaning of the parameters:

* The “dataset” parameter is the name of the data that you are working on.
* The “eval\_split” is the set of data that you want to evaluate.
* The “model\_variant” is the same as training.
* The “eval\_crop\_size” is the image size that you want to evaluate, it should be greater than or equal to the max size of images in validation set, if not the code will be crashed. And it is calculated like “train\_crop\_size”.
* The “max\_number\_of\_evaluation” is set to 1 because I just evaluate the model once after training it.

I wrote 2 separate scripts named “evaluate\_mobilenetv2\_trainset.sh” and “evaluate\_mobilenetv2\_valset.sh” to evaluate the training model continuously. The former evaluates on “train” set and the latter evaluates on “val” set. Open 2 new terminals, on each terminal, execute the command from models/research/deeplab directory like below:

Terminal 1: $ sh ./scripts/evaluate\_mobilenetv2\_valset.sh

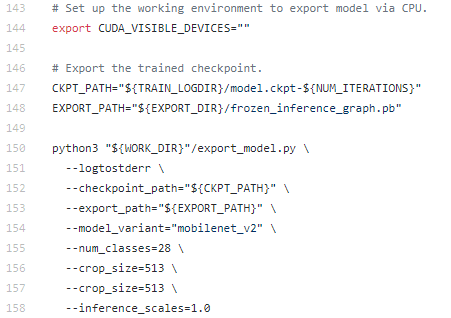
Terminal 2: $ sh ./scripts/evaluate\_mobilenetv2\_trainset.sh

Take a look at these script, I set:

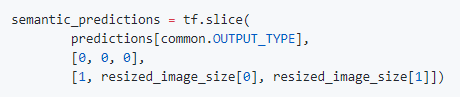
* The “eval\_interval\_secs” is the interval time between 2 evaluations in seconds.
* The “max\_number\_evaluations” is set to “-1” in order to evaluate the model continuously.

## Exporting model:

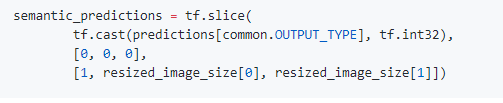
In order to export model that can be deployed on Android phone, there are 2 things you need to consider. The first thing, you must export the model by CPU because the exported model will be deployed on CPU of mobile phone, if you export the model by GPU, that model cannot be loaded onto mobile CPU. That’s why I set export CUDA\_VISIBLE\_dEVICES=”” before calling the “export\_model.py” script in “train\_car\_seg\_mobilenetv2.sh”.



The second thing, I have modified line 131 in “export\_model.py” as recommendation from [this](https://github.com/dailystudio/ml/tree/master/deeplab). From



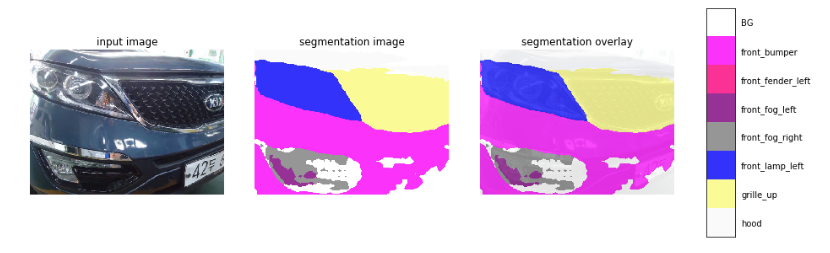
To



Changing this line is casting the INT64 Slice Operation to INT32 Slice Operation. INT64 Slice Operation is NOT supported on Android currently and it will cause a runtime exception during inference.

## Testing

In order to make prediction on trained model, I wrote a ipython notebook, you can find it in “models/research/deeplab/notebook/deeplab\_for\_car\_part\_segmentation.ipynb”. This notebook will create a colormap from labels, create a model object, load the trained model file, make prediction and display the result, all codes are self-explained via comments in the notebook. The result after predicting is illustrated as the below figure, you can try it yourself by your own images.

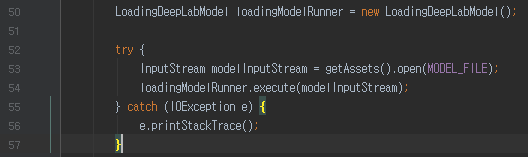


# Android application.

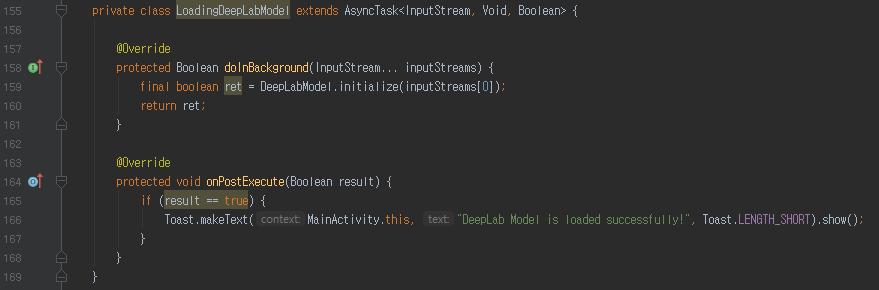
Based on [this](https://github.com/dailystudio/ml/tree/master/deeplab), I re-write the code for easier understanding. There are 3 main files in my source code

## MainActivity.java

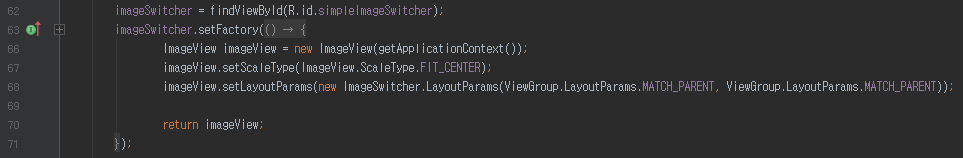
The activity file to process all Graphic User Interface. Because this is UI thread, I need to execute a long time operation like loading model and prediction on background thread. First in onCreate function, I load the model by calling an AsyncTask.



The LoadingDeepLabModel class inherits from AsyncTask class, this class will initialize the Deeplab model from background and feedback to UI Thread when it’s done.



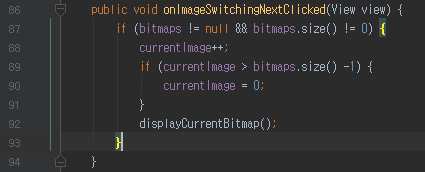
On the MainActivity, I place an ImageSwitcher to hold an ImageView and this ImageView will display images.



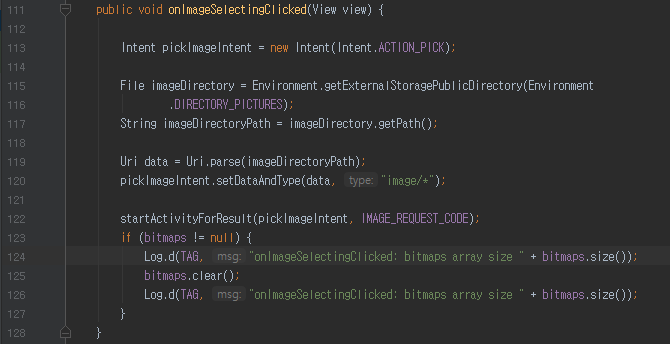
One button to switch image back:



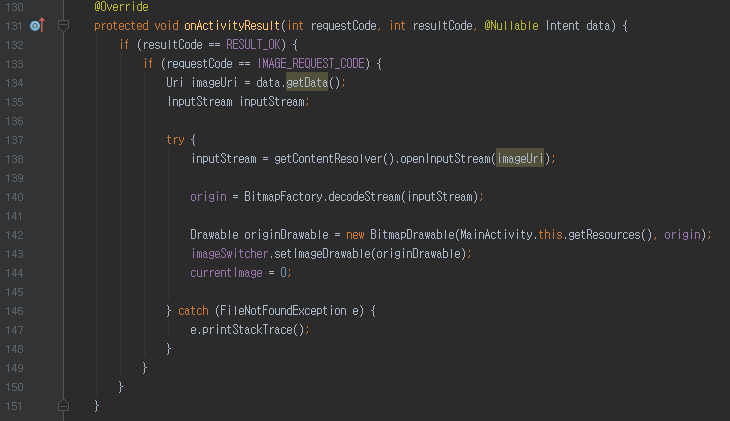
One button to switch image next:



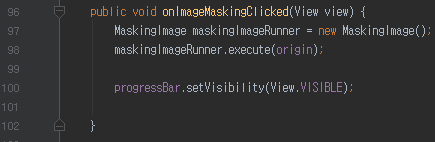
One button to load an image



I call “startActivityForResult()” function to pick up an image from internal storage. This function will call “onActivityResult()” function, and we get a returned image.



One button to execute masking operation

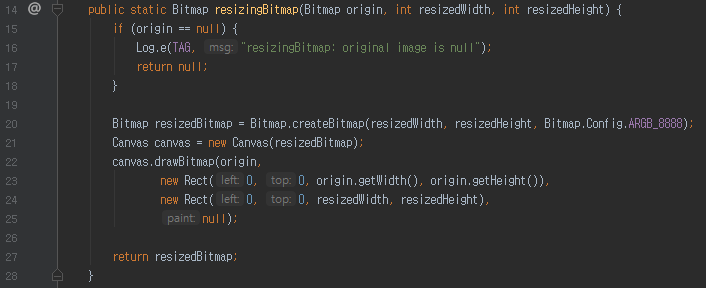


Masking operation is also a long time operation, so it must be executed in a background thread rather than the UI thread. MaskingImage is a subclass of AsyncTask class, it infers the mask and returns the result to the UI thread.

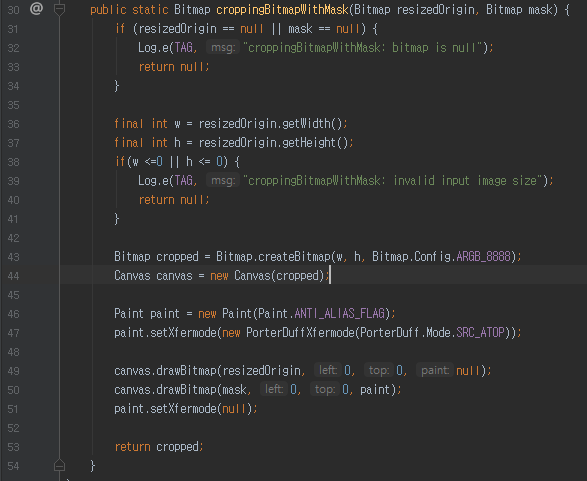


## ImageUtitls.java

This is a utility to operate on bitmaps. This class includes 2 static functions, they are resizingBitmap() to resize the bitmap

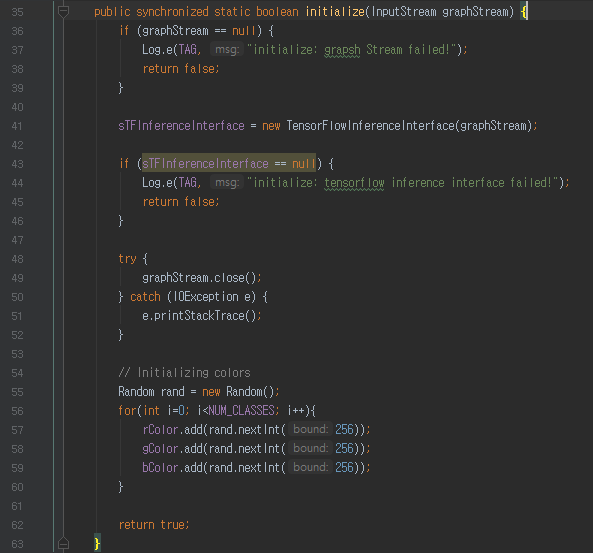


and croppingBitmapWithMash() to overlay mask over the original image.



## DeepLabModel.java

This is a core class. Its responsibility are first, initialize the model via initialize() function:

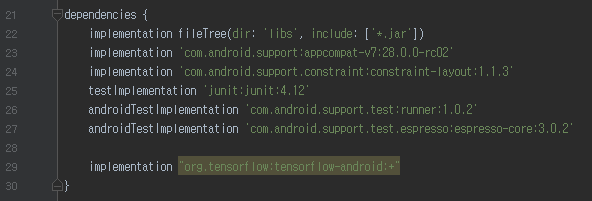


The second one is to predict car part



## Tensorflow for mobile

In order to use tensorflow on Android mobile, we need to add one line to build.gradle (Module:app) file like below:



Build.gradle will download and install all dependencies and required libraries you need.