

# Solution for Stanford-cars classification

DungNB

## I. SUMMARY

This solution for challenge <https://www.aiforsea.com/computer-vision>.

In this challenge, I have to build a cars classifier using the Stanford cars dataset, which contains 196 classes (including make and model). This is a normal image classification, but it is not easy to achieve high accuracy in the short time of challenge.

Key idea:

- Build strong single model on training set:
  - o Different CNN architectures
  - o Different image input sizes
  - o Train with different augmentation schedules
  - o 12 crops for predicting
- Ensemble different models

My final accuracy is 0.9462, higher accuracy than state-of-the-art stanford cars 2018 [1] (0.945) and nearly state-of-the-art image classification on stanford cars 2019 (0.947) [2]

## II. DATASET

- 196 classes
- Trainset: 8144 images
- Testset: 8041 images

Some images in training set:

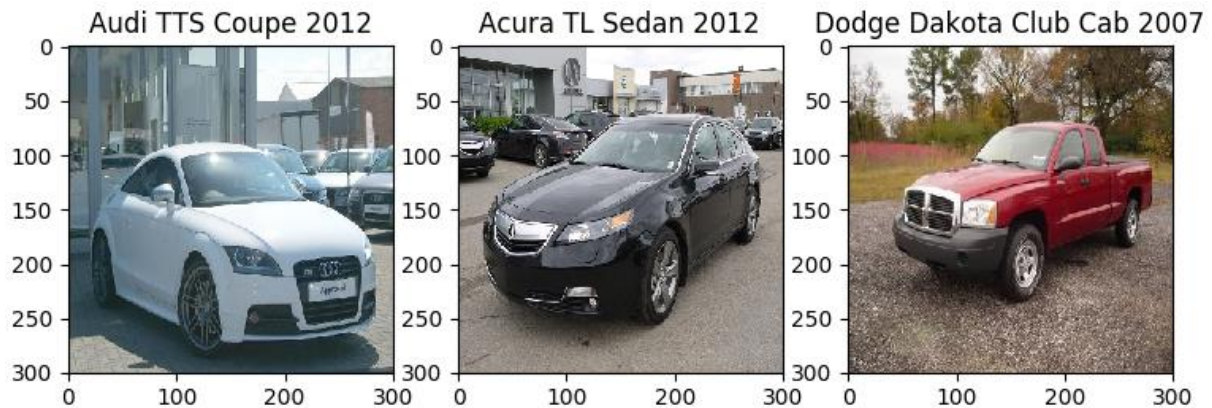


Figure 1: samples in training set

### Distribution of training set:

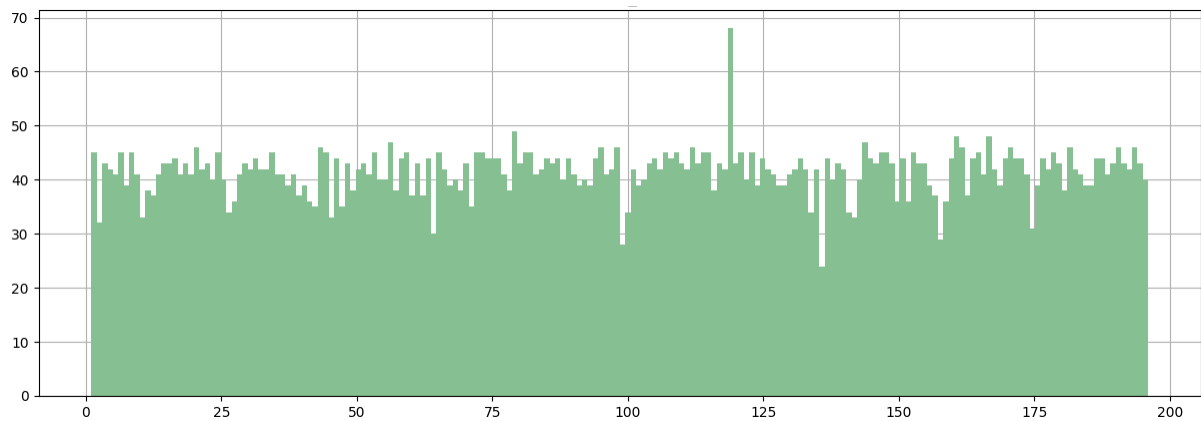


Figure 2: Distribution of training set

Min: 24 images/class, max: 68 images/class, mean: 41 images/class, so this dataset is quite balanced.

### III. TRAINING DETAILS

- Different CNN architectures, using pre-trained weights on imagenet dataset, with transfer learning to train the model. All layers will be fine tuned and the last fully connected layer will be replaced entirely. 7 main models in this solution: EfficientNetB1, EfficientNetB2, EfficientNetB3, ResNeXt101, DenseNet201, ResNet152V2, NASNetLarge.
- Different image input sizes: 224x224x3, 331x331x3, 240x240x3, 260x260x3, 300x300x3, 320x320x3, 384x384x3, 416x416x3
- Train with heavy augmentation:
  - random crops, horizontal flip, rotate, shear, AddToHueAndSaturation, AddMultiply, GaussianBlur, ContrastNormalization, sharpen, emboss
  - Random eraser [3]
  - Mixup [4]
- Using cyclical learning rates for training models [5]
- Cross-validation 5 folds
- Add dropout 0.5 at the last dense layer before output layer

#### IV. EVALUATE MODELS

To enhance the result, I applied 12 crops for validation and test prediction. Accuracy of single model is ensemble of 12 crops and 5 folds. For example with input shape of network is 224x224x3:



Figure 3: 12 crops for predicting

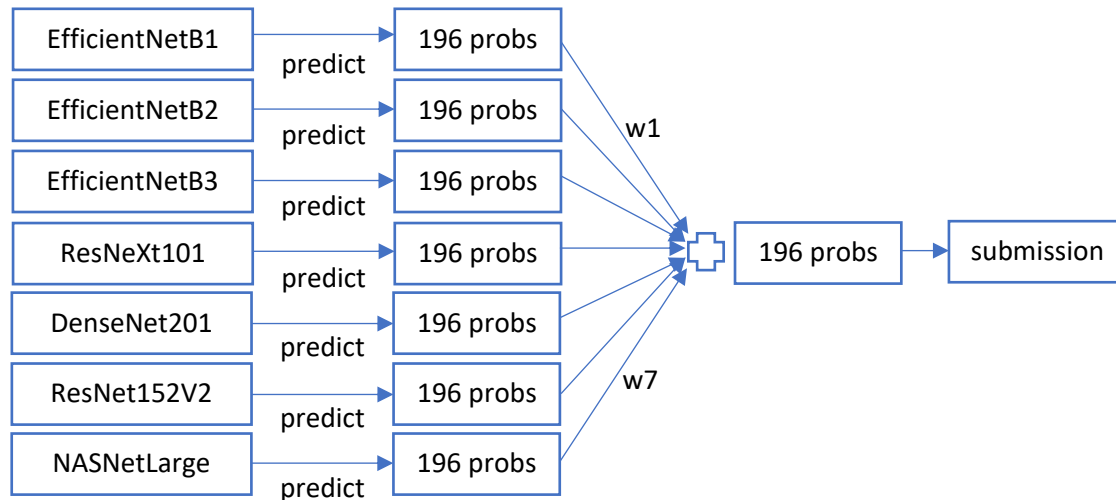
## Accuracy and size of 13 single models:

Models	Pretrain accuracy (imagenet)	Accuracy (stanford car)	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Size
ResNeXt101	0.787	Val-acc	0.8999	0.9196	0.9184	0.9214	0.9300	179 MB
		Val-acc 12 crops	0.9239	0.9349	0.9269	0.9269	0.9337	
		Test-acc 12 crops	0.9226	0.9304	0.9271	0.9236	0.9342	
		Test-acc ensemble 5 folds	0.9413					
DenseNet201	0.773	Val-acc	0.9073	0.9196	0.9104	0.9208	0.9232	83 MB
		Val-acc 12 crops	0.9177	0.9263	0.9171	0.9294	0.9330	
		Test-acc 12 crops	0.9224	0.9295	0.9169	0.9246	0.9245	
		Test-acc ensemble 5 folds	0.9396					
ResNeXt50	0.777	Val-acc	0.9073	0.9147	0.9190	0.9122	0.9128	102MB
		Val-acc 12 crops	0.9165	0.9319	0.9263	0.9257	0.9275	
		Test-acc 12 crops	0.9195	0.9289	0.9231	0.9248	0.9254	
		Test-acc ensemble 5 folds	0.9391					
NASNetLarge	0.825	Val-acc	0.8999	0.9085	0.9091	0.9104	0.9214	358 MB
		Val-acc 12 crops	0.9134	0.9202	0.9171	0.9257	0.9281	
		Test-acc 12 crops	0.9241	0.9195	0.9197	0.9235	0.9275	
		Test-acc ensemble 5 folds	0.9389					
EfficientNetB3	0.811	Val-acc	0.9067	0.9085	0.9141	0.9153	0.9214	51 MB
		Val-acc 12 crops	0.9110	0.9122	0.9171	0.9282	0.9312	
		Test-acc 12 crops	0.9239	0.9203	0.9209	0.9209	0.9277	
		Test-acc ensemble 5 folds	0.9352					
InceptionV3	0.779	Val-acc	0.9055	0.9091	0.9098	0.9184	0.9177	97MB
		Val-acc 12 crops	0.9159	0.9190	0.9196	0.9349	0.9287	
		Test-acc 12 crops	0.9218	0.9210	0.9132	0.9266	0.9197	
		Test-acc ensemble 5 folds	0.9347					
ResNet152V2	0.780	Val-acc	0.8858	0.9079	0.8920	0.8821	0.9036	243 MB
		Val-acc 12 crops	0.9128	0.9257	0.9208	0.9141	0.9244	
		Test-acc 12 crops	0.9179	0.9219	0.9151	0.9108	0.9205	
		Test-acc ensemble 5 folds	0.9342					
EfficientNetB2	0.798	Val-acc	0.9036	0.9018	0.9153	0.9012	0.9097	38 MB
		Val-acc 12 crops	0.9177	0.9147	0.9184	0.9153	0.9177	
		Test-acc 12 crops	0.9204	0.9217	0.9202	0.9126	0.9149	
		Test-acc ensemble 5 folds	0.9325					
Xception	0.790	Val-acc	0.8877	0.8956	0.9018	0.8969	0.9097	101 MB
		Val-acc 12 crops	0.9085	0.9110	0.9110	0.9159	0.9232	
		Test-acc 12 crops	0.9136	0.9185	0.9154	0.9110	0.9171	
		Test-acc ensemble 5 folds	0.9312					
MobileNetV2	0.713	Val-acc	0.8815	0.9006	0.8938	0.9024	0.9048	15 MB
		Val-acc 12 crops	0.8938	0.9122	0.9079	0.9184	0.9214	
		Test-acc 12 crops	0.9024	0.9141	0.9023	0.9102	0.9134	
		Test-acc ensemble 5 folds	0.9294					
InceptionResNetV2	0.803	Val-acc	0.8877	0.8926	0.8760	0.8883	0.9115	225 MB
		Val-acc 12 crops	0.9036	0.9110	0.8969	0.9177	0.9251	
		Test-acc 12 crops	0.9121	0.9126	0.8985	0.9113	0.9194	
		Test-acc ensemble 5 folds	0.9280					
EfficientNetB1	0.788	Val-acc	0.8956	0.9030	0.9036	0.9024	0.9023	33 MB
		Val-acc 12 crops	0.9006	0.9116	0.9141	0.9141	0.9165	
		Test-acc 12 crops	0.9105	0.9151	0.9118	0.9146	0.9152	
		Test-acc ensemble 5 folds	0.9276					
EfficientNetB0	0.763	Val-acc	0.8834	0.8987	0.8920	0.8735	0.8907	23 MB
		Val-acc 12 crops	0.8907	0.9122	0.9030	0.8932	0.9165	
		Test-acc 12 crops	0.9023	0.9070	0.9040	0.8935	0.9054	
		Test-acc ensemble 5 folds	0.9182					

Figure 4: Accuracy and size of 13 models

## V. ENSEMBLE

To boost accuracy, I ensemble 7 main model with suitable ratio



My result:

## Cars 196 Submission Site

Your account is **dungnb1333@gmail.com**. If this is incorrect, please contact [jkrause@cs.stanford.edu](mailto:jkrause@cs.stanford.edu).

Please provide your name

Name:

Submit results using the form below. **You must wait 24 hours between submissions.**

The current time is 2019/06/15 11:28:10

last upload: [Ensemble.txt](#). size: 35806 bytes. uploaded at 2019/06/13 10:17:43 Pacific time.

Accuracy: 94.6150976246736%

Result file: [here](#)

## VI. GUIDELINE

Step by step how to run my source code: <https://github.com/dungnb1333/stanford-cars-classification>

- Environments
  - o Ubuntu 16.04 LTS
  - o Cuda 10.0, cuDNN v7.5.0
  - o Python 3.5, Keras 2.2.4, Tensorflow 1.13.1, Efficientnet
  - o Quick install dependencies:
  - o \$ pip install --upgrade -r requirement.txt

- Step1: Prepare dataset

Download dataset

```
$ bash quick_download.sh
```

Cross-validation 5 folds

```
$ python3 prepare.py
```

- Step2: Training 7 main models:

```
$ python3 train.py --network EfficientNetB1 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network EfficientNetB2 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network EfficientNetB3 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network ResNeXt101 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network DenseNet201 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network ResNet152V2 --gpu 0 --epochs 200 --multiprocessing False
```

```
$ python3 train.py --network NASNetLarge --gpu 0 --epochs 200 --multiprocessing False
```

Note: my repository contains the checkpoint and logs of 13 models trained on Stanford-cars dataset.

If you don't want to train models, you just download them and put into folder checkpoints

Link checkpoints:

<https://www.dropbox.com/sh/jv7dbd5ksj2exun/AAATZFgaxe7rMEjv10PG1BYha?dl=0>

Link training logs:

<https://github.com/dungnb1333/stanford-cars-classification/tree/master/logs>

- Step3: Predict 7 models on stanford cars test set:

```
$ python3 predict.py --network EfficientNetB1 --gpu 0
```

```
$ python3 predict.py --network EfficientNetB2 --gpu 0
```

```
$ python3 predict.py --network EfficientNetB3 --gpu 0
```

```
$ python3 predict.py --network ResNeXt101 --gpu 0
```

```
$ python3 predict.py --network DenseNet201 --gpu 0
```

```
$ python3 predict.py --network ResNet152V2 --gpu 0
```

```
$ python3 predict.py --network NASNetLarge --gpu 0
```

Output: 7 network.txt in folder submission and 7 raw output network.npy in folder data

If you don't want to predict, just download raw output and put them to folder data

Link: <https://www.dropbox.com/sh/gem7sd15oc974w8/AACeMvoChN3GnFfRCeSBEoHAa?dl=0>

- Step4: Ensemble:

```
$ python3 ensemble.py
```

Output: submission file Ensemble.txt in folder submission.

Now you can submit it at stanford-cars evaluation server

<http://imagenet.stanford.edu/internal/car196/submission/submission.php>

- Demo on single image:

You should run on single model to save time, otherwise you can run on multiple models and ensemble results to achieve higher accuracy

```
$ python3 demo.py --network network --gpu gpu_id --image_path path --imshow True/False
```



For example:

```
$ python3 demo.py --network ResNeXt101 --gpu 0 --image_path images/samples/02381.jpg --imshow True
```



## VII. IMPORTANT NOTE FOR PRIVATE TEST SET

*I don't like this challenge because it's too unclear: not ranking, not provide private set (just images), not provide format of private test submission ... If candidate's solution runs very well on stanford cars set test (even higher accuracy than state-of-the-art) but your evaluators can't run ours models on a private test dataset by reasons like format of private test set ... That's really bad because we spent a lot of time for this challenge.*

If you run my solution on private test set, please

- replace images in folder **datasets/cars\_test** by your private test set, format of private test images must be "%05d.jpg"
- provide bounding box of car in private test images by replace **datasets/devkit/cars\_test\_annos.mat**
- replace line 6 (self.conf.TEST\_NUMS = 8041) in file **config.py** by number of private test images
- Then run `python3 prepare.py >> step3 >> step4` in GUIDELINE

## REFERENCES

- [1] State-of-the-art stanford cars dataset 2018 <https://paperswithcode.com/sota/fine-grained-image-classification-on-stanford>
- [2] State-of-the-art stanford cars dataset 2019 <https://paperswithcode.com/sota/image-classification-on-stanford-cars>
- [3] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, Yi Yang: Random Erasing Data Augmentation
- [4] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz: mixup - Beyond Empirical Risk Minimization
- [5] Leslie N. Smith: Cyclical Learning Rates for Training Neural Networks